



HDS-5960-02: Capstone Experience

**“A Supervised Machine Learning and NLP Framework for Predicting FDA
Pregnancy Drug Safety Categories Using Structured and Unstructured Drug
Data”**

Group 11

Final Project report

Group Members:

Sahithi Kalapala

Mohan Naik Palithya

Venkata Sai Pallavi Pallapolu

Harika Pamulapati

Group Advisors:

Prof. Divya Subramaniam and Prof. Dipti Subramaniam

[Project files](#)

[Application](#)

Abstract:

Background: Medication use during pregnancy is widespread, yet safety data remains limited due to ethical constraints in clinical trials and underreporting in post-marketing surveillance. Existing drug classification systems, including the FDA's discontinued A-X categories, often lack clarity and real-world applicability. This study proposes a machine learning (ML) framework to predict pregnancy drug risk categories by integrating structured pharmacologic attributes and unstructured patient-reported side effect narratives.

Methods: A publicly available dataset from Drugs.com containing 2,931 drug records was used. Structured features (e.g: drug class, Rx/OTC status) and unstructured side-effect texts underwent extensive preprocessing, including TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and one-hot encoding. Multiple models, Logistic Regression, SVM, KNN, Decision Trees, Random Forests, Gradient Boosting, XGBoost, and an Artificial Neural Network (ANN), were trained using stratified 80/20 splits. Hyperparameter tuning and Synthetic Minority Oversampling Technique (SMOTE) were applied selectively to optimize model performance. Interpretability was assessed using SHAP values, and topic modeling was performed via Non-negative Matrix Factorization (NMF) to extract latent themes from side-effect narratives.

Results: The SMOTE-enhanced Artificial Neural Network (ANN) outperformed all other models, achieving the highest test accuracy (90.94%), macro F1-score (0.9242), and micro-averaged AUC-ROC (0.9791). SHapley Additive exPlanations (SHAP) identified both structured variables (e.g: drug class, Rx/OTC status) and TF-IDF-derived side-effect terms as influential predictors. NMF uncovered coherent symptom clusters, such as neurological, gastrointestinal, and dermatological themes, that aligned with higher-risk pregnancy categories. In addition, chi-square and ANOVA tests confirmed significant associations between multiple drug metadata features and pregnancy risk classification, reinforcing the predictive value of the integrated data.

Conclusion: This study demonstrates that combining structured and unstructured drug data with supervised learning can yield accurate and interpretable predictions of pregnancy-related drug safety. Unlike traditional molecular models, this real-world, human-facing approach captures contextual nuances of medication use. Future work should focus on integrating trimester-specific data, electronic health records, and narrative-based labeling systems (PLLR) to enhance clinical applicability and support personalized maternal-fetal risk assessment.

Introduction: Ensuring drug safety during pregnancy is a critical concern due to the potential risks posed to both the mother and the developing fetus.^{1,2} Globally, it is estimated that up to 90% of pregnant women are prescribed or use at least one medication during their pregnancy to manage chronic illnesses, treat gestational complications, or address acute infections, concerns over fetal safety often outweigh therapeutic needs.^{1,3,4} However, the data guiding these decisions is frequently limited or inconclusive due to the systematic exclusion of pregnant women from randomized controlled trials (RCTs), driven by ethical, legal, and liability concerns.^{2,3,4} The resulting scarcity of human pregnancy exposure data forces clinicians to rely on retrospective observational studies, animal models, or post-marketing surveillance, each of which has notable limitations.^{1,2}

To bridge this knowledge gap, the U.S. Food and Drug Administration (FDA) introduced the Pregnancy Risk Classification system in 1979, assigning drugs to categories A through X based on available evidence.^{4,5} However, the system was widely criticized for being oversimplified, inconsistently applied, and poorly understood by clinicians and patients alike.^{2,4,6} These limitations led to the development of the Pregnancy and Lactation Labeling Rule (PLLR) in

2015, which replaced the category labels with narrative risk summaries intended to provide more contextual information.^{4,7} While the PLLR aimed to improve communication, it still does not offer easily accessible, structured, or trimester-specific risk classification needed for decision support.^{4,7}

Moreover, safety information for most drugs used in pregnancy remains incomplete, approximately 90% of medications approved since 1980 lack adequate human data to determine fetal risk, underscoring the limitations of existing labeling systems and registries.^{5,6} Post-marketing surveillance platforms such as FAERS and pregnancy exposure registries also suffer from underreporting and data fragmentation, making them less reliable for timely risk estimation.²

Given these constraints and to provide more accurate data-driven drug safety predictions, emerging computational tools, particularly supervised machine learning (ML), are being actively explored.^{3,8,9} ML models can learn complex patterns from diverse datasets and have shown promise in predicting adverse drug reactions (ADRs) and fetal risk outcomes.^{1,3,8} For instance, Wu et al. used ML to assess miscarriage risk in immunologically high-risk pregnancies, while Boland et al. applied Random Forest models to classify Category C drugs by linking molecular descriptors with clinical outcomes.^{8,10}

Despite these advances, most existing models rely heavily on molecular features like SMILES codes or structural fingerprints, which overlook how drugs are used in real-world settings.^{1,10} Peng et al. and Shtar et al. both acknowledged that these chemical-centric approaches often exclude key contextual variables such as side effects, indications, and patient-reported data, factors crucial to real-life risk assessment.^{1,3} Furthermore, limited model interpretability has hindered clinical adoption. While Shtar et al. integrated explainable AI (XAI) into a multimodal ML framework, their dependence on curated expert data limited scalability and real-time application.³

In parallel, transformer-based natural language processing (NLP) models like BERT have been used to extract pregnancy drug safety information from unstructured texts.¹¹ De Filippis et al. demonstrated that these systems can classify drugs into ordinal risk levels and detect trimester-specific risks by mining biomedical literature.¹¹ However, these NLP applications are more aligned with information retrieval than real-world risk prediction.

Despite the growing availability of real-world drug information on platforms like Drugs.com, data-driven approaches utilizing such metadata for predictive modeling in pregnancy drug safety remain underexplored. Ailes et al. used supervised learning to identify concerning drugs during pregnancy, but their work was descriptive and did not build predictive models.⁴

To bridge this gap, the present study proposes a supervised ML framework that uses structured and unstructured drug data from Drugs.com,¹² which is a real-world, human facing data to classify drugs into FDA pregnancy risk categories. By integrating both structured drug attributes and unstructured side effect narratives, the model aims to generate interpretable and clinically meaningful predictions that enhance decision-making in maternal-fetal medicine.

Methods:

Data Source and Study Population: This project utilized the publicly available “Drugs, Side Effects and Medical Condition” dataset,¹² a cross-sectional collection of prescription drug records and associated metadata. It comprises 2,931 records and 17 fields, occupying roughly 389 KB of memory. Although the exact data-collection timeframe is unspecified, it reflects information compiled up through December 2022. Each record describes a single medication, its intended indication, usage metrics, and safety metadata. Key fields include both structured and

unstructured data: medication names (Drug_Name, Generic_Name, Brand_Names), therapeutic classification (Drug_Classes), and target indication (Medical_Condition, with accompanying Medical_Condition_Description and URL links). Usage and user feedback are captured numerically in activity (percent usage), rating (average user score), and No_Of_Reviews. Regulatory attributes appear in Rx_Otc (prescription versus over-the-counter), Pregnancy_Category (FDA A-X classification),¹³ Csa (Controlled Substances Act schedule),¹⁴ and Alcohol (interaction flag). Adverse events are recorded in free-text form under Side_Effects, with supplementary URLs (Drug_Link, Medical_Condition_Url) and related drug listings (Related_Drugs). In total, the dataset comprises two numeric fields (Rating, No_Of_Reviews) and fifteen object-type fields that mix categorical entries and text narratives, enabling a combined ML and NLP approach.¹⁵

Variables and Operationalization: The target variable in this study is the FDA pregnancy-risk category (pregnancy_category), encoded as an ordered label ranging from “A” (safest) to “X” (contraindicated).¹³ The predictor variables are divided into two main sets: textual and structured features. The textual feature (X_text) consists of cleaned side-effect descriptions (Clean_Side_Effects). These narratives underwent natural language preprocessing, which included lowercasing, punctuation removal, stopword and domain-specific filler word filtering, and lemmatization to reduce words to their root forms. The structured features (X_other) include variables such as Drug_Classes (the therapeutic classification of the drug), Rx_Otc (indicating whether a drug is prescription-only or available over-the-counter), Medical_Condition (the condition for which the drug is prescribed), Csa (Controlled Substances Act schedule classification), and Alcohol_Interaction (a binary flag for known alcohol-drug interactions). Additionally, three auxiliary variables, Activity, Rating, and No_Of_Reviews, were retained for descriptive and exploratory analysis. Missing values in the Rating field were imputed with -1, and those in No_Of_Reviews were imputed with 0. However, these auxiliary variables were excluded from the final modeling pipeline.

Inclusion and Exclusion Criteria: Records were retained only if they contained non-missing values for the primary outcome (Pregnancy_Category) and the key textual predictor (Clean_Side_Effects). Any entry lacking data in either of these two core columns was excluded. Tertiary fields with excessive missingness (e.g: Medical_Condition_Description, Drug_Link, Medical_Condition_Url, Alcohol, and Related_Drugs) were also removed entirely from the dataset. After applying these filters, a total of 2,591 records remained for downstream analysis.

Data Preprocessing: Structured fields were first standardized and cleaned to ensure consistency and completeness. The dataset’s original 2,931 records and 17 columns were examined for missing values, revealing gaps in key features such as Side_Effects, Generic_Name, Brand_Names, Pregnancy_Category, Alcohol, Related_Drugs, Rating, and No_Of_Reviews. Non-essential columns with excessive missingness, Medical_Condition_Description, Drug_Link, Medical_Condition_Url, and Related_Drugs, were dropped. The alcohol column was transformed into a binary alcohol_interaction feature by filling missing values with “No” and converting entries marked as “X” to “Yes”, indicating the presence of known alcohol-drug interactions. Remaining categorical variables (Brand_Names, Generic_Name, Drug_Classes) had null entries imputed with “Unknown,” while rating and No_Of_Reviews were filled with -1 and 0, respectively, to distinguish missingness from genuine values. The activity field’s percent signs were stripped and converted to integers, and inconsistent capitalization across Medical_Condition entries (e.g: “Copd” to “COPD”) was corrected. Finally, all column names

were harmonized to the title case and underscored replaced with spaces, yielding a uniform, analysis-ready table.

Unstructured side-effects text underwent a multi-step NLP pipeline.¹⁵ Free-text entries in Side_Effects were lowercased and stripped of non-alphabetic characters, followed by removal of English stopwords and drug-specific filler terms. SciSpaCy was then applied to perform lemmatization,¹⁶ reducing each token to its base form (for example, “swelling” to “swell”). A new feature, Side_Effects_Len, was calculated to capture the character length of each cleaned description, offering a numerical proxy for textual complexity. After these cleaning and transformation steps, the dataset consisted of 2,591 complete records across 13 refined features, fully prepared for modeling.

Statistical Analysis: To examine associations between the pregnancy risk category and key explanatory variables, inferential statistical tests were conducted prior to model development. Chi-square tests were used to assess the relationship between the categorical outcome (Pregnancy_Category) and five structured categorical predictors: Controlled Substance Act schedule (Csa), prescription/OTC status (Rx_Otc), alcohol interaction flag, therapeutic drug class, and targeted medical condition.¹⁷ For continuous variables (Rating, No_Of_Reviews, and Activity), one-way Analysis of Variance (ANOVA) was performed to evaluate mean differences across pregnancy categories.¹⁷ In both tests, a significance threshold of 0.05 was used. Additionally, 95% confidence intervals were computed for chi-square statistics and ANOVA group means to quantify uncertainty and support reproducibility. These statistical assessments helped identify patterns of association and informed subsequent modeling decisions, particularly around variable importance and potential confounding.

Feature engineering: All categorical inputs were transformed into machine-readable formats using one-hot encoding for the five structured predictors,¹⁸ drug class, Controlled Substances Act schedule, prescription versus OTC status, medical condition, and alcohol-interaction flag, ensuring each category became a separate binary feature. The pregnancy-risk outcome itself was label-encoded into ordinal integers (0 through 5) to satisfy algorithm requirements.¹⁸ Side-effects narratives were converted into a TF-IDF (Term Frequency-Inverse Document Frequency) matrix via Scikit-learn’s CountVectorizer and TfidfTransformer,^{19,20} capturing term importance while down-weighting common but uninformative words. Two continuous variables, drug activity percentage and review count, were standardized with z-score scaling to align their ranges and improve model convergence.^{21,22} A numeric feature representing the length of each cleaned side-effects entry (Side_Effects_Len) was also retained to reflect textual complexity as a potential risk indicator. Because of the pronounced imbalance in pregnancy categories, Synthetic Minority Oversampling Technique (SMOTE) was applied selectively to the training data of the final,^{23,24} best-performing model; this targeted resampling was chosen to effectively equalize class frequencies and strengthen minority-class predictions without overfitting. Together, these engineered features produced a unified, high-dimensional dataset ready for robust multi-class classification.

Train/Test Split: Employed a stratified 80/20 split, preserving the pregnancy-category distribution, via Scikit-learn’s train_test_split() (random_state = 42). This yielded (X_text_train, X_text_test, X_other_train, X_other_test, y_train, y_test) for downstream modeling.

Model development and evaluation: A standardized training pipeline was implemented via a reusable train_and_evaluate() utility function. This function accepts any Scikit-learn or Keras classifier, fits it on the training data, generates predictions on both training and hold-out test sets, and produces consistent evaluation outputs (classification reports and confusion matrices). By

encapsulating data fitting, prediction, and metric calculation, this design streamlined comparisons across diverse algorithms under a uniform interface.

Multiple classes of machine learning algorithms were evaluated under this framework. For linear and kernel-based methods, a multinomial logistic regression model was trained using L2 regularization.²⁵ To address the pronounced class imbalance in the pregnancy category outcome, class weights were set inversely proportional to class frequencies. Hyperparameter tuning was conducted using GridSearchCV, exploring different values of the regularization strength (C) and solver types, with optimization based on the weighted F1-score and stratified cross-validation.²⁵ Support Vector Machines (SVMs) were also trained using both linear and radial basis function (RBF) kernels.²⁶ Similar to logistic regression, the SVM models employed grid search to optimize key hyperparameters, including the penalty parameter (C), kernel type, and kernel coefficient (gamma), with balanced class weights to mitigate skewed class distributions.²⁶ Instance-based learning was implemented using K-Nearest Neighbors (KNN) classifiers.²⁷ These models were trained across a range of k-values from 1 to 30, while grid search explored combinations of distance metrics (Euclidean vs. Manhattan) and weighting schemes (uniform vs. distance-based).²⁷ Stratified folds were employed to ensure robust performance estimation even in the presence of class imbalance.

For tree-based methods, a single decision tree classifier served as the initial model, with an initial maximum depth and balanced class weights.²⁸ This was followed by hyperparameter optimization involving depth, minimum samples per split, and splitting criteria.²⁸ Building on this, a Random Forest ensemble of 1,000 trees was trained. Hyperparameters such as tree depth, minimum samples per node, and class weighting schemes were tuned through grid search to maximize model performance.²⁹ Gradient Boosting (GB) and Extreme Gradient Boosting (XGBoost) classifiers were also developed, with comprehensive hyperparameter searches conducted over learning rates, number of estimators, and maximum tree depth.^{30,31} Both models incorporated early stopping and customized class weights to improve learning in minority classes.

The deep learning component of this study involved constructing a feedforward Artificial Neural Network (ANN) using TensorFlow and Keras.³² The model architecture consisted of multiple dense layers with ReLU activation functions, dropout regularization to prevent overfitting, and a final softmax output layer suitable for multiclass classification.³² Class weights were applied to the loss function to counter class imbalance, and early stopping monitored validation loss to prevent overfitting. Integration with scikit-learn was achieved via a custom KerasClassifier wrapper, enabling RandomizedSearchCV over key hyperparameters, number of layers, neuron counts, dropout rates, batch sizes, and learning rates, using stratified cross-validation for robust model selection.

By applying this consistent methodology, baseline modeling, systematic hyperparameter tuning, and evaluation under a unified framework, each algorithm's capacity to learn from combined structured and TF-IDF-vectorized text features could be fairly compared, informing the final model choice for deployment. The final model selection was based on performance across accuracy, F1-score, and AUC metrics, with consideration for model interpretability and clinical relevance.

Interpretability and Visualization: A comprehensive set of exploratory and interpretability visualizations was integrated to illuminate both data characteristics and model behavior.

Exploratory Analysis: To gain a comprehensive understanding of the dataset and inform downstream preprocessing and modeling decisions, a series of exploratory data analysis (EDA)

techniques were conducted. Visualizations such as pie charts, bar plots, and histograms were generated using Matplotlib and Seaborn to examine the marginal distributions of key categorical variables, including FDA pregnancy risk categories, Controlled Substances Act (CSA) schedules, and associated medical conditions. Similarly, the distributions of continuous variables, such as activity percentages, drug ratings, and the number of user reviews, were analyzed to identify skewness, outliers, and inconsistencies.

These initial findings played a key role in shaping important preprocessing decisions. For instance, skewed distributions in review counts and activity levels prompted consideration of log-transformations to reduce variance. The visual evidence of class imbalance, particularly among pregnancy risk categories, motivated the later use of resampling techniques like SMOTE during model development.

Beyond univariate analysis, cross-feature relationships were explored through heatmaps and scatter plots. Relationships between variables such as Rx/OTC status and alcohol interaction flags, as well as activity levels and user ratings, were examined to uncover potential dependencies or confounding patterns. These analyses not only highlighted redundancies and correlations but also influenced feature engineering decisions by identifying which variables might carry overlapping or additive predictive power.

Topic Modeling & Word Clouds: Non-negative Matrix Factorization (NMF) was applied to the TF-IDF-vectorized side-effects texts to uncover the latent themes.³³ The top terms per topic were rendered as word clouds, offering an intuitive overview of dominant symptom clusters, ranging from respiratory emergencies to dermatological reactions, prior to any supervised modeling.

Feature Importance (SHAP): To interpret the predictions made by the best-performing model, SHAP (SHapley Additive exPlanations) was employed as a model-agnostic interpretability tool. A representative background sample was drawn from the resampled training data and supplied to a DeepExplainer, allowing the model to estimate baseline expectations for feature contributions. SHAP values were then calculated on the test set to determine how individual features, both textual and structured, impacted the model's output, without directly revealing performance metrics. This approach highlighted the influence of TF-IDF-weighted tokens (such as clinical terms found in side effect narratives) and one-hot encoded categorical variables, including drug class and Rx/OTC status. A summary plot displaying the top twenty contributing features confirmed that both text-derived and structured attributes played a meaningful role in pregnancy category prediction. To ensure that the interpretability remained consistent and unbiased after addressing class imbalance, this SHAP analysis was repeated on the SMOTE-enhanced data, validating that the resampling strategy did not distort the model's explanation patterns.

Model Calibration & Confidence Visualization: Prediction confidence was visualized by sorting test instances according to their maximum predicted probability and marking correct vs. incorrect outcomes. A reference line at the six-class random guess probability (≈ 0.167) contextualized model certainty. This plot facilitated assessment of where the classifier was most and least confident, crucial for high-stakes deployment.

Global Discrimination (ROC-AUC): Micro-averaged, one-vs-rest ROC curves were plotted for each candidate algorithm to compare overall discriminative patterns across multiple classes.³⁵ Although detailed AUC values are omitted here, these curves informed selection of top models based on their ability to separate risk categories.

Class Distribution Before & After SMOTE: To ensure unbiased comparison of baseline model performance, all candidate models were trained and evaluated on the original, imbalanced dataset. SMOTE was applied selectively to the training set of the final, best-performing model to

improve minority class representation without compromising generalizability.^{23,24} This approach is consistent with best practices in imbalanced classification, where synthetic oversampling is often reserved for deployment-phase model optimization. To demonstrate the impact of SMOTE,^{23,24} side-by-side count plots illustrated original vs. resampled training-set class frequencies. This confirmed the achievement of balanced representation across pregnancy categories, a prerequisite for fair model learning. Together, these visualization and interpretability steps ensured transparent exploration of the data's underlying structure and clear communication of how each feature influenced the final risk-classification models.

Results:

Statistical Analysis: All five categorical variables, CSA schedule, Rx/OTC status, Alcohol Interaction, Drug Class, and Medical Condition, were found to be significantly associated with pregnancy category ($p < 0.001$) after chi-square test, confirming their predictive relevance (see Table 1). Drug Class and Medical Condition, in particular, yielded notably high chi-square statistics (9729.81 and 5046.65, respectively), indicating strong associations with pregnancy risk classification. Among the numeric variables, significant group-level differences in rating, number of reviews, and activity were observed across pregnancy categories ($p < 0.001$) after running one-way ANOVA. For instance, mean drug ratings varied from 3.23 to 4.78 across groups, as shown in Table 2, suggesting a possible link between perceived drug effectiveness and assigned pregnancy risk.

Table 1: Chi-Square Test Results for Categorical Features and Their Association with Pregnancy Risk Categories (Chi-square statistic=Chi², degrees of freedom=df, Confidence Intervals=CI)

| Variable | Chi2 | df | p-value | 95% CI (Chi ²) | Significant |
|---------------------|---------|------|---------|----------------------------|-------------|
| Csa | 157.47 | 25 | 0.0 | [13.12, 40.65] | Yes |
| Rx_Otc | 793.72 | 10 | 0.0 | [3.25, 20.48] | Yes |
| Alcohol_Interaction | 67.01 | 5 | 0.0 | [0.83, 12.83] | Yes |
| Drug_Classes | 9729.81 | 1315 | 0.0 | [1216.39, 1417.39] | Yes |
| Medical_Condition | 5046.65 | 230 | 0.0 | [189.89, 273.9] | Yes |

Table 2: ANOVA Results for Continuous Features Across Pregnancy Risk Categories (F-statistic = F-stat, confidence interval = CI)

| Variable | F-stat | p-value | 95% CI (Group Means) | Significant |
|---------------|--------|---------|---|-------------|
| Rating | 12.72 | 0.0 | 0: 4.78 [3.07, 6.48]; 1: 3.23 [2.84, 3.61]; 2: 3.46 [3.23, 3.69]; 3: 4.09 [3.55, 4.64]; 4: 2.21 [1.81, 2.62]; 5: 5.12 [4.48, 5.77] | Yes |
| No_Of_Reviews | 13.58 | 0.0 | 0: 70.50 [9.28, 131.72]; 1: 34.07 [22.42, 45.72]; 2: 45.26 [38.12, 52.41]; 3: 51.13 [33.57, 68.70]; 4: 14.80 [9.49, 20.11]; 5: 133.98 [65.29, 202.67] | Yes |

| | | | | |
|----------|-------|-----|--|-----|
| Activity | 15.85 | 0.0 | 0: 16.38 [2.56, 30.19]; 1: 13.31 [11.28, 15.34]; 2: 8.05 [7.18, 8.92]; 3: 9.57 [7.25, 11.90]; 4: 3.72 [2.72, 4.71]; 5: 11.19 [7.69, 14.69] | Yes |
|----------|-------|-----|--|-----|

Topic Modeling on Side-Effect Narratives: The NMF model yielded 20 interpretable topics from the side-effect narratives, many aligning with clinically meaningful symptom clusters. Prominent themes included neurological complaints (e.g: “headache,” “dizziness,” “seizure”), gastrointestinal effects (e.g: “nausea,” “vomiting,” “diarrhea”), and dermatological reactions (e.g: “rash,” “itch,” “redness”). These topics were disproportionately associated with moderate to lower-risk drug categories (e.g: B, C, N), suggesting a potential link between adverse effect profiles and pregnancy risk perception. Visual exploration using word clouds enhanced interpretability and contributed to downstream modeling. Figure 1 illustrates four representative topic clusters, emphasizing thematic coherence across the extracted side-effect patterns.



Fig 1: Representative Word Clouds of Four Side-Effect Topics Identified by NMF

Model Performance Metrics: A comprehensive set of machine learning models were evaluated to predict pregnancy risk categories using structured attributes and TF-IDF-transformed side-effect narratives, including Logistic Regression, SVM, KNN, Decision Tree, Random Forest, Gradient Boosting, XGBoost, and an ANN implemented with TensorFlow and Keras. Each model underwent two evaluation stages: baseline assessment using default hyperparameters and post-tuning assessment following optimization via randomized or grid search. To address class imbalance, balanced class weights were applied across models, and SMOTE was used exclusively in the training phase of the final selected model.^{23,24} Performance metrics, including test accuracy, macro-averaged precision, recall, F1-score, and micro-averaged AUC-ROC, were computed using stratified cross-validation (k=3, except k=2 for XGBoost and Gradient Boosting due to computational constraints). As detailed in Tables 3 and 4, the SMOTE-enhanced tuned Artificial Neural Network achieved the highest performance, with a test accuracy of 90.94%, a macro F1-score of 0.9242, and an AUC of 0.9791, outperforming all other models across most metrics.

Table 3: Pre-Tuning Performance Metrics of Machine Learning Models

| Model | Test Accuracy | Macro Precision | Macro Recall | Macro F1-Score | AUC-ROC |
|---------------------|---------------|-----------------|--------------|----------------|---------|
| Logistic Regression | 0.8940 | 0.8949 | 0.9206 | 0.9066 | 0.9755 |

| | | | | | |
|----------------------------------|--------|--------|--------|--------|--------|
| SVM | 0.8921 | 0.9224 | 0.8789 | 0.8976 | 0.9820 |
| KNN | 0.7726 | 0.8125 | 0.7858 | 0.7955 | 0.9439 |
| Decision Tree | 0.8304 | 0.8490 | 0.8153 | 0.8294 | 0.8855 |
| Random Forest | 0.8631 | 0.9091 | 0.8499 | 0.8742 | 0.9782 |
| Gradient Boost | 0.8285 | 0.8957 | 0.8148 | 0.8477 | 0.9724 |
| XGBoost | 0.8882 | 0.9376 | 0.8667 | 0.8969 | 0.9829 |
| Artificial Neural Network | 0.8535 | 0.8488 | 0.8886 | 0.8649 | 0.9731 |

Table 4: Post-Tuning Performance Metrics of Machine Learning Models

| Model | Test Accuracy | Macro Precision | Macro Recall | Macro F1-Score | AUC-ROC |
|----------------------------------|----------------------|------------------------|---------------------|-----------------------|----------------|
| Logistic Regression | 0.8940 | 0.8951 | 0.9219 | 0.9072 | 0.9748 |
| SVM | 0.8843 | 0.8960 | 0.9095 | 0.9012 | 0.9792 |
| KNN | 0.8786 | 0.8920 | 0.8749 | 0.8819 | 0.9316 |
| Decision Tree | 0.8362 | 0.8661 | 0.8163 | 0.8372 | 0.8861 |
| Random Forest | 0.8766 | 0.9317 | 0.8469 | 0.8823 | 0.9863 |
| Gradient Boost | 0.8227 | 0.8730 | 0.8188 | 0.8404 | 0.9629 |
| XGBoost | 0.8670 | 0.9256 | 0.8336 | 0.8696 | 0.9811 |
| Artificial Neural Network | 0.8959 | 0.9102 | 0.9098 | 0.9053 | 0.9825 |
| SMOTE applied ANN | 0.9094 | 0.9239 | 0.9273 | 0.9242 | 0.9791 |

Pre-Tuning Analysis: Before tuning, Logistic Regression, SVM, and XGBoost emerged as top performers, achieving test accuracies of 0.8940, 0.8921, and 0.8882, respectively. XGBoost led in macro-average precision (0.9376) and F1-score (0.8969), showcasing its strength in handling the high-dimensional, hybrid input space comprising TF-IDF-transformed side effect narratives and one-hot encoded categorical variables. The Neural Network also performed competitively, reaching a test accuracy of 0.8535 and macro F1-score of 0.8649, with a strong AUC-ROC of 0.9731. Simpler models such as KNN (accuracy: 0.7726) and Decision Tree (0.8304) showed weaker performance, likely due to their sensitivity to noise and limited capacity to model nonlinear decision boundaries in a complex, multiclass setting. Gradient Boosting performed moderately, with a test accuracy of 0.8285 and macro F1-score of 0.8477.

Post-Tuning Analysis: After hyperparameter tuning, the Artificial Neural Network (ANN) emerged as the best-performing model among the baseline models, achieving a test accuracy of 0.8959, macro F1-score of 0.9053, and an AUC-ROC of 0.9825, highlighting its strong generalization and balanced classification capability. Logistic Regression also performed consistently well, with a test accuracy of 0.8940 and macro F1-score of 0.9072, confirming that its default configuration was already near-optimal and resilient to overfitting. Random Forest demonstrated notable improvements in macro precision (0.9317) and macro F1 (0.8823), with an AUC-ROC of 0.9863, suggesting that tuning enhanced its ability to capture minority class patterns.

K-Nearest Neighbors (KNN) benefited significantly from tuning, improving from lower baseline metrics to a test accuracy of 0.8786 and F1-score of 0.8819, due to better selection of k , weighting strategies, and distance metrics. XGBoost experienced a slight decline in test accuracy (0.8670) and F1-score (0.8696), likely due to regularization-induced underfitting, though it retained a strong AUC (0.9811). Similarly, Support Vector Machine (SVM) and Gradient Boosting saw marginal reductions in test accuracy and F1-score post-tuning, indicating diminishing returns from hyperparameter optimization. Decision Tree, while showing a modest gain in macro precision (0.8661), continued to underperform compared to ensemble and deep models.

Crucially, SMOTE-enhanced ANN surpassed all models, achieving the highest performance across nearly all metrics: test accuracy of 0.9094, macro precision of 0.9239, macro recall of 0.9273, macro F1-score of 0.9242, and an AUC-ROC of 0.9791. These results demonstrate how addressing class imbalance through SMOTE not only stabilized predictions for minority categories but also improved the model's overall robustness. This configuration was selected as the final optimal model for its balanced and superior performance across all six pregnancy safety categories.

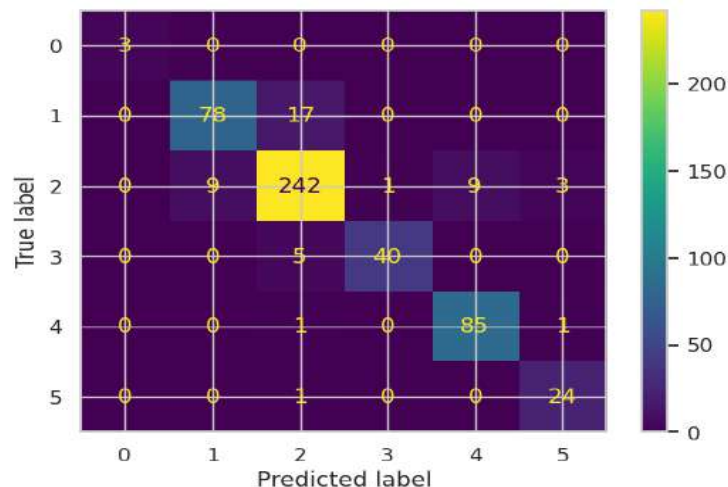


Fig 2: Confusion Matrix for SMOTE-applied Tuned ANN Model

ROC-AUC Curve Analysis: As shown in Figure 3, the tuned Artificial Neural Network (ANN), XGBoost, and Random Forest models yielded the highest AUC scores, all exceeding 0.98. Among them, the ANN model demonstrated the most consistent performance across all six pregnancy categories, with minimal variation between classes.

Figure 4 illustrates the ROC curve for the final model, SMOTE-applied tuned ANN, which achieved a micro-average AUC of 0.98. The curve's steep rise and proximity to the top-left corner confirm its strong true positive rate and low false positive rate across classes. The uniform performance across imbalanced classes highlights the effectiveness of SMOTE in enhancing minority class representation, ultimately reinforcing the model's robustness and clinical potential.

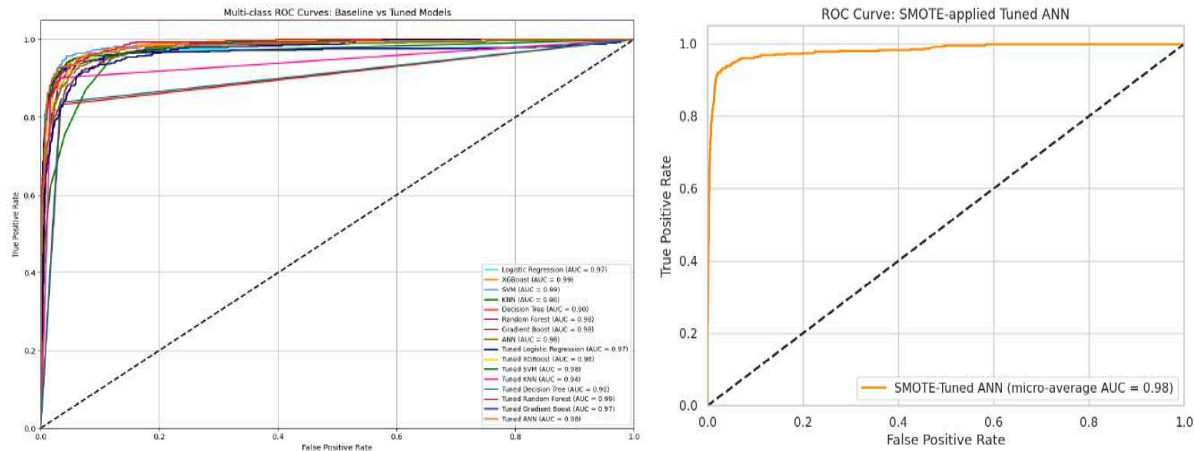


Figure 3: Multi-class ROC-AUC Curves for All Baseline and Tuned Models (left)

Figure 4: ROC-AUC Curve for Final SMOTE-applied Tuned ANN Model (right)

Final Model and Impact of SMOTE: The Artificial Neural Network (ANN) outperformed all other models post-tuning, achieving a test accuracy of 0.8959, macro F1-score of 0.9053, and AUC-ROC of 0.9825, indicating strong generalization and class-wise discrimination. To further mitigate the impact of class imbalance, SMOTE was selectively applied to the training set of the tuned ANN model.^{23,24} As visualized in Figure 5, the class distribution before and after SMOTE application shows a substantial correction of the original imbalance, with all six pregnancy categories becoming equally represented in the resampled training set.

This oversampling led to modest yet meaningful performance gains: test accuracy increased from 0.8959 to 0.9094, and macro precision, recall, and F1-score stabilized at 0.91. Notably, improvements were observed in the minority classes (particularly Class 1 and Class 5), without degrading performance on majority classes. These enhancements, alongside strong generalization (train accuracy: 0.9948), established the SMOTE-enhanced ANN as the final optimal model for predicting pregnancy drug safety categories based on both structured and unstructured features.

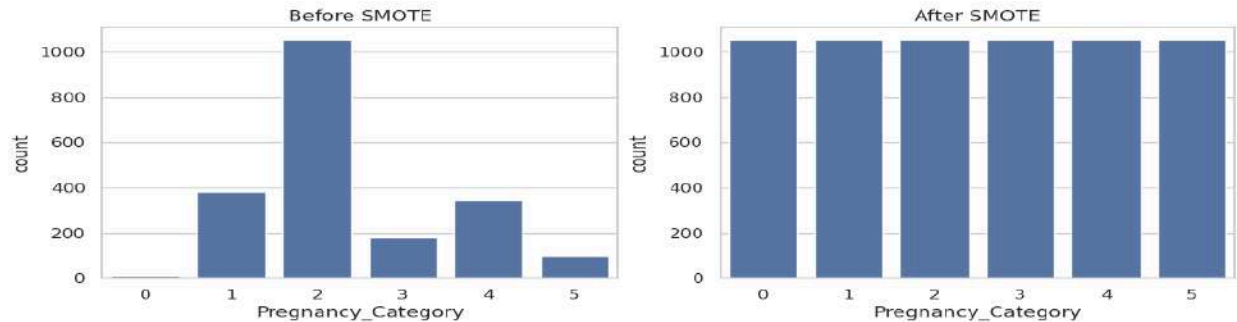


Figure 5: Class distribution of pregnancy safety categories before and after SMOTE application

Feature Importance Analysis: Feature contributions in the final SMOTE-applied, hyperparameter-tuned ANN model were evaluated using SHAP values.³⁴ The multi-class SHAP summary plot (Figure 6) highlights the top predictors influencing model decisions across the six pregnancy risk categories. Categorical variables, especially drug classes such as thyroid drugs (cat_Drug_Classes_235), and upper respiratory combinations (cat_Drug_Classes_256), and medical conditions like hypothyroidism (cat_Medical_Condition_29), and hayfever (cat_Medical_Condition_26), showed high impact on prediction outputs. The type of prescription (OTC(cat_Rx_Otc_0), and prescription(cat_Rx_Otc_1)) also contributed meaningfully to classification. Several text-based features derived from side-effect narratives, such as “temporary,” “hair,” “insomnia,” “dry,” and “difficulty,” emerged as important, indicating the model’s ability to integrate linguistic cues from patient-reported data. These features particularly influenced Class 0 and Class 2 predictions, helping distinguish safer drugs from higher-risk categories. This analysis supports the effectiveness of combining structured and unstructured features to enhance interpretability and predictive performance.

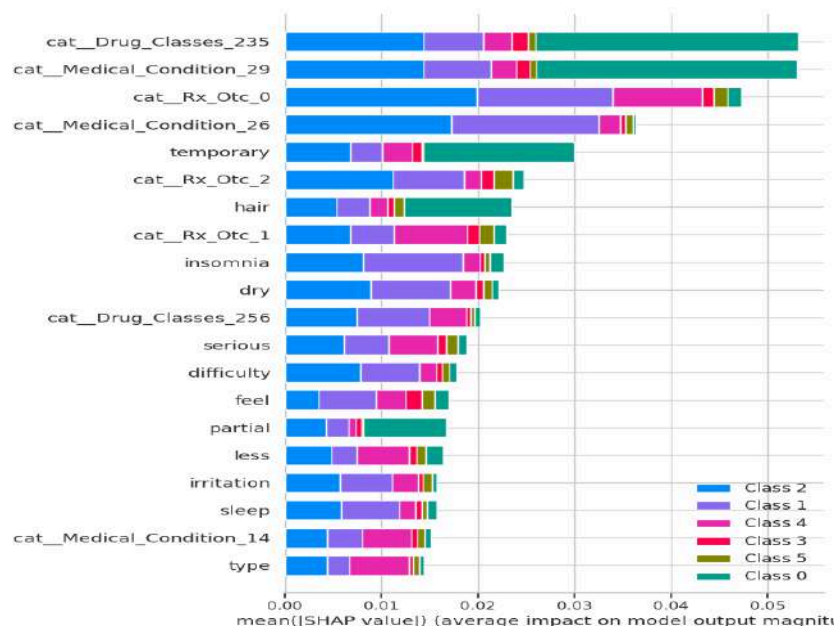


Figure 6: SHAP Summary Plot of Top Predictors in the Final SMOTE-Enhanced ANN Model

Discussion: This study explored a machine learning framework for predicting FDA pregnancy drug safety categories using a combination of structured pharmacologic attributes and unstructured side-effect narratives. By integrating traditional classification algorithms with modern deep learning models, we identified the SMOTE-enhanced Artificial Neural Network (ANN) as the best-performing model. It achieved a test accuracy of 90.94%, macro F1-score of 0.9242, and micro-average AUC of 0.9791, outperforming all baseline and tuned models.

The strong performance of the final ANN model illustrates the effectiveness of combining structured features (e.g: drug class, Rx/OTC status) with patient-reported text, particularly when class imbalance is addressed through SMOTE. Unlike prior studies that relied heavily on curated molecular descriptors or chemical structures alone, which often fail to capture real-world drug behavior and patient variability,^{1,8,10} this work underscores the value of contextual and human-facing metadata. SHAP-based interpretability further reinforced clinical relevance, with

top predictors reflecting meaningful signals such as severe medical conditions, insomnia, and regulatory classifications.

Topic modeling of side-effect narratives revealed clinically coherent clusters, dermatologic, gastrointestinal, systemic, and advisory, which mapped meaningfully to pregnancy risk levels. These latent patterns, derived from TF-IDF, vectorized patient narratives, served as soft signals for model learning, complementing structured fields. Additionally, chi-square and ANOVA analyses demonstrated statistically significant associations between drug class, medical condition, review ratings, and pregnancy category, reinforcing the predictive utility of real-world variables.

Despite its promise, this study has limitations. The dataset lacks trimester-specific annotations, dosage information, or clinical outcomes, critical variables for personalized maternal-fetal risk estimation. It also employs the outdated FDA A-X classification system, which has since been replaced by the PLLR narrative framework due to concerns over ambiguity and oversimplification.^{4,5,7} These constraints limit the direct clinical applicability of the model in its current form.

To address these gaps, future research should integrate electronic health records (EHRs), pharmacovigilance databases, and longitudinal outcome data to validate predictions in a clinical setting. Enhanced interpretability may also be achieved through biomedical transformers such as BioBERT or ClinicalBERT,¹¹ which can capture richer semantic features from clinical narratives. Moreover, aligning models with trimester-specific risk stratification and the PLLR narrative structure could better support clinical decision-making at the point of care.^{4,7}

In conclusion, this study demonstrates that supervised machine learning, especially when powered by both structured drug metadata and patient-facing text, can provide meaningful and interpretable predictions of pregnancy drug risk. Unlike chemical-centric approaches that lack contextual awareness,^{1,3,10} our framework reflects real-world usage patterns and patient experiences. While further validation and clinical integration are required, the results establish a scalable foundation for data-driven pharmacovigilance and maternal medication safety tools in obstetric care.

References:

1. Peng J, Fu L, Yang G, Cao D. Advanced AI-Driven Prediction of Pregnancy-Related Adverse Drug Reactions. *Journal of Chemical Information and Modeling*. 2024;64(24):9286-9298. doi:<https://doi.org/10.1021/acs.jcim.4c01657>
2. Kennedy D, Batagol R. Drug safety in pregnancy. *Australian Prescriber*. 2025;48(1):5-9. doi:<https://doi.org/10.18773/austprescr.2025.008>
3. Shtar G, Rokach L, Shapira B, Kohn E, Berkovitch M, Berlin M. Explainable multimodal machine learning model for classifying pregnancy drug safety. Wren J, ed. *Bioinformatics*. 2021;38(4):1102-1109. doi:<https://doi.org/10.1093/bioinformatics/btab769>
4. Ailes EC, Zimmerman J, Lind JN, et al. Using Supervised Learning Methods to Develop a List of Prescription Medications of Greatest Concern during Pregnancy. *Maternal and Child Health Journal*. 2020;24(7):901-910. Doi: <https://doi.org/10.1007/s10995-020-02942-2>
5. Law R, Bozzo P, Koren G, Einarson A. FDA pregnancy risk categories and the CPS: do they help or are they a hindrance?. *Can Fam Physician*. 2010;56(3):239-241.
6. Chambers CD, Polifka JE, Friedman JM. Drug safety in pregnant women and their babies: ignorance not bliss. *Clin Pharmacol Ther*. 2008;83(1):181-183.
7. Sahin L, Nallani SC, Tassinari MS. Medication use in pregnancy and the pregnancy and lactation labeling rule. *Clin Pharmacol Ther*. 2016;100(1):23-25. doi:10.1002/cpt.380
8. Wu Y, Yu X, Li M, et al. Risk prediction model based on machine learning for predicting miscarriage among pregnant patients with immune abnormalities. *Frontiers in pharmacology*. 2024;15. Doi: <https://doi.org/10.3389/fphar.2024.1366529>
9. Di Filippo JJ, Bollini M, Cavasotto CN. A Machine Learning Model to Predict Drug Transfer Across the Human Placenta Barrier. *Frontiers in Chemistry*. 2021;9. Doi: <https://doi.org/10.3389/fchem.2021.714678>
10. Boland MR, Polubriaginof F, Tatonetti NP. Development of A Machine Learning Algorithm to Classify Drugs Of Unknown Fetal Effect. *Scientific Reports*. 2017;7(1). Doi: <https://doi.org/10.1038/s41598-017-12943-x>
11. Filippis R de, Abdullah AI Foysal. AI-Powered NLP Framework for Extracting Drug Safety Information in Pregnancy. *OALib*. 2025;12(06):1-21. Doi: <https://doi.org/10.4236/oalib.1113509>
12. Varghese J. Drugs, Side Effects and Medical Condition. Kaggle.com. Published 2022. Accessed May 29, 2025. <https://www.kaggle.com/datasets/jithinanievarghese/drugs-side-effects-and-medical-condition/data>
13. FDA Pregnancy Categories - CHEMM. Hhs.gov. Published 2021. <https://chemm.hhs.gov/pregnancycategories.htm>
14. DEA. Drug Scheduling. www.dea.gov. Published 2025. <https://www.dea.gov/drug-information/drug-scheduling>
15. DeepLearning.AI. Natural Language Processing (NLP) - A Complete Guide. www.deeplearning.ai. Published January 11, 2023. <https://www.deeplearning.ai/resources/natural-language-processing/>
16. Lemmatizer · spaCy API Documentation. Lemmatizer. <https://spacy.io/api/lemmatizer>

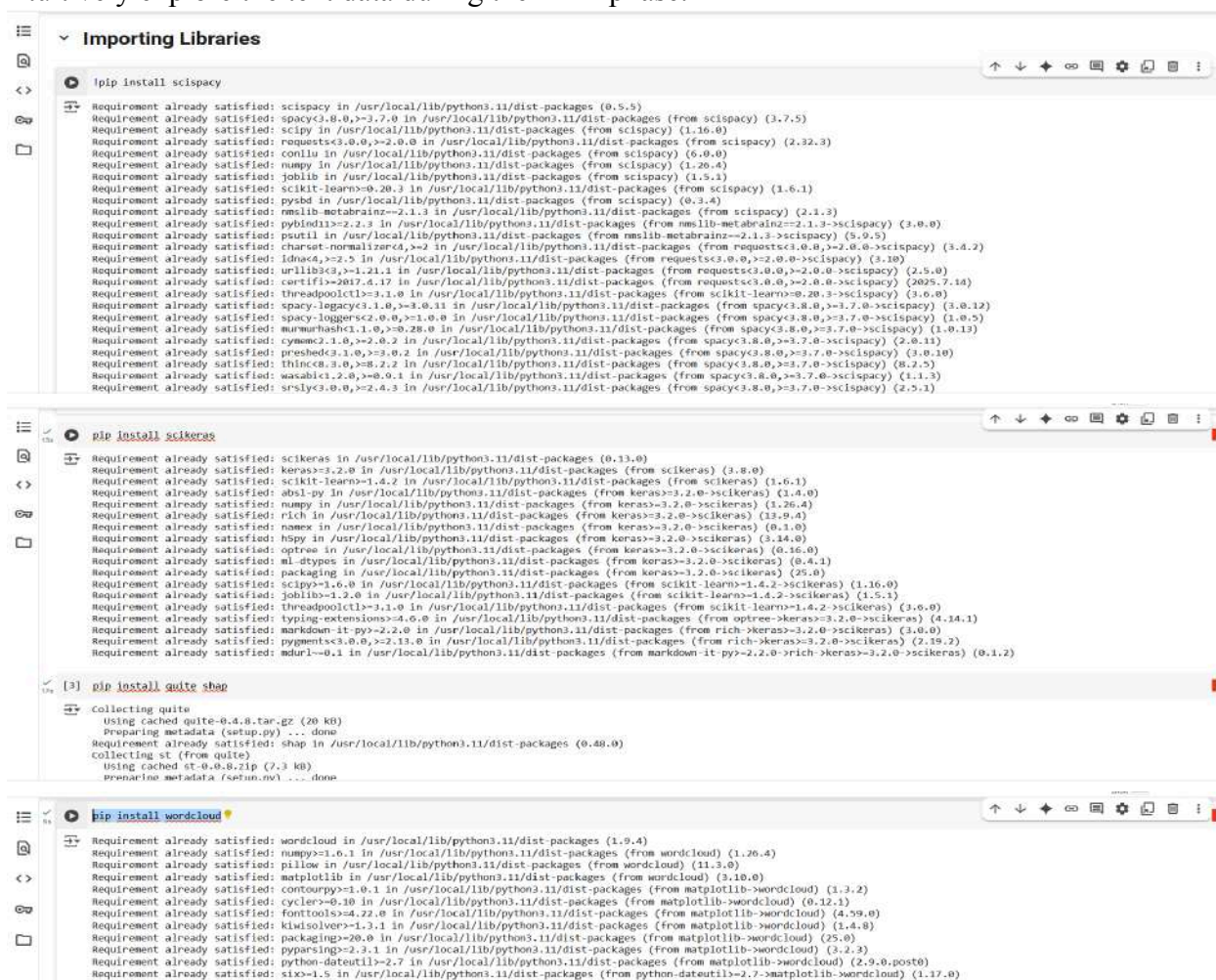
17. Siegle D. ANOVA, Regression, and Chi-Square | Educational Research Basics by Del Siegle. Uconn.edu. Published June 14, 2015. Accessed June 15, 2025.
https://researchbasics.education.uconn.edu/anova_regression_and_chi-square/
18. GeeksforGeeks. Label Encoding in Python. GeeksforGeeks. Published October 15, 2018. Accessed June 15, 2025.
<https://www.geeksforgeeks.org/machine-learning/ml-label-encoding-of-datasets-in-python/>
19. Singh B, Desai R, Ashar H, Tank P, Neha Katre. A Trade-off between ML and DL Techniques in Natural Language Processing. *Journal of physics*. 2021;1831(1):012025-012025. Doi: <https://doi.org/10.1088/1742-6596/1831/1/012025>
20. Geeksforgeeks. Understanding TF-IDF (Term Frequency-Inverse Document Frequency). GeeksforGeeks. Published January 20, 2021. Accessed June 15, 2025.
<https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/>
21. GeeksforGeeks. ZScore Normalization: Definition and Examples. GeeksforGeeks. Published July 26, 2024.
<https://www.geeksforgeeks.org/data-analysis/z-score-normalization-definition-and-examples/>
22. Transforming Skewed Data for Statistical Analysis: log, square root, inverse. Anatomise Biostats. Published June 28, 2017.
<https://anatomisebiostats.com/biostatistics-blog/transforming-skewed-data/>
23. Mohammed AJ. Improving Classification Performance for a Novel Imbalanced Medical Dataset using SMOTE Method. *International Journal of Advanced Trends in Computer Science and Engineering*. 2020;9(3):3161-3172. Doi: <https://doi.org/10.30534/ijatcse/2020/104932020>
24. GeeksforGeeks. SMOTE for Imbalanced Classification with Python. GeeksforGeeks. Published May 3, 2024.
<https://www.geeksforgeeks.org/machine-learning/smote-for-imbalanced-classification-with-python/>
25. scikit-learn. LogisticRegression. Scikit-learn.org. Published 2014.
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
26. One-class SVM with non-linear kernel (RBF). scikit-learn. Published 2024.
https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html
27. sklearn.neighbors.KNeighborsClassifier - scikit-learn 0.23.1 documentation. scikit-learn.org.
<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier>
28. scikit-learn. 1.10. Decision Trees - scikit-learn 0.22 documentation. Scikit-learn.org. Published 2025. <https://scikit-learn.org/stable/modules/tree.html>
29. Scikit-Learn. sklearn.ensemble.RandomForestClassifier - scikit-learn 0.20.3 Documentation. Scikit-learn.org. Published 2025.
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

30. Scikit-learn. 3.2.4.3.5. sklearn.ensemble.GradientBoostingClassifier - scikit-learn 0.20.3 documentation. Scikit-learn.org. Published 2009. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
31. XGBoost4J-Spark-GPU Tutorial - xgboost 3.1.0-dev documentation. Readthedocs.io. Published 2025. Accessed July 30, 2025. https://xgboost.readthedocs.io/en/latest/jvm/xgboost4j_spark_gpu_tutorial.html
32. GeeksforGeeks. Feedforward Neural Network. GeeksforGeeks. Published June 20, 2024. <https://www.geeksforgeeks.org/nlp/feedforward-neural-network/>
33. Vlad Andreichuk. Non-negative Matrix Factorization (NMF) for the Grouping of Articles' Titles. Medium. Published December 11, 2023. Accessed July 30, 2025. <https://medium.com/@vlad.andreichuk/non-negative-matrix-factorization-nmf-for-the-grouping-of-articles-titles-a73e654b6244>
34. Abid Ali Awan. An Introduction to SHAP Values and Machine Learning Interpretability. Datacamp.com. Published June 28, 2023. <https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability>
35. GeeksforGeeks. AUC ROC Curve in Machine Learning. GeeksforGeeks. Published November 25, 2020. <https://www.geeksforgeeks.org/machine-learning/auc-roc-curve/>

Appendix:

We have chosen to work in the GoogleColab environment because of its notebook shareability access to everyone in the team.

- 1. Installing necessary Libraries and Packages:** To begin our work, we installed the necessary packages to support various stages of data preprocessing, modeling, and interpretation. The 3 images below illustrate the installation of key libraries including ScispaCy, SciKeras, SHAP, and WordCloud. ScispaCy is a specialized NLP toolkit built on spaCy, tailored for processing biomedical and clinical text, making it valuable for future enhancements involving drug-related text analysis. SciKeras provides a seamless integration between Keras (TensorFlow-based deep learning models) and Scikit-learn, allowing us to treat neural networks like any other Scikit-learn estimator, ideal for hyperparameter tuning and cross-validation. SHAP (SHapley Additive exPlanations) helps in interpreting complex model predictions by assigning importance values to input features, which is essential for transparency in health-related models. WordCloud enables visual representation of the most frequent words in side effect descriptions, helping us intuitively explore the text data during the EDA phase.



```
Importing Libraries

!pip install scispacy

Requirement already satisfied: scispacy in /usr/local/lib/python3.11/dist-packages (0.5.5)
Requirement already satisfied: spacy<3.0.0,>=2.7.0 in /usr/local/lib/python3.11/dist-packages (from scispacy) (3.7.5)
Requirement already satisfied: scip in /usr/local/lib/python3.11/dist-packages (from scispacy) (1.16.0)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from scispacy) (2.32.3)
Requirement already satisfied: conllu in /usr/local/lib/python3.11/dist-packages (from scispacy) (6.0.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from scispacy) (1.26.4)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from scispacy) (1.5.1)
Requirement already satisfied: scikit-learn<=0.20.3 in /usr/local/lib/python3.11/dist-packages (from scispacy) (1.6.1)
Requirement already satisfied: pytsdb in /usr/local/lib/python3.11/dist-packages (from scispacy) (0.3.4)
Requirement already satisfied: nmslib-metabrainz==2.1.3 in /usr/local/lib/python3.11/dist-packages (from scispacy) (2.1.3)
Requirement already satisfied: pybind11>=2.3 in /usr/local/lib/python3.11/dist-packages (from nmslib-metabrainz==2.1.3->scispacy) (3.0.0)
Requirement already satisfied: putil in /usr/local/lib/python3.11/dist-packages (from nmslib-metabrainz==2.1.3->scispacy) (5.9.5)
Requirement already satisfied: charcat-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->scispacy) (3.4.2)
Requirement already satisfied: idna<4,>=2.2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->scispacy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->scispacy) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->scispacy) (2025.7.14)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn<=0.20.3->scispacy) (3.6.0)
Requirement already satisfied: scipy-logger<1.0,>=0.8.11 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (1.0.13)
Requirement already satisfied: cycore<1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (2.0.11)
Requirement already satisfied: thinc<8.0.0,>=8.2.2 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (8.2.5)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (1.1.3)
Requirement already satisfied: srscyc<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy<3.0.0,>=3.7.0->scispacy) (2.5.1)

!pip install scikeras

Requirement already satisfied: scikeras in /usr/local/lib/python3.11/dist-packages (0.13.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from scikeras) (3.8.0)
Requirement already satisfied: scikit-learn>=1.4.2 in /usr/local/lib/python3.11/dist-packages (from scikeras) (1.6.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (13.9.4)
Requirement already satisfied: nameex in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.1.0)
Requirement already satisfied: hspv in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (3.14.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.16.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras>=3.2.0->scikeras) (25.0)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4.2->scikeras) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4.2->scikeras) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.4.2->scikeras) (3.6.0)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from optree>=0.9.0->scikeras) (4.14.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>keras>=3.2.0->scikeras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>keras>=3.2.0->scikeras) (2.19.2)
Requirement already satisfied: mdurl<=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>keras>=3.2.0->scikeras) (0.1.2)

!pip install quite shap

collecting quite
Using cached quite-0.4.8.tar.gz (26 kB)
Preparing metadata (setup.py) ... done
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.48.0)
collecting st (from quite)
Using cached st-0.8.2.zip (7.3 kB)
Preparing metadata (setup.py) ... done

!pip install wordcloud

Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)
Requirement already satisfied: numpy<1.11 in /usr/local/lib/python3.11/dist-packages (from wordcloud) (1.26.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from wordcloud) (113.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.17.0)
```

To build a robust and interpretable pregnancy drug safety classification model using machine learning (ML) and deep learning (DL), we utilized a wide range of Python libraries. Each import was carefully selected based on the tasks we performed during the

data cleaning, preprocessing, modeling, evaluation, and visualization phases. Below is a description of each group of packages and their relevance:

- pandas, numpy: Core libraries for handling tabular data structures, performing numerical computations, and manipulating DataFrames efficiently throughout the project.
- matplotlib.pyplot, seaborn: Used for creating static and informative visualizations such as histograms, bar charts, confusion matrices, and heatmaps to support Exploratory Data Analysis (EDA) and model evaluation.
- matplotlib.patches.Patch, matplotlib.lines.Line2D: Used for custom legends and styling in plots.
- scipy.stats, chi2_contingency, f_oneway: Employed for hypothesis testing (e.g., chi-squared and ANOVA) to understand associations between features and target classes.
- re, string: Regular expressions and string manipulations for cleaning raw text in the 'Side_Effects' column.
- nltk, nltk.corpus.stopwords, nltk.word_tokenize, pos_tag, WordNetLemmatizer, wordnet: Used for natural language preprocessing, including tokenization, POS tagging, stopword removal, and lemmatization, ensuring meaningful feature extraction from free-text side effects.
- collections.Counter: Helpful for frequency analysis of tokens and feature inspection.
- sklearn.feature_extraction.text.CountVectorizer, TfidfTransformer: Transformed cleaned side effect text into numerical vectors using TF-IDF for ML modeling.
- sklearn.compose.ColumnTransformer, OneHotEncoder: Encoded categorical features for modeling.
- StandardScaler: Standardized numerical features to improve model convergence and performance.
- scipy.sparse.hstack, issparse: Combined sparse text features and encoded features efficiently.
- sklearn.linear_model.LogisticRegression, sklearn.svm.SVC, sklearn.neighbors.KNeighborsClassifier, sklearn.tree.DecisionTreeClassifier, sklearn.ensemble.RandomForestClassifier, GradientBoostingClassifier: Implemented baseline and advanced ML classifiers.
- xgboost.XGBClassifier: Deployed an optimized gradient boosting model for performance comparison.
- sklearn.pipeline.Pipeline, GridSearchCV, StratifiedKFold, RandomizedSearchCV: Used for hyperparameter tuning, cross-validation, and creating model pipelines.
- sklearn.utils.class_weight: Addressed class imbalance in training by assigning appropriate weights.
- tensorflow, Sequential, Dense, Dropout, Adam, EarlyStopping: Built, compiled, and trained a neural network model with dropout and early stopping to avoid overfitting.
- scikeras.wrappers.KerasClassifier: Integrated Keras models within Scikit-learn workflows for tuning and evaluation.
- sklearn.metrics: Calculated key metrics like accuracy, precision, recall, F1-score, AUC-ROC, and confusion matrix to evaluate model performance.

- roc_curve, auc, label_binarize, cycle: Enabled plotting of multi-class ROC curves.
- sklearn.exceptions.NotFittedError: Helped handle model validation errors gracefully.
- imblearn.over_sampling.SMOTE: Used Synthetic Minority Oversampling Technique to balance the pregnancy drug categories for better learning by the model.
- joblib, cloudpickle: Saved and loaded models and vectorizers efficiently during deployment.
- random: Used for reproducibility and sampling consistency across training stages.
- wordcloud.WordCloud: Created visual summaries of frequent terms in side effect descriptions.
- sklearn.decomposition.NMF: Used Non-negative Matrix Factorization for topic modeling as an exploratory insight.

The 3 below pictures depicts importing the necessary libraries for the entire project:

The image displays three sequential screenshots of a Jupyter Notebook, showing the import of various Python libraries for a machine learning project. The first screenshot shows the initial imports, including pandas, numpy, matplotlib, seaborn, scipy, sklearn, and nltk. The second screenshot shows the imports for sklearn, tensorflow, keras, and other machine learning related libraries. The third screenshot shows the imports for cloudpickle, wordcloud, and other libraries, along with the execution of the imports, showing the progress of downloading and unzipping the nltk data packages.

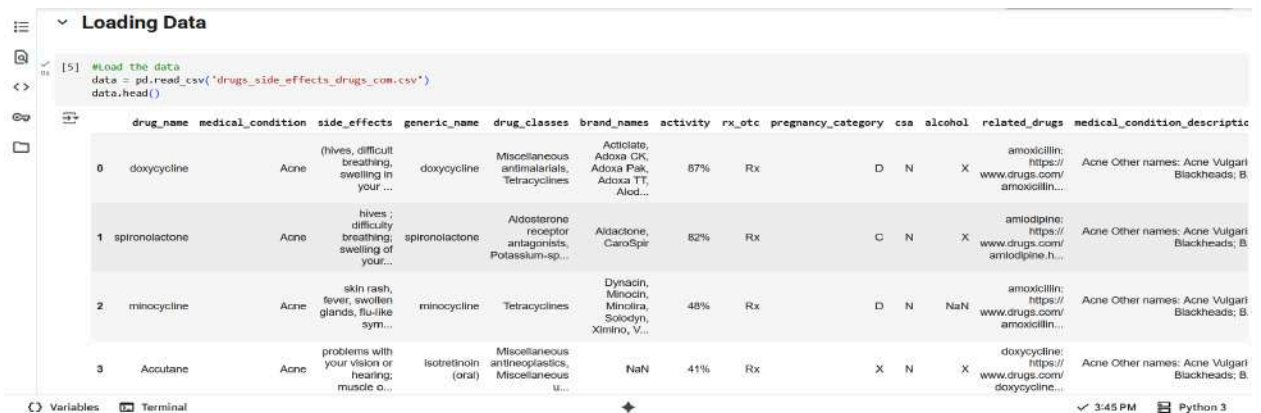
```
# Import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import chi2_contingency
from scipy.stats import f_oneway
from sklearn.preprocessing import LabelEncoder
import re
import nltk
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import wordnet
from nltk import pos_tag, word_tokenize
from nltk.stem import WordNetLemmatizer
import string
from nltk.corpus import stopwords
from collections import Counter
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from scipy.sparse import hstack

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV, StratifiedKFold
import tensorflow as tf
from sklearn.utils import class_weight
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
from tensorflow.keras.optimizers import Adam
from sklearn.base import BaseEstimator, ClassifierMixin
from sklearn.preprocessing import Label_Binarize
from sklearn.exceptions import NotFittedError
from itertools import cycle
from sklearn.wrappers import KerasClassifier
from sklearn.model_selection import RandomizedSearchCV
import shap
import joblib

import cloudpickle as cp
from wordcloud import WordCloud
from sklearn.decomposition import NMF
import random
from sklearn.metrics import (precision_recall_fscore_support)
from scipy.sparse import issparse
from imblearn.over_sampling import SMOTE
from matplotlib.patches import Patch
from matplotlib.lines import Line2D

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data] /root/nltk_data...
[nltk_data] Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

2. **Loading the Data:** The first step in our project involved loading the dataset, which contained structured information on prescription drugs, their side effects, usage conditions, and corresponding pregnancy safety categories. We used pandas to efficiently read the dataset into a DataFrame, enabling easy inspection, manipulation, and preprocessing. This foundational step allowed us to verify the shape of the data, check for missing values, and begin initial exploratory analysis. Loading the data properly ensured that all subsequent transformations and modeling efforts were built on a reliable and consistent base.



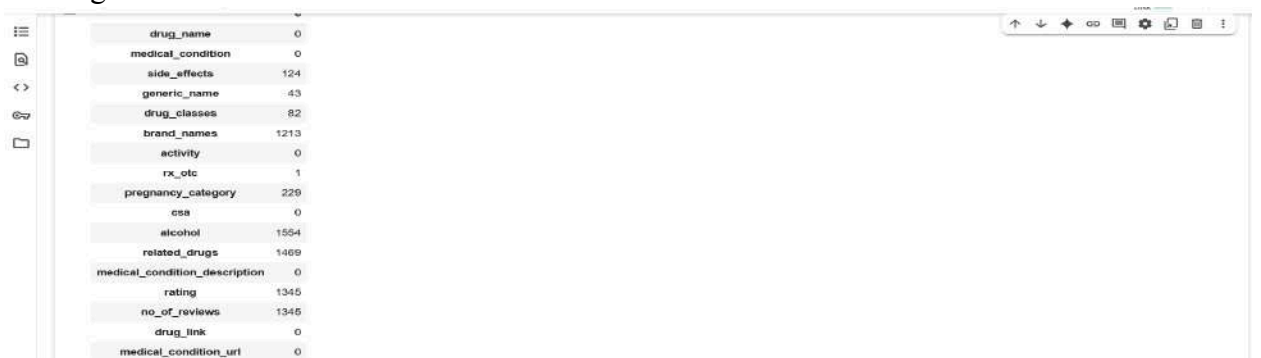
| | drug_name | medical_condition | side_effects | generic_name | drug_classes | brand_names | activity | rx_etc | pregnancy_category | csa | alcohol | related_drugs | medical_condition_descriptio |
|---|----------------|-------------------|---|---------------------|---|--|----------|--------|--------------------|-----|---------|---------------|---|
| 0 | doxycycline | Acne | (hives, difficult breathing, swelling in your... | doxycycline | Miscellaneous antimicrobials, Tetracyclines | Actidione, Adoxa CR, Adoxa Pak, Adoxa TT, Alod... | 57% | Rx | | D | N | X | amoxicillin: https://www.drugs.com/amoxicillin... |
| 1 | spironolactone | Acne | hives; difficulty breathing; swelling of your... | spironolactone | Aldosterone receptor antagonists, Potassium-sp... | Aldactone, CaroSpir | 52% | Rx | | C | N | X | amlodipine: https://www.drugs.com/amlodipine.h... |
| 2 | minocycline | Acne | skin rash, fever, swollen glands, flu-like sym... | minocycline | Tetracyclines | Dynacin, Minocin, Minocitra, Solodyn, Ximino, V... | 48% | Rx | | D | N | NaN | amoxicillin: https://www.drugs.com/amoxicillin... |
| 3 | Accutane | Acne | problems with your vision or hearing; muscle o... | isotretinoin (oral) | Miscellaneous antineoplastics, Miscellaneous U... | NaN | 41% | Rx | | X | N | X | doxycycline: https://www.drugs.com/doxycycline... |

3. **Data Cleaning:** To begin the data cleaning process, we first examined the shape of the dataset using `data.shape`, which revealed that our dataset consists of 2,931 rows and 17 columns. This step helps us understand the scale of the data and sets expectations for further processing.



| Column | Count |
|-------------------------------|-------|
| drug_name | 0 |
| medical_condition | 0 |
| side_effects | 124 |
| generic_name | 43 |
| drug_classes | 82 |
| brand_names | 1213 |
| activity | 0 |
| rx_etc | 1 |
| pregnancy_category | 229 |
| csa | 0 |
| alcohol | 1554 |
| related_drugs | 1469 |
| medical_condition_description | 0 |
| rating | 1345 |
| no_of_reviews | 1345 |
| drug_link | 0 |
| medical_condition_url | 0 |

Next, we used `data.isna().sum()` to identify the number of missing values in each column. This step is crucial because missing data can negatively affect model training and performance. The output showed that several columns, like `side_effects`, `generic_name`, `brand_names`, `pregnancy_category`, `alcohol`, `related_drugs`, `rating`, and `no_of_reviews`, contain significant numbers of null entries. Knowing which columns have missing values and how many helps guide our decisions on whether to impute, drop, or further investigate these columns. This understanding ensures we handle missing data thoughtfully, preserving as much useful information as possible for accurate model building.



| Column | Count |
|-------------------------------|-------|
| drug_name | 0 |
| medical_condition | 0 |
| side_effects | 124 |
| generic_name | 43 |
| drug_classes | 82 |
| brand_names | 1213 |
| activity | 0 |
| rx_etc | 1 |
| pregnancy_category | 229 |
| csa | 0 |
| alcohol | 1554 |
| related_drugs | 1469 |
| medical_condition_description | 0 |
| rating | 1345 |
| no_of_reviews | 1345 |
| drug_link | 0 |
| medical_condition_url | 0 |

Next, we executed `data.info()` to gain an overview of the dataset structure, including the data types and the number of non-null values for each column. This step is critical for identifying the nature of each feature (e.g: numeric vs. categorical vs. text) and spotting potential data type mismatches or issues. It also reconfirms missing values identified earlier and helps us plan necessary type conversions or imputations.

Then, we used `data.describe(include="all")` to generate descriptive statistics for all columns. This command provides insights into data distribution, including counts, unique values, and most frequent categories for object-type columns, and mean, standard deviation, and range for numerical fields. For instance, the `rating` column had an average of 6.81 with a max of 10, which gives a sense of how patients rated drugs. The `no_of_reviews` column shows a wide range, from 1 to over 2900 reviews, indicating varying sample sizes per drug. These statistics are foundational for understanding the variability and central tendencies in the data, helping us decide on feature selection, encoding strategies, and normalization requirements later in the pipeline.

[illegible]

To improve the dataset's quality and make it ready for modeling, we continued refining it through several targeted cleaning steps. First, we handled the alcohol column by creating a new feature called alcohol_interaction, replacing missing values with 'No' and interpreting 'X' entries as 'Yes' to capture the presence of alcohol interaction in a drug. Then, since pregnancy_category is our target variable and side_effects is a crucial feature, we dropped rows with missing values in those columns to ensure model reliability. Next, we cleaned the activity column by removing the '%' sign and converting it to integers, making it suitable for numerical analysis. For categorical columns like brand_names, generic_name, and drug_classes, we filled missing entries with 'Unknown' using a custom function, ensuring completeness without introducing bias. Missing values in rating were marked as -1 to denote unavailability, while no_of_reviews was set to 0 when absent, assuming no feedback was provided. This column was also cast to an integer type for consistency.

```
[11] #Fill nan with 'no' and convert 'x' to 'Yes'
data['alcohol_interaction'] = data['alcohol'].fillna('No')
data['alcohol_interaction'] = data['alcohol_interaction'].apply(lambda x: 'Yes' if x == 'X' else 'no')

[12] data = data.dropna(subset=['pregnancy_category'])#Dropping the rows with missing target value
data = data.dropna(subset=['side_effects'])#Dropping the missing values of side effects

[13] #removing the '%' symbol, then convert the resulting values to integers
data['activity'] = data['activity'].str.replace('%', '').astype(int)

[14] #Function to fill missing values
def fill_missing_values(df, columns, fill_value="unknown"):
    for col in columns:
        df[col] = df[col].fillna(fill_value)
    return df

[15] #filling the missing values with 'unknown'
columns_to_fill = ['brand_names', 'generic_name', 'drug_classes']
data = fill_missing_values(data, columns_to_fill, fill_value="unknown")

[16] data['rating'] = data['rating'].fillna(-1) #Fill missing values in rating with -1 to indicate missing value
data['no_of_reviews'] = data['no_of_reviews'].fillna(0) #Fill missing values in no_of_reviews with 0 assuming no reviews were given

[17] #Changing the datatype
data['no_of_reviews'] = data['no_of_reviews'].astype(int)
```

To standardize text formatting, we applied title case to generic_name and medical_condition, and capitalized the first letter of drug_name entries that began with a lowercase letter. Finally, we removed several irrelevant columns (medical_condition_description, drug_link, medical_condition_url, alcohol, and related_drugs) that were either redundant or had too many missing values. These refinements ensured a cleaner, more consistent, and analysis-ready dataset moving into the modeling phase. To ensure consistency in categorical values, we manually corrected inconsistent capitalization in the Medical_Condition column, for example, changing entries like "Copd" to "COPD" and "Covid 19" to "COVID 19". This step ensures better grouping during analysis and prevents the same condition from being treated as separate categories due to minor textual differences. We then standardized all column names by replacing underscores with spaces and applying title case formatting. This improved readability and brought uniformity to our dataset structure.

```
#Convert the words to title format
title_columns = ['generic_name', 'medical_condition']
#Apply title case to each of the specified columns
for col in title_columns:
    data[col] = data[col].str.title()

[19] #Capitalize the first letter of drug_name only if it starts with a lowercase letter
def capitalize_first_letter(name):
    if isinstance(name, str) and name and name[0].islower():
        return name[0].upper() + name[1:]
    return name

[20] data['drug_name'] = data['drug_name'].apply(capitalize_first_letter)

[21] #Drop the unnecessary columns
data = data.drop(columns=['medical_condition_description', 'drug_link', 'medical_condition_url', 'alcohol', 'related_drugs'])

[22] #Correct inconsistent capitalization and formatting in the medical_condition column
corrections = {'copd': 'COPD', 'ibd (bowel)': 'IBD (Bowel)', 'covid 19': 'COVID 19', 'adhd': 'ADHD', 'Aids/Hiv': 'AIDS/HIV',
               'Gerd (Heartburn)': 'GERD (Heartburn)', 'UTI': 'UTI', 'Alzheimer's': 'Alzheimer's'}
data['medical_condition'] = data['medical_condition'].replace(corrections)

[23] #Standardizing Column names
data.columns = [col.replace('_', ' ').title().replace(' ', '_') for col in data.columns]
```

The impact of the above-mentioned changes is evident when comparing the updated `describe()` output with the previous version, column names are now consistently formatted, and data entries appear cleaner.

```
data.describe(include = "all")
```

| | Drug_Name | Medical_Condition | Side_Effects | Generic_Name | Drug_Classes | Brand_Names | Activity | Rx_Otc | Pregnancy_Category | Csa | Rating | No_Of_Reviews | Alcohol_Interaction |
|--------|---------------|-------------------|--|--------------|--------------------------------|-------------|-------------|--------|--------------------|------|-------------|---------------|---------------------|
| count | 2591 | 2591 | 2591 | 2591 | 2591 | 2591 | 2591.000000 | 2591 | 2591 | 2591 | 2591.000000 | 2591.000000 | 2591 |
| unique | 2572 | 47 | 2543 | 1190 | 264 | 1457 | NaN | 3 | 6 | 6 | NaN | NaN | 25 |
| top | Triamcinolone | Colds & Flu | hives; difficult breathing; swelling of your ... | Unknown | Upper respiratory combinations | Unknown | NaN | Rx | C | N | NaN | NaN | |
| freq | 3 | 233 | 10 | 22 | 235 | 969 | NaN | 1720 | 1318 | 2384 | NaN | NaN | 13 |
| mean | NaN | NaN | NaN | NaN | NaN | NaN | 8.624469 | NaN | NaN | NaN | 3.352258 | 43.026631 | N |
| std | NaN | NaN | NaN | NaN | NaN | NaN | 17.380298 | NaN | NaN | NaN | 4.260288 | 147.485916 | N |
| min | NaN | NaN | NaN | NaN | NaN | NaN | 0.000000 | NaN | NaN | NaN | -1.000000 | 0.000000 | N |
| 25% | NaN | NaN | NaN | NaN | NaN | NaN | 0.000000 | NaN | NaN | NaN | -1.000000 | 0.000000 | N |
| 50% | NaN | NaN | NaN | NaN | NaN | NaN | 2.000000 | NaN | NaN | NaN | 4.000000 | 1.000000 | N |
| 75% | NaN | NaN | NaN | NaN | NaN | NaN | 7.000000 | NaN | NaN | NaN | 7.400000 | 17.000000 | N |
| max | NaN | NaN | NaN | NaN | NaN | NaN | 100.000000 | NaN | NaN | NaN | 10.000000 | 2834.000000 | N |

Additionally, we reconfirmed the structure and completeness of our final dataset using `data.info()` and `data.isna().sum()`.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2591 entries, 0 to 2590
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Drug_Name             2591 non-null   object
 1   Medical_Condition     2591 non-null   object
 2   Side_Effects          2591 non-null   object
 3   Generic_Name          2591 non-null   object
 4   Drug_Classes          2591 non-null   object
 5   Brand_Names           2591 non-null   object
 6   Activity               2591 non-null   int64
 7   Rx_Otc                 2591 non-null   object
 8   Pregnancy_Category    2591 non-null   object
 9   Csa                    2591 non-null   object
10   Rating                 2591 non-null   float64
11   No_Of_Reviews         2591 non-null   int64
12   Alcohol_Interaction    2591 non-null   object
dtypes: float64(1), int64(2), object(10)
memory usage: 283.4+ KB
```

All 2,591 records across 13 refined columns are now free of missing values. This confirms that our data is fully cleaned and ready for the next steps in modeling, ensuring a robust and error-free analysis pipeline.

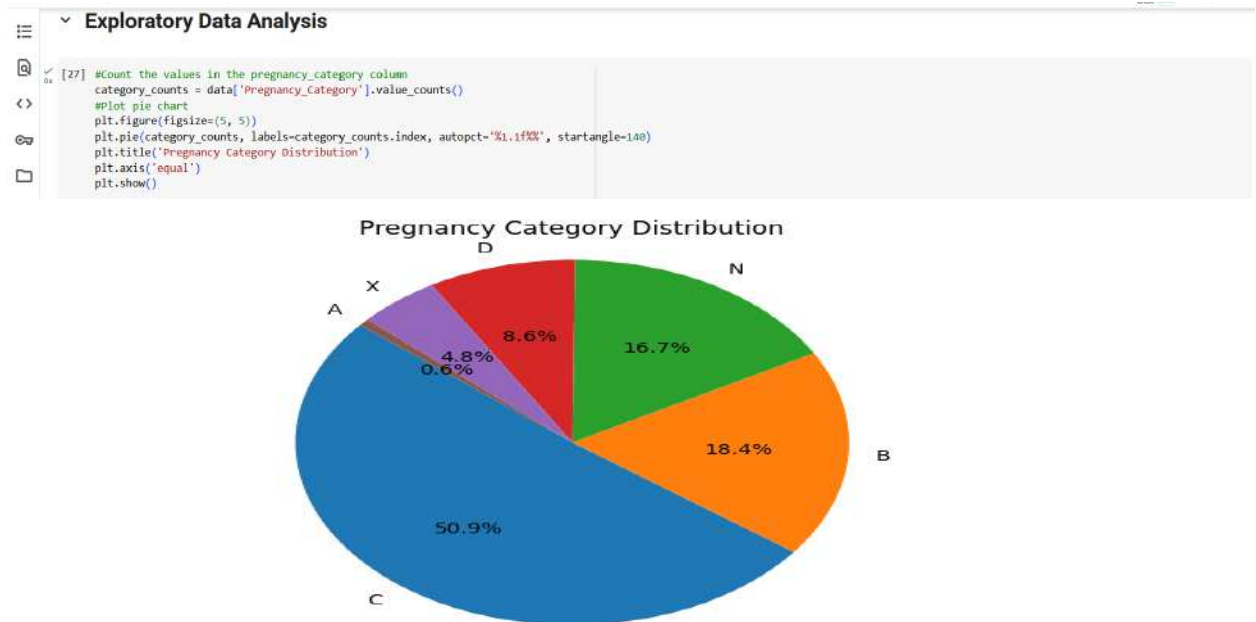
```
[26] #confirming there are no missing/na values
data.isna().sum()
```

| | |
|---------------------|---|
| Drug_Name | 0 |
| Medical_Condition | 0 |
| Side_Effects | 0 |
| Generic_Name | 0 |
| Drug_Classes | 0 |
| Brand_Names | 0 |
| Activity | 0 |
| Rx_Otc | 0 |
| Pregnancy_Category | 0 |
| Csa | 0 |
| Rating | 0 |
| No_Of_Reviews | 0 |
| Alcohol_Interaction | 0 |

dtype: int64

- Exploratory Data Analysis:** To better understand the structure and relationships within the dataset, a series of visualizations were generated during the EDA phase. These plots helped uncover trends, and distributions, guiding important decisions for preprocessing and modeling.

- Pregnancy Category Distribution Pie Chart:** This pie chart visualized the proportion of drugs across different pregnancy safety categories. It helped assess class imbalance, which is critical when selecting appropriate classification strategies.

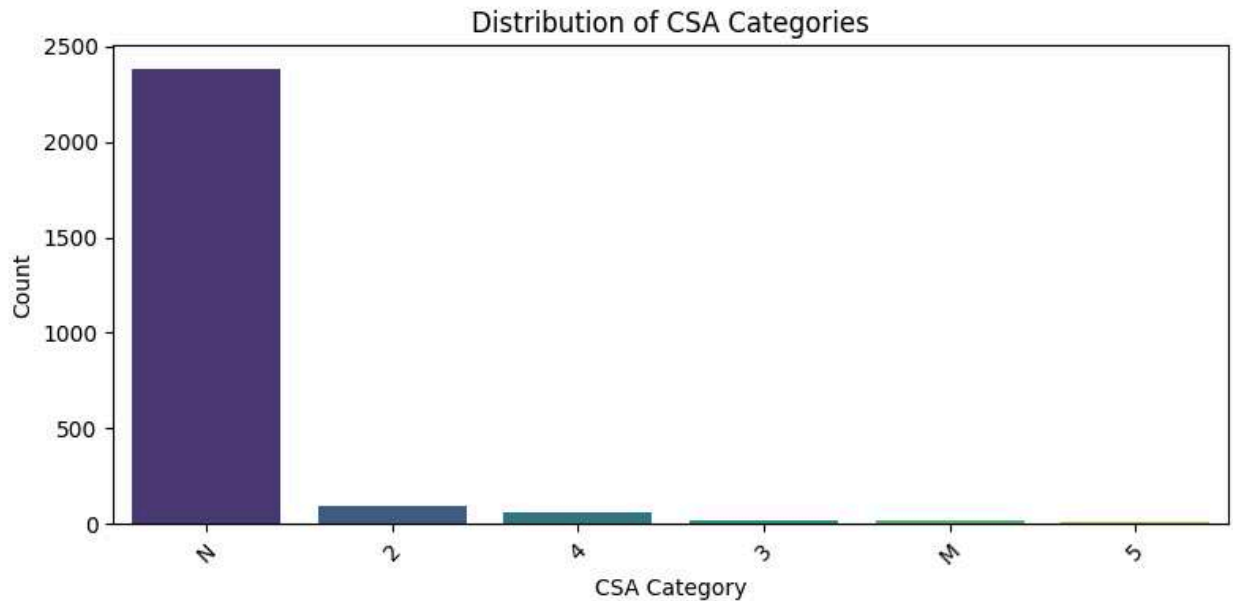


Interpretation: From the Pie chart, it is clear that the majority of drugs in the data fall under Category C (approximately 51%), which means animal studies have shown adverse effects on the fetus, but there are no adequate human studies. This suggests a significant portion of medications present potential risks during pregnancy but may still be prescribed when the benefits outweigh the risks.

Categories B and N follow next with around 18% and 17% respectively. Category B indicates no evidence of risk in humans, while Category N typically refers to drugs not yet classified by the FDA. Categories D and X (about 9% and 5%) indicate increasing levels of risk, with Category X drugs being contraindicated in pregnancy. Category A, the safest category, is underrepresented (less than 1%), which is expected as very few drugs are deemed completely risk-free in pregnancy.

- CSA Category Bar Plot:** A bar plot of Controlled Substance Act (CSA) categories highlighted the frequency of scheduled versus non-scheduled drugs. This visualization emphasized the prevalence of non-controlled drugs in the dataset.



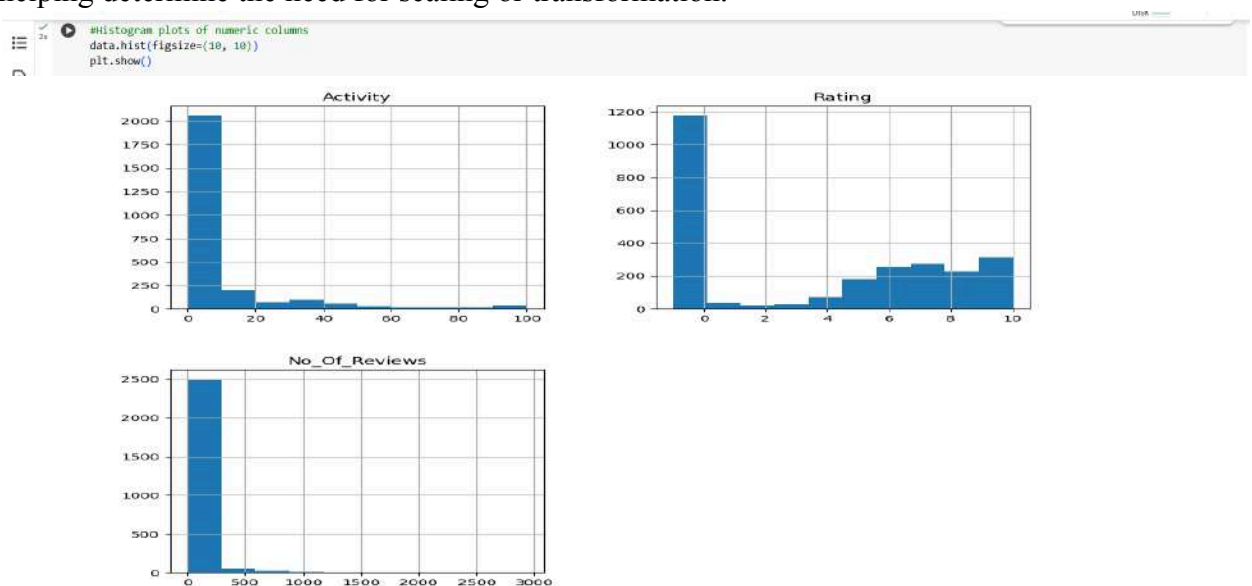


Interpretation: This barplot highlights a strong dominance of drugs classified under CSA Category 'N', which means they are not controlled substances under the Controlled Substances Act. Over 90% of the drugs in the data fall into this category. This is expected, as most medications are not considered to have abuse potential. The remaining categories (2, 3, 4, 5, and M) represent controlled substances with varying levels of restriction, where:

- Category 2 includes drugs with a high potential for abuse (e.g: opioids),
- Category 3-5 represent decreasing abuse potential,
- Category M likely denotes medically prescribed marijuana or similar agents.

These controlled categories are significantly underrepresented, with each comprising only a small fraction of the data.

- **Histograms of Numeric Features:** Histograms for Activity, Rating, and No_Of_Reviews provided insights into skewness and distribution of numerical features, helping determine the need for scaling or transformation.

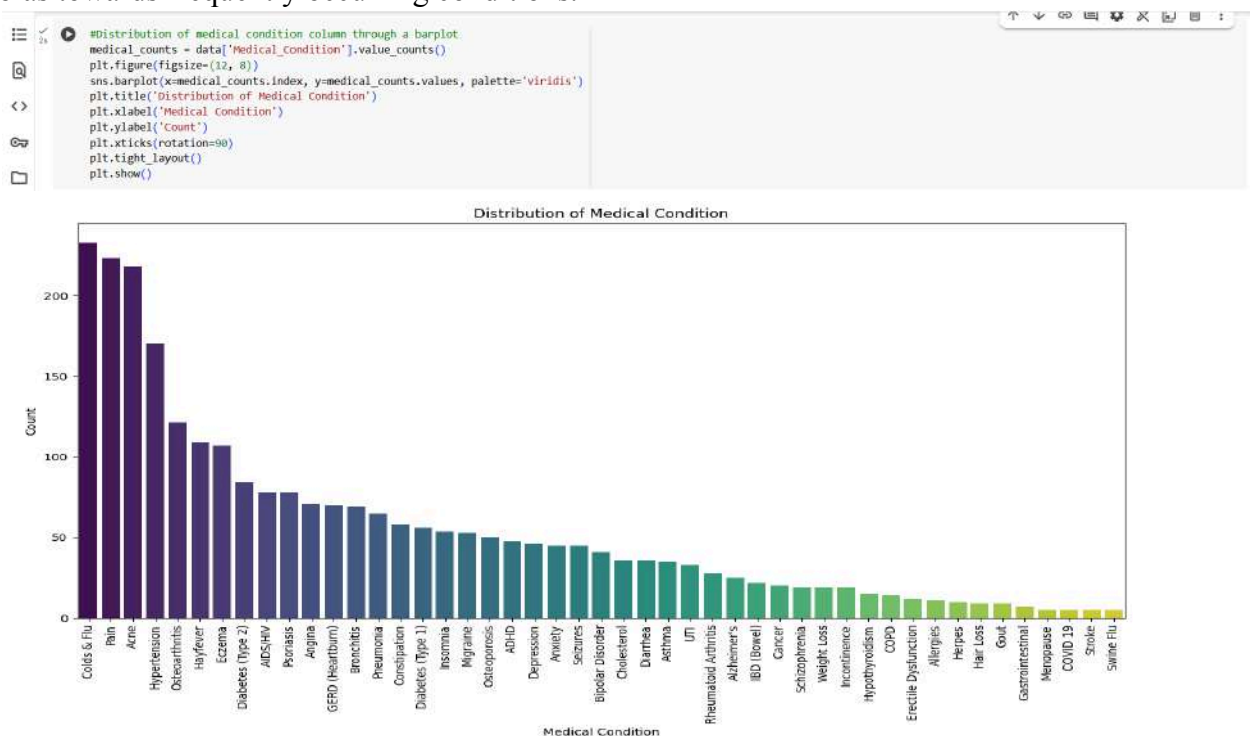


Interpretation: The histograms illustrate the distributions of three key continuous variables in the data: drug activity, user rating, and number of reviews.

- The Activity histogram is heavily right-skewed, showing that the majority of drugs have very low activity levels, with only a small number reaching high usage rates.
- Similarly, the No_Of_Reviews histogram reveals a sharp skew toward fewer reviews, with most drugs having under 200 reviews and only a handful exceeding 1,000. This indicates that drug visibility and user engagement are concentrated in a small subset of the dataset.
- In contrast, the Rating histogram follows a bimodal pattern: a large spike at zero likely reflects missing or placeholder values, while the remaining data shows a gradual increase toward the higher end of the scale, peaking near 10.

These distributions highlight the need for data transformation or normalization before feeding into machine learning models and also suggest that review-based or activity-based metrics may introduce bias if not properly accounted for. Understanding these imbalances is crucial for developing a fair and accurate pregnancy drug safety classifier.

- **Medical Condition Distribution Bar Plot:** A bar plot of Medical_Condition counts revealed which conditions had the most drug entries. This informed the model's potential bias towards frequently occurring conditions.

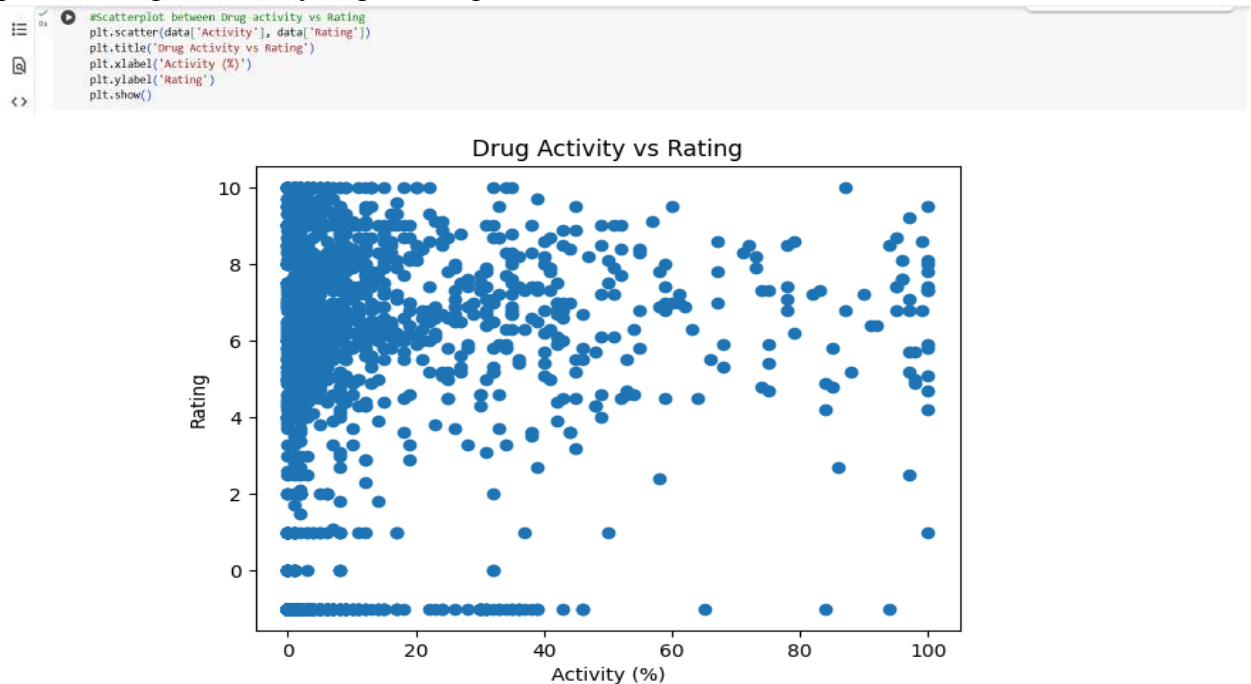


Interpretation: The bar chart illustrates the frequency of different medical conditions associated with the drugs in the data. The top three conditions Colds & Flu, Pain, and Acne are the most common, followed by chronic conditions like Hypertension, Osteoarthritis, and Diabetes (Type 2). These account for a large portion of the medications and reflect a typical pattern in community prescribing.

On the other end of the plot, rare or specific conditions like Stroke, Swine Flu, and Menopause appear much less frequently. This suggests that while the data is broad in scope, it's heavily

skewed toward commonly occurring conditions, which could affect how well the model generalizes to less common cases.

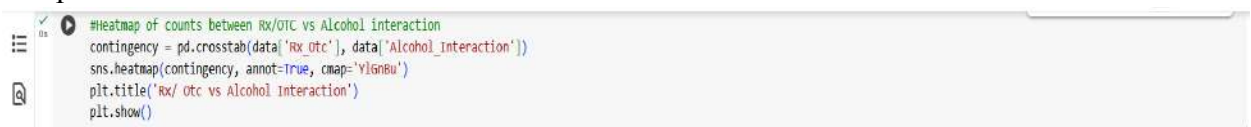
- **Drug Activity vs. Rating Scatter Plot:** This scatter plot showed the relationship between drug activity and user rating. It was helpful in evaluating whether pharmacological activity aligns with patient satisfaction.



Interpretation: The scatter plot of Drug Activity versus Rating reveals key insights into prescribing trends and user-perceived effectiveness across various drugs. A significant concentration of drugs lies within the low activity range (0–20%), yet many of these receive high user ratings (above 7), suggesting that several less frequently prescribed drugs may still be considered effective by patients. Conversely, drugs with high activity percentages (above 50%) exhibit a broad range of ratings, indicating that frequent use does not necessarily imply higher patient satisfaction. Moreover, low-rated drugs (rating < 3) are dispersed across all levels of activity, highlighting that usage frequency is not a reliable indicator of perceived safety or efficacy.

These findings reinforce the need for a machine learning approach that classifies pregnancy risk using a broader set of structured features, including pregnancy category, side effects, and drug class, rather than relying solely on conventional indicators like drug popularity or user ratings. This supports the study's objective of developing a robust predictive model to improve medication safety during pregnancy.

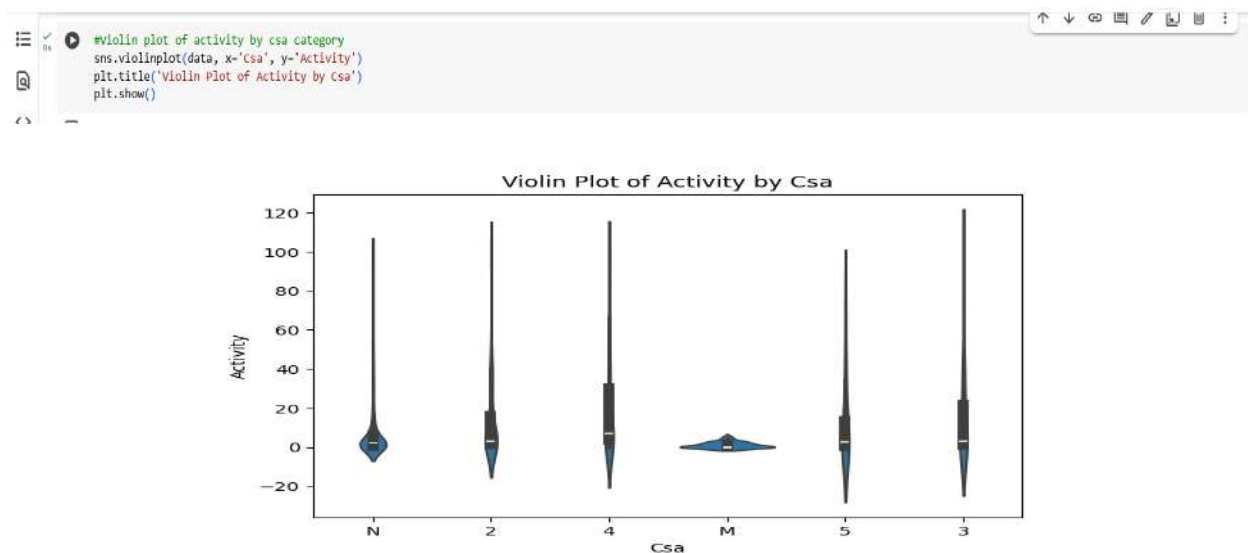
- **Rx/OTC vs. Alcohol Interaction Heatmap:** A heatmap of cross-tabulated values between Rx_Otc and Alcohol_Interaction revealed interesting correlations, such as whether over-the-counter drugs tend to have alcohol interactions, useful for clinical interpretation.



Interpretation: The box plot displays the distribution of drug activity levels across a wide range of medical conditions. Certain conditions such as IBD(bowel), Gastrointestinal issues, and Stroke exhibit higher median activity levels and wider variability, indicating that the drugs used for these conditions are both frequently prescribed and have diverse usage rates. Conversely, conditions like Osteoporosis, Bronchitis, and GERD (Heartburn) show low median activity and relatively tight distributions, suggesting more limited or consistent prescribing patterns. A few conditions such as Acne, Menopause, and Insomnia also demonstrate notable activity spread, indicating variability in how often related drugs are used.

This variation in activity by condition is an important context for pregnancy safety modeling, as drugs used for high-activity conditions may carry greater risk due to widespread exposure. Incorporating condition-specific activity into the ML model may help improve predictions of pregnancy risk and guide safer prescribing decisions.

- **CSA vs. Activity Violin Plot:** A violin plot illustrates how drug activity varies across CSA categories. It captured both distribution spread and medians, useful for comparing regulation levels to pharmacological effectiveness.

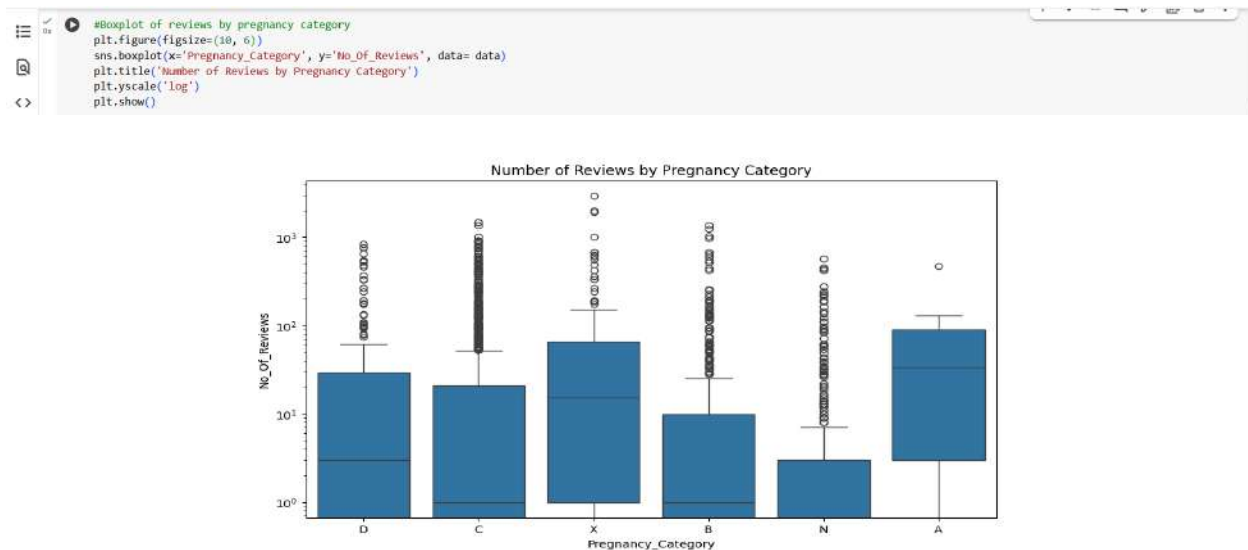


Interpretation: The violin plot visualizes the distribution of drug activity levels across different Controlled Substances Act (CSA) schedules, including non-controlled substances (N), Schedule 2–5 drugs, and miscellaneous categories (M). Non-controlled drugs (N) form the widest group, showing a large spread of activity levels with a dense concentration near zero, suggesting that most commonly used medications in the dataset are not federally scheduled substances. Schedules 2 and 4 display similar wide distributions, indicating that even tightly regulated drugs can have significant activity, possibly due to their critical therapeutic use in conditions like ADHD or anxiety. Schedule 5 and 3 drugs show moderate activity dispersion, while the 'M' category exhibits a narrow and low activity distribution, implying limited or niche usage.

The overall distribution pattern emphasizes that while many active drugs fall outside CSA scheduling, several high-activity medications still fall under controlled categories. This has important implications for pregnancy safety modeling, as CSA classification can act as a

potential feature indicating abuse potential or regulatory risk, both of which are relevant to maternal-fetal health considerations.

- **Pregnancy Category vs. Reviews Box Plot:** This box plot visualized the number of reviews across pregnancy categories, helping assess the reliability of public perception or patient-reported data within each risk group.

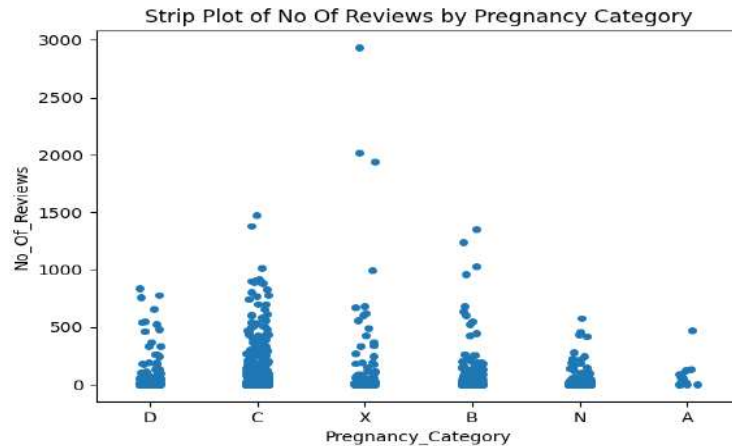


Interpretation: The box plot compares the distribution of user reviews across different pregnancy categories. The plot is displayed on a logarithmic scale due to the highly skewed distribution of review counts. Category 0 and Category 5, which may represent unknown or unclassified risk levels, exhibit higher median and upper-range review counts, suggesting these drugs are frequently discussed despite uncertain safety profiles. Categories 1 to 4, which likely correspond to FDA pregnancy risk categories (B, C, D, N), have lower median review counts and tighter interquartile ranges, with a large number of outliers in each group. This indicates that while many drugs within these categories are infrequently reviewed, a few receive disproportionate attention. The presence of high-variance and sparse reviews in several categories may reflect both uneven public awareness and the limited clinical data typically available for pregnancy-related drug safety.

These insights reinforce the importance of not overly relying on review counts as a standalone signal in predictive modeling. Instead, they should be used cautiously as a supplementary feature alongside clinical variables like side effects, drug class, and pregnancy risk level.

- **Pregnancy Category vs. Reviews Strip Plot:** The strip plot complemented the box plot by displaying the spread of individual review counts per pregnancy category, revealing outliers and overall review density.





Interpretation: The strip plot presents the distribution of drug review counts across different pregnancy categories, offering a granular look at individual data points. Pregnancy Category C, which generally represents drugs where risk cannot be ruled out, shows the widest and densest spread of reviews, including several drugs with over 1,000 reviews and a few like Category X exceeding 2,500. Categories D and X, typically associated with more significant risk, also display some highly reviewed drugs but with a narrower spread. Category B shows a moderately concentrated band, while Categories A and N (likely representing the safest or unclassified drugs) cluster in the lower range with fewer highly reviewed entries.

This visualization reinforces the notion that higher-risk drugs often receive more user feedback, possibly due to increased concern or more frequent usage. For our pregnancy safety classification model, this variability in review count may serve as a supporting indicator of clinical relevance, though care must be taken to avoid over-relying on popularity as a proxy for safety.

- **Extraction of Frequent Side Effects:** All side effect descriptions were merged into a single string, split into individual terms, and counted. This helped identify the top 1,000 most frequently mentioned side effects in the dataset. Understanding these high-frequency terms provided direction for identifying clinically relevant symptoms for further analysis.

```
[37] # Combine all text into a single string and split into individual terms
all_side_effects = " ".join(data['Side Effects'])
side_effect_list = all_side_effects.split()
side_effect_freq = Counter(side_effect_list)
most_common_effects = side_effect_freq.most_common(1000)
freq_df = pd.DataFrame(most_common_effects, columns=['Side Effect', 'Frequency'])
freq_df
```

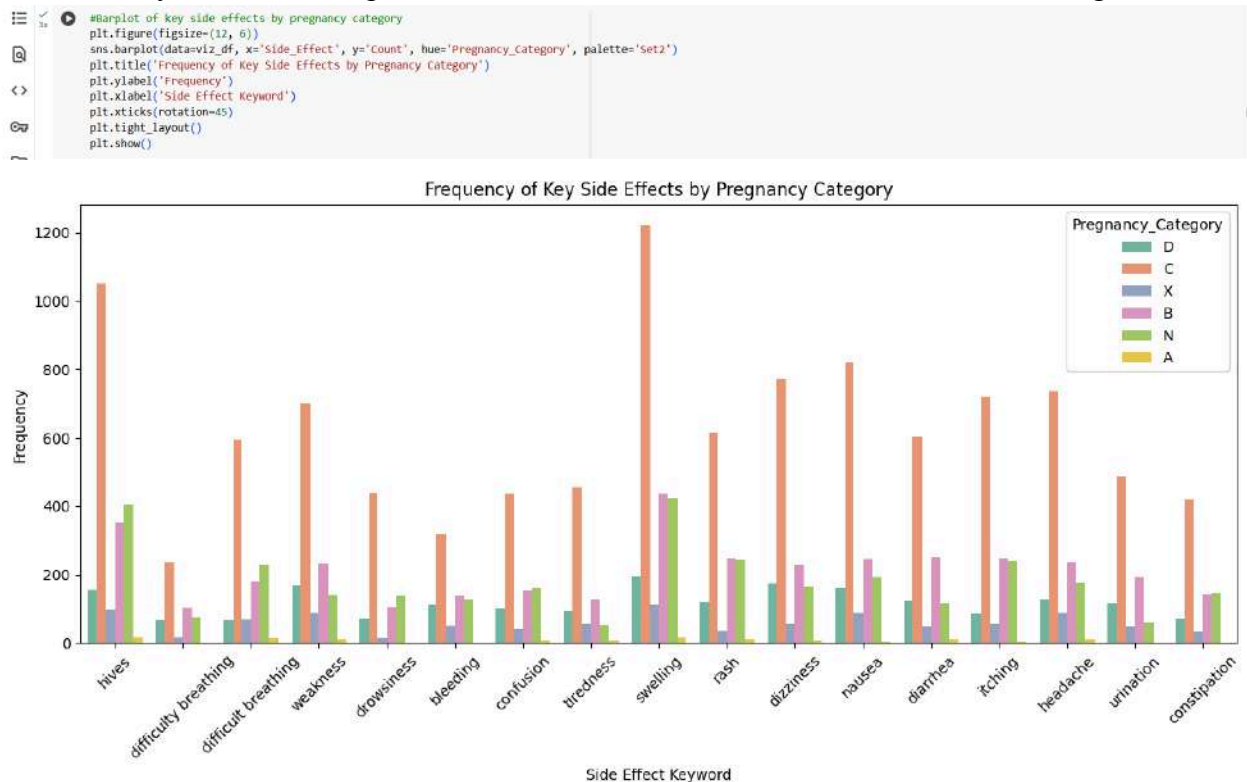
- **Keyword Filtering by Pregnancy Category:** A set of important symptom keywords (e.g., “drowsiness,” “bleeding,” “rash”) was used to examine how often each side effect appeared within each pregnancy risk category. This was done to explore whether certain adverse effects are more prevalent in high-risk drugs.


```
[38] #creating a dataframe for visualization purpose
keywords = ['hives', 'difficulty breathing', 'difficult breathing', 'weakness', 'drowsiness', 'bleeding', 'confusion', 'tiredness',
            'swelling', 'rash', 'dizziness', 'nausea', 'diarrhea', 'itching', 'headache', 'urination', 'constipation']
filtered_data = data[['Side_Effects', 'Pregnancy_Category']]
filtered_data['Side_Effects'] = filtered_data['Side_Effects'].str.lower()
results = []
for keyword in keywords:
    for category in filtered_data['Pregnancy_Category'].unique():
        count = filtered_data[
            (filtered_data['Pregnancy_Category'] == category) &
            (filtered_data['Side_Effects'].str.contains(keyword))
        ].shape[0]
        results.append({'Side_Effect': keyword, 'Pregnancy_Category': category, 'Count': count})
viz_df = pd.DataFrame(results)

/tmp/python-input-38-1401057813.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
filtered_data['Side_Effects'] = filtered_data['Side_Effects'].str.lower()
```

- **Bar Plot Visualization:** The results were visualized using a grouped bar plot, allowing clear comparison of key side effect frequencies across pregnancy categories. This plot revealed symptom trends and potential safety concerns linked to specific categories, which may aid healthcare professionals in risk communication and decision-making.



Interpretation: The bar chart displays the frequency of selected side effect keywords across different pregnancy categories, with Category C (possibly indicating moderate risk) dominating nearly every side effect type. Side effects such as swelling, hives, nausea, dizziness, and diarrhea are especially frequent in Category C drugs, suggesting that many medications in this group may carry common but potentially concerning adverse reactions during pregnancy. Category X drugs, which often indicate contraindication in pregnancy, appear less frequently across all side effects, likely reflecting clinical restrictions and lower usage in pregnant populations. Categories B and D show moderate presence across several symptoms, with rash, weakness, and headache being

notable across these categories. Category A (considered safest) consistently exhibits minimal side effect mentions, reinforcing its classification.

These patterns are valuable in guiding pregnancy-specific risk classification models; the high volume of certain side effects in moderate to high-risk categories may serve as predictive features in machine learning algorithms for identifying drugs that require greater caution when prescribed during pregnancy.

5. Statistical Analysis: To examine the relationships between our outcome variable (Pregnancy_Category) and various features, we performed two types of statistical tests:

- **Chi-Square Tests for Categorical Variables:** We conducted chi-square tests to assess whether categorical predictors, such as CSA classification, Rx/OTC status, alcohol interaction, drug class, and medical condition, were significantly associated with pregnancy safety categories. All tested variables showed statistically significant associations ($p < 0.05$), indicating their potential usefulness in predicting pregnancy risk levels.

- Chi-Square test between the pregnancy category and other categorical variables. It's null and alternate hypothesis are as follows:

1. CSA (Controlled Substance Act Classification)

H₀: Pregnancy category is independent of the CSA schedule of the drug.

H₁: Pregnancy category is associated with the CSA schedule of the drug.

2. Rx_Otc (Prescription or Over-the-Counter)

H₀: Pregnancy category is independent of whether the drug is prescription-only or over-the-counter.

H₁: Pregnancy category is associated with whether the drug is prescription-only or over-the-counter.

3. Alcohol_Interaction

H₀: Pregnancy category is independent of the presence of alcohol interaction warnings.

H₁: Pregnancy category is associated with the presence of alcohol interaction warnings.

4. Drug_Classes

H₀: Pregnancy category is independent of the drug class to which a medication belongs.

H₁: Pregnancy category is associated with the drug class.

5. Medical_Condition

H₀: Pregnancy category is independent of the medical condition for which the drug is used.

H₁: Pregnancy category is associated with the medical condition for which the drug is used.

```
#Define a function to perform chi-square test between the outcome and categorical variables
def chi_square_test(cat_col, target_col='Pregnancy_Category'):
    contingency_table = pd.crosstab(data[cat_col], data[target_col])
    chi2, p, dof, expected = chi2.contingency(contingency_table)
    print(f"\nChi-Square Test between '{cat_col}' and '{target_col}':")
    print(f"Chi2 Statistic = {chi2:.2f}, Degrees of Freedom = {dof}, p-value = {p:.4f}")
    if p < 0.05:
        print("There is a statistically significant association between the variables (p < 0.05).")
    else:
        print("There is no statistically significant association between the variables (p < 0.05).")

#List of categorical variables to test
categorical_vars = ['Csa', 'Rx_Otc', 'Alcohol_Interaction', 'Drug_Classes', 'Medical_Condition']
#Run chi-square test for each categorical variable
for cat in categorical_vars:
    if cat in data.columns:
        chi_square_test(cat)
```

```

Chi-Square Test between 'Csa' and 'Pregnancy_Category':
Chi2 Statistic = 157.47, Degrees of Freedom = 25, p-value = 0.0000
There is a statistically significant association between the variables (p < 0.05).

Chi-Square Test between 'Rx_Otc' and 'Pregnancy_Category':
Chi2 Statistic = 793.72, Degrees of Freedom = 10, p-value = 0.0000
There is a statistically significant association between the variables (p < 0.05).

Chi-Square Test between 'Alcohol_Interaction' and 'Pregnancy_Category':
Chi2 Statistic = 67.01, Degrees of Freedom = 5, p-value = 0.0000
There is a statistically significant association between the variables (p < 0.05).

Chi-Square Test between 'Drug_Classes' and 'Pregnancy_Category':
Chi2 Statistic = 9729.81, Degrees of Freedom = 1315, p-value = 0.0000
There is a statistically significant association between the variables (p < 0.05).

Chi-Square Test between 'Medical_Condition' and 'Pregnancy_Category':
Chi2 Statistic = 5046.65, Degrees of Freedom = 230, p-value = 0.0000
There is a statistically significant association between the variables (p < 0.05).

```

Interpretation: All tested variables: CSA, Rx_Otc, Alcohol_Interaction, Drug_Classes, Medical_Condition show a statistically significant association with the pregnancy category ($p < 0.05$). This suggests that features like prescription status, drug classification, and medical use context are meaningful in understanding or predicting pregnancy safety ratings of drugs. These variables are valuable inputs for model training and interpretation.

- **ANOVA for Continuous Variables:** We applied one-way ANOVA to assess whether numeric variables (Rating, No_Of_Reviews, and Activity) varied meaningfully across pregnancy categories. While results showed statistical significance, these features were ultimately excluded from the final model. This decision was based on practicality, such values are unlikely to be known or input by physicians using the app. Additionally, after testing model performance both with and without these variables, we observed no significant impact on prediction accuracy or outcomes, further justifying their removal.
- One-way ANOVA test between the pregnancy category and other continuous variables. It's null and alternate hypothesis are as follows:
 1. Rating
 - H₀: The average user rating is the same for drugs in all pregnancy categories.
 - H₁: At least one pregnancy category has a different average user rating compared to the others.
 2. No_Of_Reviews
 - H₀: The average number of reviews is the same for drugs across all pregnancy categories.
 - H₁: At least one pregnancy category has a different average number of reviews.
 3. Activity
 - H₀: The average activity level of drugs is the same across all pregnancy categories.
 - H₁: At least one pregnancy category has a different average activity level.

```

#Define a function to perform ANOVA test between the outcome and continuous variables
numeric_cols = ['Rating', 'No_Of_Reviews', 'Activity']
for col in numeric_cols:
    if col in data.columns:
        anova_df = data[['Pregnancy_Category', col]].dropna()
        grouped_data = [group[col].values for name, group in anova_df.groupby('Pregnancy_Category')]
        # Perform one-way ANOVA between the target and other continuous variables
        f_stat, p_val = f_oneway(*grouped_data)
        print(f"\nANOVA: Differences in '{col}': Across Pregnancy Categories")
        print(f"t-statistic = {f_stat:.2f}, p-value = {p_val:.4f}")
        if p_val < 0.05:
            print("There is a statistically significant difference in mean values across the categories (p < 0.05).")
        else:
            print("There is no statistically significant difference in mean values across the categories (p < 0.05).")

ANOVA: Differences in 'Rating' Across Pregnancy Categories
F-statistic = 12.72, p-value = 0.0000
There is a statistically significant difference in mean values across the categories (p < 0.05).

ANOVA: Differences in 'No_Of_Reviews' Across Pregnancy Categories
F-statistic = 11.58, p-value = 0.0000
There is a statistically significant difference in mean values across the categories (p < 0.05).

ANOVA: Differences in 'Activity' Across Pregnancy Categories
F-statistic = 15.85, p-value = 0.0000
There is a statistically significant difference in mean values across the categories (p < 0.05).

```

Interpretation: These continuous features show statistically significant differences across pregnancy categories ($p < 0.05$), indicating potential predictive power. However, they were excluded from the model as these values are not available inputs at the time of prediction and cannot be supplied by the end user.

- This code conducts preliminary statistical analysis to evaluate associations between predictor variables and the target pregnancy category. Chi-square tests were applied to categorical features (like Drug Class, CSA schedule) and one-way ANOVA was used for continuous variables (like Rating, Activity). Each test includes 95% confidence intervals to support interpretation. These tests help identify features with significant relationships to the outcome, guiding informed feature selection and validating their relevance before modeling. The results provided foundational insight into variable importance and potential predictive value.

```
from statsmodels.stats.weightstats import DescrStatsW
from scipy.stats import chi2, f_oneway
import warnings
warnings.filterwarnings("ignore")
chi_square_results = []
anova_results = []

# Chi-Square Test Function (with CI)
def chi_square_test_table(cat_col, target_col="Pregnancy_Category"):
    contingency_table = pd.crosstab(data[cat_col], data[target_col])
    chi2_stat, p_val, dof, expected = chi2_contingency(contingency_table)
    # calculate 95% CI for chi-square stat using chi-square distribution
    alpha = 0.05
    ci_lower = chi2.ppf(alpha / 2, df=dof)
    ci_upper = chi2.ppf(1 - alpha / 2, df=dof)
    chi_square_results.append({'Variable': cat_col, 'chi2': round(chi2_stat, 2), 'df': dof, 'p-value': round(p_val, 4), '95% CI (Chi2)': f'[{round(ci_lower, 2)}, {round(ci_upper, 2)}]'})
    # ANOVA Test Function (with CI for group means)
    def anova_test_table(col, target_col="Pregnancy_Category"):
        anova_df = data[[target_col, col]].dropna()
        groups = [group.col.values for name, group in anova_df.groupby(target_col)]
        f_stat, p_val = f_oneway(*groups)
        # 95% CI for each group mean
        ci_strings = []
        for name, group in anova_df.groupby(target_col):
            desc = DescrStatsW(group[col])
            mean = desc.mean
            ci_low, ci_high = desc.tconfint_mean(alpha=0.05)
            ci_strings.append(f'{name}: {mean:.2f} [{ci_low:.2f}, {ci_high:.2f}]')
        anova_results.append({'Variable': col, 'F-stat': round(f_stat, 2), 'p-value': round(p_val, 4), '95% CI (Group Means)': ' '.join(ci_strings),
                              'Significant': 'Yes' if p_val < 0.05 else 'No'})

# Run Tests
categorical_vars = ['Csa', 'Rx_Otc', 'Alcohol_Interaction', 'Drug_Classes', 'Medical_Condition']
for cat in categorical_vars:
    if cat in data.columns:
        chi_square_test_table(cat)
numeric_cols = ['Rating', 'No_Of_Reviews', 'Activity']
for col in numeric_cols:
    if col in data.columns:
        anova_test_table(col)

# Convert results to DataFrames
chi_square_df = pd.DataFrame(chi_square_results)
anova_df = pd.DataFrame(anova_results)

# Display in notebook
from IPython.display import display
print("\nChi-Square Test Results:")
display(chi_square_df)
print("\nANOVA Test Results:")
pd.set_option('display.max_colwidth', None)
display(anova_df)
```

Interpretation: A Chi-square test was performed to evaluate the association between categorical predictors and the outcome variable, Pregnancy_Category. The test results revealed that all five categorical variables, Csa, Rx_Otc, Alcohol_Interaction, Drug_Classes, and Medical_Condition, had statistically significant associations with the pregnancy safety category (all $p < 0.001$).

- Csa showed a moderate association ($\text{Chi}^2 = 157.47$, $\text{df} = 25$), with its statistic falling well outside the 95% confidence interval for random association [13.12, 40.65].
- Rx_Otc had a strong association ($\text{Chi}^2 = 793.72$, $\text{df} = 10$), indicating that the route of drug availability (prescription or over-the-counter) plays a significant role in determining safety classification.
- Alcohol_Interaction also demonstrated a significant relationship ($\text{Chi}^2 = 67.01$, $\text{df} = 5$), suggesting that warnings about alcohol may influence categorization.

- The variable `Drug_Classes` showed an extremely large χ^2 statistic (9729.81, $df = 1315$), confirming that drug class is a major determinant in how a drug is classified for pregnancy safety.
- Similarly, `Medical_Condition` showed strong statistical association ($\chi^2 = 5046.65$, $df = 230$), suggesting that therapeutic intent (e.g. conditions treated) is meaningfully linked with safety designation.

These results indicate that all examined categorical variables have statistically significant relationships with the FDA's pregnancy risk classification.

A one-way ANOVA was conducted to assess differences in the means of continuous variables across the pregnancy categories. All three numeric variables, `Rating`, `No_Of_Reviews`, and `Activity`, demonstrated statistically significant differences between groups (all $p < 0.001$), as detailed below:

- `Rating` ($F = 12.72$): The mean user ratings varied significantly across pregnancy categories. For instance, Category 0 (presumably "Safe") had the highest average rating (4.78; 95% CI: [3.07, 6.48]), while Category 1 and Category 2 showed lower means (e.g., 3.23 and 2.95 respectively), suggesting patient experience or satisfaction may be associated with safety perception.
- `No_Of_Reviews` ($F = 13.58$): The number of user-submitted reviews also differed significantly. Drugs in Category 0 had a much higher average review count (70.50; 95% CI: [9.28, 131.72]), which may reflect greater usage or public interest in drugs considered safer.
- `Activity` ($F = 15.85$): This metric, representing online drug activity or engagement, showed meaningful variance across categories. Drugs classified as safer (Category 0) had higher activity scores (16.38; 95% CI: [2.56, 30.19]), suggesting a possible relationship between perceived safety and online visibility or usage.

Together, these findings highlight that both categorical and continuous features carry significant predictive signals for pregnancy category classification, justifying their inclusion in model development.

6. Feature Engineering:

- **Label Encoding of Target Variable:** The `Pregnancy_Category` column, which was originally categorical (A, B, C, D, N, X), was label encoded into numerical values ranging from 0 to 5. This conversion is essential as most machine learning algorithms require numerical targets.
- **Class Balance Visualization:** A barplot was created to visualize the distribution of pregnancy categories after encoding. This helped us assess class imbalance and informed our decision on whether re-sampling or weighting techniques might be needed later in model training.


```

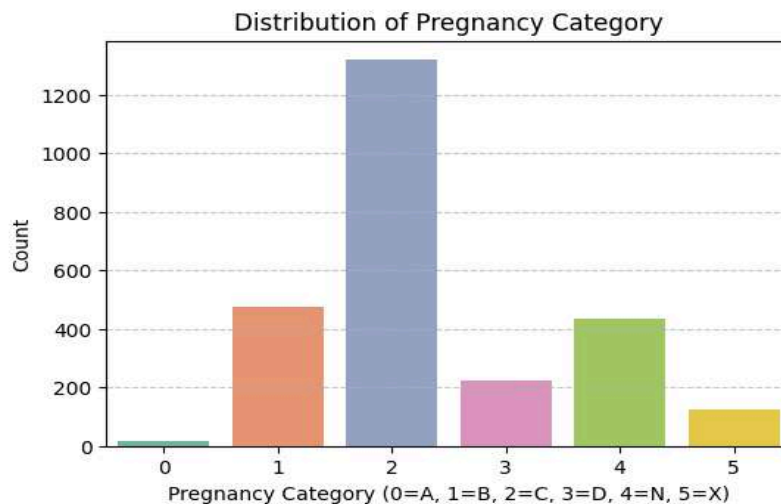
Transformations/ Feature Engineering

[42] #label encoding for target variable
label_encoder = LabelEncoder()
data['Pregnancy_Category'] = label_encoder.fit_transform(data['Pregnancy_Category'])
#checking the encoded labels for each category
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
print(label_mapping)

{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'N': 4, 'X': 5}

#Barplot of pregnancy category to view the class balance
plt.figure(figsize=(6, 4))
sns.countplot(x='Pregnancy_Category', hue='Pregnancy_Category', data=data, palette='Set2', legend=False)
plt.title("Distribution of Pregnancy Category")
plt.xlabel("Pregnancy Category (0=A, 1=B, 2=C, 3=D, 4=N, 5=X)")
plt.ylabel("Count")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



Interpretation: This bar chart shows the distribution of records across the different pregnancy safety categories. It's clear that the dataset is highly imbalanced, with Category 2 (which corresponds to Category C in the FDA system) being the most represented by a wide margin. In contrast, Categories 0 and 5 (which likely represent unknown or high-risk drugs, respectively) have very few instances.

- Label encoding for other categorical variables like Rx_Otc, Drug_Classes, and Medical_Condition is also performed to see the exact mapping of their values. Below are the pictures attached.

```

#Label encoding for target variable
data['Rx_Otc'] = data['Rx_Otc'].astype(str)
label_encoder = LabelEncoder()
data['Rx_Otc'] = label_encoder.fit_transform(data['Rx_Otc'])
#checking the encoded labels for each category
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
label_mapping

{'OTC': 0, 'Rx': 1, 'Rx/OTC': 2}

#Label encoding for target variable
data['Medical_Condition'] = data['Medical_Condition'].astype(str)
label_encoder = LabelEncoder()
data['Medical_Condition'] = label_encoder.fit_transform(data['Medical_Condition'])
#checking the encoded labels for each category
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
label_mapping

{'ADHD': 0,
'AIDS/HIV': 1,
'Acne': 2,
'Allergies': 3,
'Alzheimer's': 4,
'Angina': 5,
'Anxiety': 6,
'Asthma': 7,
'Bipolar Disorder': 8,
'Bronchitis': 9,
'COVID': 10,
'COVID 19': 11,

```

Output for Rx_Otc: {'OTC': 0, 'Rx': 1, 'Rx/OTC': 2}

Output for Medical_Condition: {'ADHD': 0, 'AIDS/HIV': 1, 'Acne': 2, 'Allergies': 3, 'Alzheimer's': 4, 'Angina': 5, 'Anxiety': 6, 'Asthma': 7, 'Bipolar Disorder': 8, 'Bronchitis': 9, 'COPD': 10, 'COVID 19': 11, 'Cancer': 12, 'Cholesterol': 13, 'Colds & Flu': 14, 'Constipation': 15, 'Depression': 16, 'Diabetes (Type 1)': 17, 'Diabetes (Type 2)': 18, 'Diarrhea': 19, 'Eczema': 20, 'Erectile Dysfunction': 21, 'GERD (Heartburn)': 22, 'Gastrointestinal': 23, 'Gout': 24, 'Hair Loss': 25, 'Hayfever': 26, 'Herpes': 27, 'Hypertension': 28, 'Hypothyroidism': 29, 'IBD (Bowel)': 30, 'Incontinence': 31, 'Insomnia': 32, 'Menopause': 33, 'Migraine': 34, 'Osteoarthritis': 35, 'Osteoporosis': 36, 'Pain': 37, 'Pneumonia': 38, 'Psoriasis': 39, 'Rheumatoid Arthritis': 40, 'Schizophrenia': 41, 'Seizures': 42, 'Stroke': 43, 'Swine Flu': 44, 'UTI': 45, 'Weight Loss': 46}

```
data['Drug_Classes'] = data['Drug_Classes'].astype(str)
label_encoder = LabelEncoder()
data['Drug_Classes'] = label_encoder.fit_transform(data['Drug_Classes'])
#checking the encoded labels for each category
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
label_mapping
```

```
{'5-aminosalicylates': 0,
 '5-aminosalicylates, Antirheumatics': 1,
 'ACE inhibitors with calcium channel blocking agents': 2,
 'ACE inhibitors with thiazides': 3,
 'AMPA receptor antagonists': 4,
 'Adamantane antivirals, Dopaminergic antiparkinsonism agents': 5,
 'Adrenergic bronchodilators': 6,
 'Adrenergic uptake inhibitors for ADHD': 7,
 'Agents for pulmonary hypertension, Impotence agents': 8,
 'Aldosterone receptor antagonists, Potassium-sparing diuretics': 9,
 'Alkylating agents': 10,
 'Allergens': 11,
 'Alpha-adrenoreceptor antagonists, Antiadrenergic agents, peripherally acting': 12,
 'Alpha-glucosidase inhibitors': 13,
 'Amebicides, Miscellaneous antibiotics': 14,
 'Aminoglycosides': 15,
 'Aminoglycosides, Inhaled anti-infectives': 16,
 'Aminopenicillins': 17,
 'Amylin analogs': 18,
 'Analgesic combinations': 19,
 'Angiotensin Converting Enzyme Inhibitors': 20,
 'Angiotensin II inhibitors with calcium channel blockers': 21,
 'Angiotensin II inhibitors with thiazides': 22,
```

Output for Medical_Condition: {'5-aminosalicylates': 0, '5-aminosalicylates, Antirheumatics': 1, 'ACE inhibitors with calcium channel blocking agents': 2, 'ACE inhibitors with thiazides': 3, 'AMPA receptor antagonists': 4, 'Adamantane antivirals, Dopaminergic antiparkinsonism agents': 5, 'Adrenergic bronchodilators': 6, 'Adrenergic uptake inhibitors for ADHD': 7, 'Agents for pulmonary hypertension, Impotence agents': 8, 'Aldosterone receptor antagonists, Potassium-sparing diuretics': 9, 'Alkylating agents': 10, 'Allergens': 11, 'Alpha-adrenoreceptor antagonists, Antiadrenergic agents, peripherally acting': 12, 'Alpha-glucosidase inhibitors': 13, 'Amebicides, Miscellaneous antibiotics': 14, 'Aminoglycosides': 15, 'Aminoglycosides, Inhaled anti-infectives': 16, 'Aminopenicillins': 17, 'Amylin analogs': 18, 'Analgesic combinations': 19, 'Angiotensin Converting Enzyme Inhibitors': 20, 'Angiotensin II inhibitors with calcium channel blockers': 21, 'Angiotensin II inhibitors with thiazides': 22, 'Angiotensin receptor blockers': 23, 'Anorectal preparations': 24, 'Anorexiant': 25, 'Anorexiant, CNS stimulants': 26, 'Anorexiant, CNS stimulants, Miscellaneous anticonvulsants': 27, 'Antacids': 28, 'Antacids, Laxatives': 29, 'Antacids, Minerals and electrolytes': 30, 'Antacids, Phosphate binders': 31, 'Anthelmintics': 32, 'Antiadrenergic agents (central) with thiazides': 33, 'Antiadrenergic agents, centrally acting': 34, 'Antiadrenergic agents, peripherally acting': 35, 'Antianginal agents': 36, 'Antianginal agents, Vasodilators': 37, 'Antiasthmatic combinations': 38, 'Antibiotics / antineoplastics': 39, 'Anticholinergic antiemetics': 40, 'Anticholinergic antiemetics, Anticholinergic antiparkinson agents, Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics': 41, 'Anticholinergic antiparkinson agents, Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics': 42, 'Anticholinergic bronchodilators': 43, 'Anticholinergics / antispasmodics': 44, 'Antidiabetic combinations': 45, 'Antidiarrheals': 46, 'Antidiarrheals, Probiotics': 47, 'Antigout agents': 48, 'Antigout agents, Antihyperuricemic agents': 49, 'Antihistamines': 50, 'Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics': 51, 'Antihistamines, Phenothiazine

antiemetics': 52, 'Antihyperlipidemic combinations': 53, 'Antihyperlipidemic combinations, Miscellaneous antihypertensive combinations': 54, 'Antihyperuricemic agents': 55, 'Antimalarial quinolines, Antirheumatics': 56, 'Antimanic agents': 57, 'Antimetabolites': 58, 'Antimetabolites, Antipsoriasis, Antirheumatics, Other immunosuppressants': 59, 'Antimigraine agents': 60, 'Antipseudomonal penicillins': 61, 'Antipsoriasis': 62, 'Antirheumatics': 63, 'Antirheumatics, CD20 monoclonal antibodies': 64, 'Antirheumatics, Interleukin inhibitors': 65, 'Antirheumatics, Other immunosuppressants': 66, 'Antirheumatics, Selective immunosuppressants': 67, 'Antirheumatics, TNF alfa inhibitors': 68, 'Antitussives, Opioids (narcotic analgesics)': 69, 'Antiviral boosters': 70, 'Antiviral boosters, Protease inhibitors': 71, 'Antiviral combinations': 72, 'Atypical antipsychotics': 73, 'Barbiturate anticonvulsants': 74, 'Barbiturate anticonvulsants, Barbiturates': 75, 'Barbiturates': 76, 'Benzodiazepine anticonvulsants': 77, 'Benzodiazepine anticonvulsants, Benzodiazepines': 78, 'Benzodiazepine anticonvulsants, Benzodiazepines, Miscellaneous antiemetics': 79, 'Benzodiazepines': 80, 'Beta blockers with thiazides': 81, 'Beta-lactamase inhibitors': 82, 'Bile acid sequestrants': 83, 'Bisphosphonates': 84, 'Bronchodilator combinations': 85, 'CD20 monoclonal antibodies': 86, 'CD52 monoclonal antibodies': 87, 'CNS stimulants': 88, 'Calcineurin inhibitors': 89, 'Calcitonin': 90, 'Calcium channel blocking agents': 91, 'Calcium channel blocking agents, Group IV antiarrhythmics': 92, 'Carbapenems': 93, 'Carbonic anhydrase inhibitor anticonvulsants': 94, 'Cardioselective beta blockers': 95, 'Cardioselective beta blockers, Group II antiarrhythmics': 96, 'Cephalosporins / beta-lactamase inhibitors': 97, 'Chemokine receptor antagonist': 98, 'Cholesterol absorption inhibitors': 99, 'Cholinesterase inhibitors': 100, 'Contraceptives': 101, 'Contraceptives, Sex hormone combinations': 102, 'Corticotropin': 103, 'Cox-2 inhibitors': 104, 'Dibenzazepine anticonvulsants': 105, 'Dipeptidyl peptidase 4 inhibitors': 106, 'Dopaminergic antiparkinsonism agents, Monoamine oxidase inhibitors': 107, 'Dopaminergic antiparkinsonism agents, Prolactin inhibitors': 108, 'Estrogens': 109, 'Expectorants': 110, 'Fatty acid derivative anticonvulsants': 111, 'Fibric acid derivatives': 112, 'First generation cephalosporins': 113, 'Fourth generation cephalosporins': 114, 'GI stimulants, Miscellaneous antiemetics': 115, 'Gamma-aminobutyric acid analogs': 116, 'Gamma-aminobutyric acid reuptake inhibitors': 117, 'General anesthetics': 118, 'Glucocorticoids': 119, 'Glucocorticoids, Inhaled corticosteroids': 120, 'Glycopeptide antibiotics': 121, 'Glycoprotein platelet inhibitors': 122, 'Glycylcyclines': 123, 'Gonadotropins': 124, 'Group I antiarrhythmics, Hydantoin anticonvulsants': 125, 'Group II antiarrhythmics, Non-cardioselective beta blockers': 126, 'H2 antagonists': 127, 'Heparins': 128, 'Hormones / antineoplastics, Selective estrogen receptor modulators': 129, 'Immune globulins': 130, 'Impotence agents': 131, 'Impotence agents, Vasodilators': 132, 'Incretin mimetics': 133, 'Inhaled anti-infectives, Neuraminidase inhibitors': 134, 'Inhaled corticosteroids': 135, 'Insulin': 136, 'Integrase strand transfer inhibitor': 137, 'Interferons': 138, 'Interleukin inhibitors': 139, 'Laxatives': 140, 'Leukotriene modifiers': 141, 'Lincomycin derivatives': 142, 'Local injectable anesthetics': 143, 'Loop diuretics': 144, 'Macrolides': 145, 'Mast cell stabilizers': 146, 'Meglitinides': 147, 'Minerals and electrolytes': 148, 'Miscellaneous GI agents': 149, 'Miscellaneous analgesics': 150, 'Miscellaneous antibiotics': 151, 'Miscellaneous anticonvulsants': 152, 'Miscellaneous antidepressants': 153, 'Miscellaneous antidepressants, Smoking cessation agents': 154, 'Miscellaneous antihyperlipidemic agents': 155, 'Miscellaneous antihyperlipidemic agents, Vitamins': 156, 'Miscellaneous antihypertensive combinations': 157, 'Miscellaneous antimalarials, Tetracyclines': 158, 'Miscellaneous antineoplastics': 159, 'Miscellaneous antineoplastics, Miscellaneous uncategorized agents': 160, 'Miscellaneous antipsychotic agents': 161, 'Miscellaneous antivirals': 162, 'Miscellaneous anxiolytics, sedatives and hypnotics': 163, 'Miscellaneous anxiolytics, sedatives and hypnotics,

Tricyclic antidepressants': 164, 'Miscellaneous bone resorption inhibitors': 165, 'Miscellaneous central nervous system agents': 166, 'Miscellaneous genitourinary tract agents': 167, 'Miscellaneous topical agents': 168, 'Miscellaneous uncategorized agents': 169, 'Mitotic inhibitors': 170, 'Monoamine oxidase inhibitors': 171, 'Mouth and throat products': 172, 'NNRTIs': 173, 'Narcotic analgesic combinations': 174, 'Nasal antihistamines and decongestants': 175, 'Nasal antihistamines and decongestants, Nasal steroids': 176, 'Nasal steroids': 177, 'Natural penicillins': 178, 'Neuraminidase inhibitors': 179, 'Next generation cephalosporins': 180, 'Non-cardioselective beta blockers': 181, 'Non-sulfonylureas': 182, 'Nonsteroidal anti-inflammatory drugs': 183, 'Nonsteroidal anti-inflammatory drugs, Salicylates': 184, 'Nucleoside reverse transcriptase inhibitors (NRTIs)': 185, 'Nutraceutical products': 186, 'Opioids (narcotic analgesics)': 187, 'Otic steroids': 188, 'Oxazolidinone antibiotics': 189, 'Parathyroid hormone and analogs': 190, 'Penicillinase resistant penicillins': 191, 'Peripheral opioid receptor antagonists': 192, 'Peripherally acting antiobesity agents': 193, 'Phenothiazine antiemetics, Phenothiazine antipsychotics': 194, 'Phenothiazine antipsychotics': 195, 'Phenylpiperazine antidepressants': 196, 'Platelet aggregation inhibitors': 197, 'Platelet aggregation inhibitors, Salicylates': 198, 'Potassium sparing diuretics with thiazides': 199, 'Potassium-sparing diuretics': 200, 'Probiotics': 201, 'Progestins': 202, 'Protease inhibitors': 203, 'Proton pump inhibitors': 204, 'Psoralens': 205, 'Psychotherapeutic combinations': 206, 'Purine nucleosides': 207, 'Pyrrolidine anticonvulsants': 208, 'Quinolones': 209, 'Renin inhibitors': 210, 'SGLT-2 inhibitors': 211, 'Salicylates': 212, 'Second generation cephalosporins': 213, 'Selective immunosuppressants': 214, 'Selective phosphodiesterase-4 inhibitors': 215, 'Selective serotonin reuptake inhibitors': 216, 'Serotonin-norepinephrine reuptake inhibitors': 217, 'Sex hormone combinations': 218, 'Skeletal muscle relaxant combinations': 219, 'Skeletal muscle relaxants': 220, 'Somatostatin and somatostatin analogs': 221, 'Statins': 222, 'Succinimide anticonvulsants': 223, 'Sulfonamides': 224, 'Sulfonylureas': 225, 'TNF alfa inhibitors': 226, 'Tetracyclic antidepressants': 227, 'Tetracyclines': 228, 'Thiazide diuretics': 229, 'Thiazolidinediones': 230, 'Thioxanthenes': 231, 'Third generation cephalosporins': 232, 'Thrombin inhibitors': 233, 'Thrombolytics': 234, 'Thyroid drugs': 235, 'Topical acne agents': 236, 'Topical acne agents, Topical anti-rosacea agents': 237, 'Topical acne agents, Topical antibiotics': 238, 'Topical acne agents, Topical antipsoriatics': 239, 'Topical acne agents, Topical keratolytics': 240, 'Topical acne agents, Vaginal anti-infectives': 241, 'Topical anesthetics': 242, 'Topical anti-infectives': 243, 'Topical antibiotics': 244, 'Topical antihistamines': 245, 'Topical antipsoriatics': 246, 'Topical antivirals': 247, 'Topical emollients': 248, 'Topical non-steroidal anti-inflammatories': 249, 'Topical rubefacient': 250, 'Topical steroids': 251, 'Topical steroids with anti-infectives': 252, 'Triazine anticonvulsants': 253, 'Tricyclic antidepressants': 254, 'Unknown': 255, 'Upper respiratory combinations': 256, 'Urinary anti-infectives': 257, 'Urinary antispasmodics': 258, 'Uterotonic agents': 259, 'Vasodilators': 260, 'Viscosupplementation agents': 261, 'Vitamin and mineral combinations': 262, 'Vitamins': 263}

- **Encoding of Categorical Predictors:** Categorical features like Csa, Rx_Otc, Alcohol_Interaction, Drug_Classes, and Medical_Condition were also label encoded to ensure compatibility with model inputs. This step allowed us to preserve categorical distinctions while converting them into numerical form.
- **Transformation and Scaling of Numerical Features:** Numeric features like Activity and No_Of_Reviews were skewed, so we applied log transformations to normalize their distribution. We then scaled them using standardization to bring all values to a comparable range, an important step for improving model convergence and accuracy.

```
[44] #label encoding the predictor categorical columns
categorical_cols = ['Csa', 'Rx_Otc', 'Alcohol_Interaction', 'Drug_Classes', 'Medical_Condition']
for col in categorical_cols:
    if col in data.columns:
        data[col] = label_encoder.fit_transform(data[col])

#Log transformations and scaling for the numeric features.
data['Activity_log'] = np.log1p(data['Activity'])
data['No_Of_Reviews_log'] = np.log1p(data['No_Of_Reviews'])
features_to_scale = ['Activity_log', 'Rating', 'No_Of_Reviews_log']
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data[features_to_scale])
```

- **Post-Transformation Summary:** A final describe() check confirmed the transformed dataset, allowing us to verify appropriate scaling, encoding, and handling of outliers or invalid entries. This step assured that the dataset was now clean, balanced, and ready for modeling.

```
data.describe()
```

| | Medical_Condition | Drug_Classes | Activity | Rx_Otc | Pregnancy_Category | Csa | Rating | No_Of_Reviews | Alcohol_Interaction | Activity_log | No_Of_Reviews_log |
|-------|-------------------|--------------|-------------|-------------|--------------------|-------------|-------------|---------------|---------------------|--------------|-------------------|
| count | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 | 2591.000000 |
| mean | 21.436897 | 157.548051 | 8.624469 | 1.093786 | 2.368198 | 4.698186 | 3.352258 | 43.026631 | 0.488614 | 1.327883 | 1.576701 |
| std | 13.301747 | 81.847437 | 17.380298 | 0.572271 | 1.122040 | 1.086239 | 4.260288 | 147.485916 | 0.499967 | 1.253508 | 1.906496 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 10.000000 | 85.000000 | 0.000000 | 1.000000 | 2.000000 | 5.000000 | -1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 20.000000 | 168.000000 | 2.000000 | 1.000000 | 2.000000 | 5.000000 | 4.000000 | 1.000000 | 0.000000 | 1.098612 | 0.693147 |
| 75% | 35.000000 | 240.000000 | 7.000000 | 1.000000 | 3.000000 | 5.000000 | 7.400000 | 17.000000 | 1.000000 | 2.079442 | 2.890372 |
| max | 46.000000 | 263.000000 | 100.000000 | 2.000000 | 5.000000 | 5.000000 | 10.000000 | 2934.000000 | 1.000000 | 4.615121 | 7.984463 |

7. **Natural Language Processing:** To extract meaningful insights from the unstructured Side_Effects text column, we applied a series of NLP techniques aimed at cleaning, standardizing, and preparing the data for analysis.

- **Initial Text Cleaning:** All side effect entries were converted to lowercase and stripped of non-alphabetic characters. This standardization helped reduce noise and ensured uniformity in text processing.

```
Natural Language Processing

[47] #cleaning the text column by removing non-alphabetic characters, and making the words standardized by converting into lower case
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z ]+', '', text)
    return text

[48] data['Side_Effects'] = data['Side_Effects'].apply(clean_text)

data['Side_Effects']
```

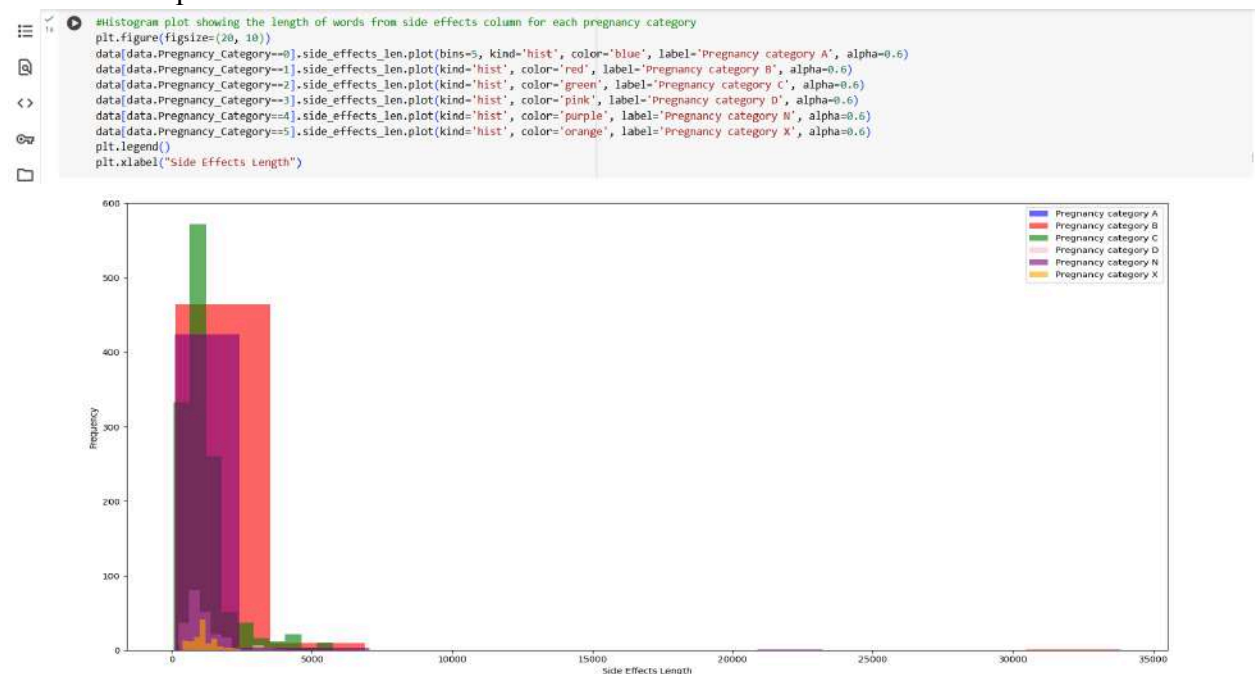
| | Side_Effects |
|------|---|
| 0 | hives difficult breathing swelling in your fa... |
| 1 | hives difficulty breathing swelling of your fa... |
| 2 | skin rash fever swollen glands flu like sympto... |
| 3 | problems with your vision or hearing muscle or... |
| 4 | hives difficult breathing swelling of your fac... |
| ... | ... |
| 2923 | hives difficult breathing swelling of your fac... |
| 2924 | hives difficult breathing swelling of your fac... |

- **Length Feature Creation:** We calculated the character length of each cleaned side effects entry and stored it in a new feature, side_effects_len. This numeric representation offered a quick glimpse into the textual density of drug warnings.

```
#Creating a column showing the length of the words from side effects column for each entry
data['side_effects_len'] = data.Side_Effects.apply(len)
data.head()
```

| | Drug_Name | Medical_Condition | Side_Effects | Generic_Name | Drug_Classes | Brand_Names | Activity | Rx_Otc | Pregnancy_Category | Csa | Rating | No_Of_Reviews | Alcohol_Interaction | Activity |
|---|--------------|-------------------|---|---------------------|--------------|---|----------|--------|--------------------|-----|--------|---------------|---------------------|----------|
| 0 | Doxycycline | 2 | hives difficult breathing swelling in your fa... | Doxycycline | 158 | Adiclate, Adoxa CK, Adoxa Pak, Adoxa TT, Alod... | 87 | 1 | 3 | 5 | 6.8 | 760 | 1 | 4.47 |
| 1 | Spirolactone | 2 | hives difficulty breathing swelling of your fa... | Spirolactone | 9 | Aldactone, CaroSpr | 82 | 1 | 2 | 5 | 7.2 | 449 | 1 | 4.41 |
| 2 | Minocycline | 2 | skin rash fever swollen glands flu like sympto... | Minocycline | 228 | Dynacin, Minocin, Minolira, Solodyn, Ximino, V... | 48 | 1 | 3 | 5 | 5.7 | 482 | 0 | 3.89 |
| 3 | Accutane | 2 | problems with your vision or hearing muscle or... | Isotretinoin (Oral) | 160 | Unknown | 41 | 1 | 5 | 5 | 7.9 | 623 | 1 | 3.73 |
| 4 | Clindamycin | 2 | hives difficult breathing swelling of your fac... | Clindamycin Topical | 241 | Cleocin T, Clindacin ETZ, Clindacin P, Clindacin | 39 | 1 | 1 | 5 | 7.4 | 146 | 0 | 3.68 |

- Length Distribution Analysis:** A histogram was plotted to show how the length of side effects varies across different pregnancy categories. This visual helped reveal whether drugs in higher-risk categories tend to have longer or more detailed adverse effect descriptions.



Interpretation: The histogram illustrates the distribution of side effects length across different pregnancy categories, reflecting how verbose or detailed the listed side effects are for each drug. Most drugs fall within a side effect length of under 5,000 characters, with the highest density observed between 500 and 2,500. Pregnancy Categories B, C, and N dominate this range, indicating that drugs in these categories typically have longer or more descriptive side effect documentation. A few outliers, especially in Categories B and N, stretch beyond 10,000 characters, suggesting rare cases with extensive reported side effects.

In contrast, Categories A and D show more compact distributions, which may be due to fewer known adverse effects or more conservative documentation. Category X shows moderate spread, potentially capturing drugs with contraindications that require more detailed warnings. These patterns are relevant to machine learning feature engineering, as side effect text length may indirectly signal risk level or regulatory caution, especially in high-risk or less-studied categories. Including side effect length as a feature may enhance model sensitivity to pharmacovigilance-related variables in the classification of pregnancy-safe drugs.

- **Raw Text Observation:** We reviewed specific entries (e.g: the one with a length of 1325) to confirm the richness and variability of the content. Many entries included detailed medical symptoms, making them prime candidates for deeper text mining.

```
[52] data[data.Pregnancy_Category==1].describe()
```

| | Medical_Condition | Drug_Classes | Activity | Rx_Otc | Pregnancy_Category | Csa | Rating | No_Of_Reviews | Alcohol_Interaction | Activity_log | No_Of_Reviews_Log | side_effect |
|-------|-------------------|--------------|------------|------------|--------------------|------------|------------|---------------|---------------------|--------------|-------------------|-------------|
| count | 476.000000 | 476.000000 | 476.000000 | 476.000000 | 476.0 | 476.000000 | 476.000000 | 476.000000 | 476.000000 | 476.000000 | 476.000000 | 476.0 |
| mean | 21.376050 | 141.932773 | 13.306723 | 1.069328 | 1.0 | 4.962185 | 3.226891 | 34.071429 | 0.369748 | 1.662817 | 1.380955 | 1073.2 |
| std | 13.104082 | 75.510895 | 22.525645 | 0.508273 | 0.0 | 0.418819 | 4.248030 | 129.370114 | 0.483244 | 1.375034 | 1.706330 | 1728.1 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.0 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 149.0 |
| 25% | 9.000000 | 72.000000 | 1.000000 | 1.000000 | 1.0 | 5.000000 | -1.000000 | 0.000000 | 0.000000 | 0.693147 | 0.000000 | 547.0 |
| 50% | 22.000000 | 140.000000 | 3.000000 | 1.000000 | 1.0 | 5.000000 | 3.300000 | 1.000000 | 0.000000 | 1.386294 | 0.693147 | 724.5 |
| 75% | 35.000000 | 207.000000 | 13.000000 | 1.000000 | 1.0 | 5.000000 | 7.200000 | 10.000000 | 1.000000 | 2.639057 | 2.397899 | 1166.5 |
| max | 46.000000 | 258.000000 | 100.000000 | 2.000000 | 1.0 | 5.000000 | 10.000000 | 1367.000000 | 1.000000 | 4.815121 | 7.213768 | 33816.0 |

```
[53] #Each instance of side effects column looks like this
data[data.side_effects_len == 1325].Side_Effects.iloc[0]
```

'hives difficult breathing swelling in your face or throat or a severe skin reaction fever sore throat burning in your eyes skin pain red or purple skin rash that spreads and causes blistering and peeling seek medical treatment if you have a serious drug reaction that can affect many parts of your body symptoms may include skin rash fever swollen glands flu like symptoms muscle aches severe weakness unusual bruising or yellowing of your skin or eyes this reaction may occur several weeks after you began using doxycycline doxycycline may cause serious side effects call your doctor at once if you have severe stomach pain diarrhea that is watery or bloody throat irritation trouble swallowing chest pain irregular heart rhythm feeling short of breath little or no urination low white blood cell counts fever chills swollen glands body aches weakness pale skin easy bruising or bleeding severe headaches ringing in your ears dizziness nausea vision problems pain behind your eyes loss of appetite'

- **Stopword Removal & Custom Filtering:** We created a clean_side_effects column by removing punctuation, common English stopwords, and domain-specific filler terms (like drug names and vague phrases). This focused the data on medically relevant terms.
- **Text Comparison:** Comparing the original and cleaned versions of side effects text helped validate that only non-informative or repetitive terms were removed, while key clinical descriptors remained intact (e.g: “hives,” “dizziness,” “rash”).

```
#Processing the text to remove punctuations, and stop words
def text_process(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
    3. Returns a list of the cleaned text
    """
    STOPWORDS = stopwords.words('english') + ['you', 'immediately', 'professional', 'your', 'medicine', 'doctor', 'call', 'may', 'might', 'check', 'include', 'side', 'effects', 'you', 'can', 'take', 'get', 'typeof', 'icon', 'just', 'contains', 'symptoms', 'take', 'need', 'using', 'causing', 'causes', 'have', 'seek', 'medical', 'within', 'treatment', 'be', 'loading', 'g', 'many', 'parts', 'reaction', 'several', 'severe', 'none', 'much', 'return', 'behind', 'away', 'taking', 'such', 'need', 'go', 'likely', 'questions', 'bothersome', 'also', 'these', 'where', 'those', 'this', 'that', 'an', 'a', 'or', 'of', 'to', 'have', 'be', 'at', 'common', 'very', 'with', 'and', 'if', 'as', 'on', 'not', 'like', 'occur', 'is', 'are', 'ways', 'was', 'yourself', 'using', 'should', 'need', 'will', 'by', 'still', 'within', 'happen', 'behind', 'few', 'called', 'just', 'both', 'too', 'who', 'does', 'could', 'has', 'given', 'material', 'occurs', 'been', 'over', 'first', 'after', 'before', 'rare', 'clay', 'without', 'cause', 'signs', 'following', 'tell', 'sometimes', 'people', 'signs', 'stop', 'although', 'around', 'fda', 'especially', 'so', '1988', '12', '24', '30', '400', '65', '800', 'abacavir', 'able', 'acetaminophen', 'advil', 'african', 'ala', 'allegria', 'american', 'amlodipine', 'among', 'ammonia', 'anthralin', 'antibiotic', 'anywhere', 'ask', 'appear', 'apply', 'aspirin', 'bandage', 'benzocaine', 'benzoyl', 'caffeine', 'calcium', 'canker', 'capsaicin', 'cause', 'cdad', 'chance', 'change', 'case', 'chlorpheniramine', 'claritin', 'codeine', 'coffee', 'come', 'contain', 'cr', 'cover', 'count', 'cont', 'continue', 'continuous', 'd', 'dextromethorphan', 'dextromethorphan', 'diphenhydramine', 'di', 'dos', 'doxycycline', 'doxylamine', 'ee', 'eg', 'empty', 'emtricitabine', 'enjoy', 'er', 'false', 'forte', 'get', 'glo', 'hc', 'hct', 'humalog', 'humulin', 'hydrochlorothiazide', 'hydrocodone', 'hydrocortisone', 'ibuprofen', 'insulin', 'iodoquinol', 'iv', 'johnson', 'kwikpen', 'la', 'lab', 'lamivudine', 'lead', 'magnesia', 'magnesium', 'mee', 'meet', 'menthol', 'metformin', 'methylene', 'methyl', 'mometasone', 'motrin', 'mucinex', 'naloxone', 'need', 'never', 'not', 'novolog', 'oh', 'one', 'orinoid', 'others', 'nas', 'net', 'net', 'necode', 'nepharmacist', 'phenylephrine', 'pinpoint', 'pl', 'place', 'metylene', 'methyl', 'mometasone', 'motrin', 'mucinex', 'naloxone', 'need', 'never', 'not', 'novolog', 'on', 'one', 'opoid', 'others', 'pas', 'pe', 'per', 'peroxide', 'pharmacist', 'phenylephrine', 'pinpoint', 'pl', 'place', 'plus', 'potassium', 'promexine', 'pseudoephedrine', 'pu', 'sallylate', 'serotonin', 'sodium', 'something', 'steroid', 'stevens', 'sure', 'take', 'tar', 'temofovir', 'theraflu', 'tylenol', 'unit', 'use', 'warn', 'wind', 'xl', 'x', 'yet', 'well', 'wear', 'way']

    nopunc = [char for char in mess if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    return ' '.join(word for word in nopunc.split() if word.lower() not in STOPWORDS)
```

```
data['clean_side_effects'] = data.Side_Effects.apply(text_process)
data.head()
```

| | Drug_Name | Medical_Condition | Side_Effects | Generic_Name | Drug_Classes | Brand_Names | Activity | Rx_Otc | Pregnancy_Category | Csa | Rating | No_Of_Reviews | Alcohol_Interaction | Activity_log |
|---|---------------|-------------------|---|---------------|--------------|---|----------|--------|--------------------|-----|--------|---------------|---------------------|--------------|
| 0 | Doxycycline | 2 | hives difficult breathing swelling in your fa... | Doxycycline | 158 | AdScilate, Adoxa CK, Adoxa Pak, Adoxa TT, Alod... | 87 | 1 | 3 | 5 | 6.8 | 760 | 1 | 4.47 |
| 1 | Spiroglactone | 2 | hives difficulty breathing swelling of your fa... | Spiroglactone | 9 | Aldactone, CaroSpr | 82 | 1 | 2 | 5 | 7.2 | 449 | 1 | 4.41 |
| 2 | Minocycline | 2 | skin rash fever swollen glands flu like sympto... | Minocycline | 228 | Dynacin, Minocin, Minolira, Solodyn, Ximino, V... | 48 | 1 | 3 | 5 | 5.7 | 482 | 0 | 3.89 |

- **Lemmatization:** The cleaned text was further processed using lemmatization, which reduced each word to its base form (e.g: “swelling” became “swell”). This step ensured that different grammatical forms of the same word were treated uniformly.

- **Output Preview:** After full preprocessing, the text was significantly more compact and clinically focused. For example, an original phrase like “hives difficult breathing swelling of your face” was reduced to “hive difficult breathe swell face”, making it more suitable for text vectorization and modeling.

```
[56] #lemmatizing the side effects column to reduce word variations to their base or dictionary form
lemmatizer = WordNetLemmatizer()
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
def lemmatize_text(text):
    words = word_tokenize(text)
    pos_tags = pos_tag(words)
    lemmatized_words = [lemmatizer.lemmatize(word, get_wordnet_pos(tag)) for word, tag in pos_tags]
    return ' '.join(lemmatized_words)

[57] data['clean_side_effects'] = data['clean_side_effects'].apply(lemmatize_text)

#comparing the cleaned side effects column with the original side effects column
print(data[['side_effects', 'clean_side_effects']].sample(5))
```

| | side_effects | clean_side_effects |
|------|---|-----------------------------------|
| 436 | along with its needed effects a medicine may c... | |
| 1298 | nausea upper stomach pain itching loss of appe... | |
| 1533 | hives difficulty breathing swelling of your fa... | hive difficult breathe swell face |
| 520 | hives difficulty breathing swelling of your fa... | hive difficult breathe swell face |

- **Extracting Common Side Effects for Pregnancy Category A:** In this step, we filtered the dataset for drugs labeled under Pregnancy Category A and tokenized the cleaned side effect text into individual lowercase words. Using Counter, we identified the 100 most frequently mentioned side effects. This provided a clear view of the most common and relatively mild reactions (e.g: pain, headache, appetite changes) associated with the safest drug category.

```
[59] data.clean_side_effects.head()
```

| | clean_side_effects |
|---|---|
| 0 | hive difficult breathe swell face throat skin ... |
| 1 | hive difficulty breathe swell face lip tongue ... |
| 2 | skin rash fever swell gland flu muscle ache we... |
| 3 | problem vision hear muscle joint pain bone pai... |
| 4 | hive difficult breathe swell face lip tongue L... |

```
dtype: object

#viewing the 100 most common words from side effects column related to pregnancy category 0-> A
words = data[data.Pregnancy_Category==0].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_0_words = Counter()
for effects in words:
    cat_0_words.update(effects)
cat_0_words.most_common(100)
```

Output: [('pain', 37), ('loss', 32), ('feel', 31), ('breathe', 22), ('irregular', 22), ('headache', 22), ('change', 22), ('hair', 21), ('chest', 20), ('face', 19), ('appetite', 18), ('swell', 17), ('weakness', 17), ('skin', 16), ('weight', 16), ('hive', 15), ('difficult', 15), ('lip', 15), ('tongue', 15), ('throat', 15), ('trouble', 15), ('heartbeat', 14), ('irritable', 14), ('menstrual', 14), ('feeling', 14), ('fast', 13), ('tremor', 13), ('leg', 13), ('muscle', 13), ('diarrhea', 13), ('hot', 12), ('problem', 12), ('nervous', 12), ('child', 11), ('cramp', 11), ('discomfort', 11), ('neck', 11), ('jaw', 10), ('sweat', 10), ('cold', 10), ('sleep', 10), ('period', 10), ('increase', 10), ('less', 9), ('shortness', 9), ('breath', 9), ('rash', 9), ('sudden', 9), ('fever', 8), ('tiredness', 8), ('vomit', 8), ('eye', 8), ('attention', 8), ('pulse', 8), ('arm', 8), ('health', 8), ('care', 8), ('dizziness', 7), ('slow', 7), ('unusual', 7), ('temporary', 6), ('spread', 6), ('shoulder', 6), ('insomnia', 6), ('pound', 6), ('weak', 6), ('confusion', 6), ('vision', 6), ('redness', 6), ('consciousness', 6), ('lightheadedness', 5), ('flash', 4), ('unusually', 4), ('memory', 4), ('depress', 4), ('ache', 4), ('dryness', 4), ('certain', 4), ('old', 4), ('adult', 4), ('partial', 4), ('need', 4), ('unwanted', 4), ('tightness', 4), ('decrease', 4), ('urine', 4), ('output', 4), ('dilate', 4), ('vein', 4),

('extreme', 4), ('fatigue', 4), ('irritability', 4), ('nausea', 4), ('back', 4), ('blur', 4), ('overdose', 4), ('clammy', 4), ('usually', 4), ('body', 4), ('adjust', 4)]

- **Extracting Common Side Effects for Pregnancy Category B:** Here, we repeated the same process for Pregnancy Category B, which includes drugs with no evidence of risk in humans. The results showed a higher frequency and intensity of symptoms like swelling, hive, diarrhea, dizziness, and urination issues, helping us understand how perceived side effects begin to shift as we move into slightly riskier medications.

```
#viewing the 100 most common words from side effects column related to pregnancy category 1-> B
words = data[data.Pregnancy_Category==1].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_1_words = Counter()
for effects in words:
    cat_1_words.update(effects)
cat_1_words.most_common(100)
```

Output: [('pain', 1262), ('skin', 1002), ('swell', 907), ('throat', 650), ('feel', 605), ('breathe', 528), ('stomach', 528), ('face', 444), ('tongue', 437), ('loss', 409), ('lip', 407), ('diarrhea', 405), ('fever', 403), ('rash', 379), ('eye', 375), ('hive', 374), ('problem', 363), ('weakness', 334), ('vomit', 325), ('muscle', 323), ('nausea', 319), ('urine', 312), ('mouth', 311), ('unusual', 307), ('chest', 307), ('trouble', 303), ('dizziness', 295), ('blood', 292), ('itch', 289), ('sore', 286), ('headache', 284), ('serious', 276), ('low', 272), ('urination', 267), ('difficult', 260), ('heartbeat', 259), ('breath', 244), ('weight', 244), ('cough', 241), ('foot', 237), ('redness', 229), ('feeling', 223), ('b', 221), ('infection', 217), ('red', 212), ('fast', 209), ('heart', 206), ('body', 205), ('1', 203), ('dry', 198), ('vision', 189), ('bloody', 186), ('yellow', 185), ('color', 184), ('increase', 182), ('constipation', 181), ('right', 173), ('burn', 170), ('confusion', 170), ('difficulty', 169), ('numbness', 167), ('change', 163), ('appetite', 162), ('tiredness', 161), ('drowsiness', 160), ('even', 159), ('blister', 155), ('irregular', 154), ('bruise', 152), ('dark', 152), ('cold', 150), ('back', 149), ('new', 149), ('hand', 147), ('joint', 147), ('sweat', 144), ('0', 144), ('bleed', 141), ('upper', 138), ('gain', 138), ('nose', 137), ('light', 136), ('swallow', 134), ('legs', 133), ('seizure', 128), ('tingle', 126), ('arm', 125), ('attention', 124), ('movement', 123), ('c', 122), ('painful', 121), ('liver', 121), ('chill', 121), ('mild', 119), ('pas', 118), ('sudden', 117), ('gland', 115), ('watery', 114), ('peel', 113), ('bad', 113)]

- **Extracting Common Side Effects for Pregnancy Category C:** In this step, we analyzed **Category C** drugs, where risk cannot be ruled out. The output indicated a substantial rise in terms associated with more severe discomforts such as vomiting, vision problems, and heartbeat irregularities. This helped distinguish Category C with a richer and denser side effect vocabulary compared to earlier categories.

```
#viewing the 100 most common words from side effects column related to pregnancy category 2-> C
words = data[data.Pregnancy_Category==2].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_2_words = Counter()
for effects in words:
    cat_2_words.update(effects)
cat_2_words.most_common(100)
```

Output: [('skin', 3242), ('pain', 2906), ('swell', 2159), ('throat', 1750), ('breathe', 1688), ('feel', 1538), ('face', 1518), ('stomach', 1380), ('lip', 1269), ('loss', 1269), ('nausea', 1235), ('tongue', 1173), ('hive', 1129), ('vomit', 1055), ('problem', 1045), ('dizziness', 1037), ('eye', 1004), ('muscle', 988), ('weakness', 979), ('increase', 964), ('chest', 923), ('headache', 919), ('mouth', 918), ('fever', 913), ('unusual', 892), ('trouble', 892), ('feeling', 882), ('change', 872), ('blood', 871), ('vision', 869), ('difficult', 825), ('serious', 819), ('itch', 802), ('heartbeat', 769), ('urine', 769), ('diarrhea', 766), ('rash', 765), ('fast', 762), ('redness', 748), ('body', 700), ('breath', 692), ('light', 673), ('urination', 626), ('weight', 626), ('dry', 609), ('tiredness', 599), ('sore', 592), ('red', 587), ('burn', 577), ('drowsiness', 572), ('attention', 566), ('heart', 552), ('color', 547), ('sweat', 547), ('appetite', 546), ('confusion', 531), ('slow', 527), ('rate', 491), ('right', 490), ('low', 482), ('nose',

482), ('back', 480), ('cough', 479), ('foot', 477), ('difficulty', 475), ('constipation', 469), ('bad', 444), ('head', 442), ('sleep', 430), ('upper', 425), ('blur', 423), ('care', 417), ('irritation', 414), ('gain', 406), ('blister', 398), ('numbness', 397), ('pas', 396), ('movement', 392), ('peel', 383), ('yellow', 379), ('hand', 366), ('irregular', 363), ('health', 363), ('cold', 355), ('stool', 353), ('high', 352), ('neck', 342), ('ear', 341), ('hair', 335), ('mood', 333), ('child', 330), ('swallow', 318), ('legs', 318), ('dark', 314), ('healthcare', 313), ('worsen', 309), ('arm', 305), ('mild', 304), ('report', 304), ('treat', 303)]

- **Extracting Common Side Effects for Pregnancy Category D:** For Pregnancy Category D (positive evidence of human fetal risk), our word frequency analysis captured terms like confusion, bruising, jaundice, and tiredness. These findings indicate heightened physiological impacts, emphasizing the critical need to monitor side effects for these medications.

```
#viewing the 100 most common words from side effects column related to pregnancy category 3-> D
words = data[data.Pregnancy_Category==3].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_3_words = Counter()
for effects in words:
    cat_3_words.update(effects)
cat_3_words.most_common(100)
```

Output: [('pain', 706), ('skin', 577), ('swell', 420), ('feel', 393), ('loss', 334), ('throat', 294), ('problem', 294), ('stomach', 264), ('weakness', 256), ('eye', 248), ('muscle', 239), ('unusual', 235), ('breathe', 227), ('nausea', 227), ('fever', 225), ('face', 208), ('dizziness', 207), ('sore', 206), ('feeling', 200), ('chest', 197), ('trouble', 194), ('lip', 191), ('vomit', 190), ('tongue', 185), ('hive', 179), ('foot', 175), ('diarrhea', 173), ('light', 166), ('urine', 164), ('vision', 163), ('rash', 161), ('mouth', 161), ('breath', 156), ('headache', 155), ('heartbeat', 154), ('urination', 152), ('blood', 151), ('head', 149), ('yellow', 146), ('cough', 131), ('low', 129), ('hand', 129), ('confusion', 124), ('appetite', 115), ('movement', 115), ('right', 112), ('change', 112), ('bruise', 111), ('short', 107), ('tiredness', 107), ('red', 106), ('weight', 106), ('body', 105), ('increase', 105), ('pas', 105), ('drowsiness', 105), ('difficult', 104), ('color', 103), ('serious', 100), ('back', 98), ('dark', 92), ('cold', 92), ('irregular', 91), ('chill', 91), ('difficulty', 91), ('little', 90), ('numbness', 90), ('bad', 90), ('jaundice', 88), ('burn', 83), ('high', 83), ('new', 82), ('itch', 82), ('joint', 80), ('stool', 78), ('pale', 77), ('redness', 77), ('swallow', 76), ('kidney', 75), ('bleeding', 75), ('constipation', 75), ('tire', 75), ('fast', 74), ('sleep', 74), ('slow', 73), ('tingly', 70), ('swollen', 69), ('easy', 69), ('speech', 68), ('worsen', 67), ('attention', 65), ('upper', 64), ('thought', 64), ('blister', 63), ('bleed', 63), ('level', 62), ('ear', 61), ('legs', 61), ('painful', 61), ('blur', 61)]

- **Extracting Common Side Effects for Pregnancy Category N:** Drugs with Pregnancy Category N (not classified) often had vague or severe-sounding side effects. We noted frequently occurring terms such as convulsion, hallucination, restlessness, and deadly. This step provided insight into the uncertainty and potential danger tied to medications without proper classification.

```
#viewing the 100 most common words from side effects column related to pregnancy category 4-> N
words = data[data.Pregnancy_Category==4].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_4_words = Counter()
for effects in words:
    cat_4_words.update(effects)
cat_4_words.most_common(100)
```

Output: [('skin', 763), ('throat', 641), ('pain', 631), ('swell', 604), ('breathe', 529), ('face', 462), ('stomach', 459), ('lip', 450), ('tongue', 434), ('hive', 406), ('bad', 358), ('feel', 333), ('serious', 307), ('mouth', 303), ('eye', 298), ('chest', 286), ('urine', 281), ('rash', 277), ('unusual', 275), ('loss', 269), ('difficult', 266), ('right', 255), ('fever', 247), ('headache', 242), ('problem', 242), ('blister', 240), ('nausea', 233), ('peel', 230), ('dizziness', 228), ('trouble', 228), ('red', 205), ('drowsiness',

200), ('weakness', 199), ('b', 196), ('itch', 194), ('confusion', 189), ('feeling', 186), ('heartbeat', 180), ('blood', 180), ('1', 178), ('even', 177), ('vomit', 176), ('fast', 176), ('constipation', 173), ('redness', 171), ('vision', 169), ('color', 167), ('light', 163), ('dry', 163), ('seizure', 162), ('yellow', 158), ('swollen', 156), ('drug', 155), ('muscle', 153), ('appetite', 146), ('dark', 146), ('0', 145), ('help', 142), ('blur', 135), ('upset', 133), ('nose', 126), ('diarrhea', 123), ('c', 122), ('bleed', 120), ('tightness', 120), ('ear', 120), ('restless', 119), ('stool', 118), ('mild', 117), ('movement', 116), ('effect', 115), ('swallow', 115), ('sleep', 114), ('convulsion', 111), ('wheeze', 110), ('caution', 109), ('talk', 109), ('high', 109), ('uneven', 109), ('hoarseness', 108), ('change', 107), ('upper', 107), ('allergic', 105), ('anxiety', 105), ('related', 104), ('warn', 103), ('though', 103), ('deadly', 103), ('cough', 102), ('hallucination', 101), ('slow', 99), ('body', 99), ('bruise', 99), ('difficulty', 99), ('jaundice', 96), ('irritation', 94), ('pas', 93), ('pass', 93), ('head', 92), ('stools', 91)]

- **Extracting Common Side Effects for Pregnancy Category X:** In the final category, Category X (contraindicated in pregnancy), the analysis surfaced alarming terms like stroke, clot, memory loss, depression, and breast tenderness. These side effects clearly align with the known high-risk nature of these drugs and validate their placement in the most dangerous category.

```
#viewing the 100 most common words from side effects column related to pregnancy category 5-> X
words = data[data.Pregnancy_Category==5].clean_side_effects.apply(lambda x: [word.lower() for word in x.split()])
cat_5_words = Counter()
for effects in words:
    cat_5_words.update(effects)
cat_5_words.most_common(100)
```

Output: [('pain', 594), ('problem', 284), ('swell', 250), ('skin', 247), ('stomach', 187), ('headache', 180), ('nausea', 168), ('loss', 162), ('blood', 162), ('weakness', 160), ('feel', 151), ('muscle', 145), ('vision', 138), ('throat', 138), ('chest', 135), ('sudden', 132), ('eye', 130), ('change', 124), ('urine', 110), ('face', 109), ('vomit', 104), ('breathe', 104), ('lip', 103), ('hive', 100), ('unusual', 96), ('tongue', 95), ('increase', 94), ('trouble', 94), ('yellow', 90), ('dark', 88), ('breath', 88), ('color', 84), ('liver', 83), ('difficult', 81), ('back', 80), ('numbness', 79), ('tiredness', 78), ('pressure', 77), ('body', 76), ('jaundice', 73), ('foot', 71), ('fever', 71), ('appetite', 69), ('dizziness', 69), ('mood', 68), ('right', 68), ('upper', 67), ('short', 67), ('cough', 65), ('breast', 65), ('tenderness', 62), ('shoulder', 61), ('legs', 59), ('serious', 59), ('itch', 58), ('warmth', 58), ('diarrhea', 57), ('heart', 54), ('neck', 53), ('feeling', 52), ('mouth', 51), ('sweat', 50), ('arm', 50), ('depression', 49), ('sore', 48), ('weight', 48), ('clot', 48), ('urination', 47), ('spread', 47), ('speech', 47), ('hand', 47), ('confusion', 47), ('vaginal', 46), ('sleep', 45), ('bleeding', 45), ('tire', 45), ('ear', 42), ('burn', 42), ('nose', 40), ('jaw', 40), ('rash', 40), ('low', 39), ('stool', 39), ('slur', 39), ('balance', 39), ('gain', 39), ('joint', 38), ('attack', 38), ('kidney', 38), ('stroke', 37), ('hair', 37), ('peel', 36), ('pound', 36), ('bad', 36), ('light', 35), ('behavior', 35), ('leg', 35), ('blur', 34), ('heartbeat', 34), ('memory', 34)]

Summary: These six steps collectively form a core part of the NLP-based exploratory analysis in our project. By isolating and comparing the most frequent side effect terms across each pregnancy category, we obtained a qualitative fingerprint of how symptoms shift from mild to severe as drug safety decreases. This analysis not only strengthened our feature understanding but also supported our downstream modeling by suggesting which side effect patterns are more associated with higher-risk drug categories. This type of pattern recognition is especially helpful when building interpretable and explainable healthcare prediction systems.

8. Model Train-Test Split:

- **Splitting Predictors and Target:** We began by defining two sets of input features:
X_text: The cleaned and preprocessed textual data from the Side_Effects column.
X_other: Structured categorical predictors including Drug_Classes, Rx_Otc,

Medical_Condition, Csa, and Alcohol_Interaction. The target variable (y) was the encoded Pregnancy_Category. These datasets were split into training and test sets using a stratified 80/20 split, ensuring that the class distribution was preserved across both sets.

```

Model training and evaluation

#defining predictors and target
#X_text: textual feature (cleaned side effects)
#X_other: structured features (other categorical predictors)
X_text = data.clean_side_effects
X_other = data[['Drug_Classes', 'Rx_Otc', 'Medical_Condition', 'Csa', 'Alcohol_Interaction']]
y = data.Pregnancy_Category
print(X_text.shape)
print(X_other.shape)
print(y.shape)
X_text_train, X_text_test, X_other_train, X_other_test, y_train, y_test = train_test_split(X_text, X_other, y, test_size=0.2, random_state=42, stratify=y)
print(X_text_train.shape)
print(X_other_train.shape)
print(X_text_test.shape)
print(X_other_test.shape)
print(y_train.shape)
print(y_test.shape)

(2501,)
(2501, 5)
(2501,)
(2072,)
(2072, 5)
(519, 5)
(519,)
(2072,)
(519,)

```

- **Vectorizing Textual Features:** To convert raw side effect descriptions into numerical features, we applied a TF-IDF (Term Frequency-Inverse Document Frequency) transformation using a pipeline with CountVectorizer followed by TfidfTransformer. This allowed the model to capture the importance of specific side effect terms while reducing the influence of commonly occurring but less informative words.
- **Encoding Structured Features:** The structured categorical predictors were transformed using one-hot encoding through a ColumnTransformer. This ensured that all categorical features were properly encoded into binary vectors, allowing machine learning algorithms to interpret them effectively. Unknown categories were safely ignored during transformation to maintain robustness.
- **Combining Textual and Structured Features:** Next, we combined the TF-IDF-transformed text data with the one-hot encoded structured features using hstack(). This formed a single, unified feature matrix for both training and testing, enabling the models to learn from both textual and categorical information simultaneously.
- **Feature Scaling:** For algorithms sensitive to feature magnitude (like logistic regression, SVM, and KNN), we converted the combined sparse matrices to dense format and then applied standard scaling. This step normalized the feature values to ensure that all predictors contributed proportionally to the model training process.

```

[] tfidf_vectorizer_pipeline = Pipeline([('vect', CountVectorizer(max_features=None, min_df=5, max_df=0.7)), ('tfidf', TfidfTransformer())])
# Fit the pipeline on training text data
tfidf_vectorizer_pipeline.fit(X_text_train)
# Transform the text data
X_text_train_tfidf = tfidf_vectorizer_pipeline.transform(X_text_train)
X_text_test_tfidf = tfidf_vectorizer_pipeline.transform(X_text_test)

categorical_cols = ['Drug_Classes', 'Rx_Otc', 'Medical_Condition', 'Csa', 'Alcohol_Interaction']
# Column transformer -> one-hot encodes categorical, passes numeric as-is
preprocessor = ColumnTransformer(transformers=[('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)], remainder='passthrough')
X_other_train_encoded = preprocessor.fit_transform(X_other_train)
X_other_test_encoded = preprocessor.transform(X_other_test)

# Combine TF-IDF text features and encoded structured features into a single feature matrix for model training and testing
X_train_combined = hstack([X_text_train_tfidf, X_other_train_encoded])
X_test_combined = hstack([X_text_test_tfidf, X_other_test_encoded])

# Scaling the features for logistic, SVM, and KNN models
X_train_dense = X_train_combined.toarray()
X_train_scaled = scaler.fit_transform(X_train_dense)
X_test_dense = X_test_combined.toarray()
X_test_scaled = scaler.transform(X_test_dense)

```

- **Extracting and Saving Feature Names:** After transforming both the textual and structured data, we extracted the names of all features used in the final model:
 - The TF-IDF vectorizer provided the textual feature names.
 - The OneHotEncoder within the column transformer yielded the names of encoded structured features.
 - These were combined and saved as column names for the model's training dataset (X_train_df) and the scaled dataset (X_train_scaled_df), ensuring transparency and traceability of the features used during modeling.

```

# Get feature names from the actual fitted vectorizer
vect_fitted = tfidf_vectorizer_pipeline.named_steps['vect']
tfidf_feature_names = vect_fitted.get_feature_names_out()
encoded_feature_names = preprocessor.get_feature_names_out()
all_feature_names = list(tfidf_feature_names) + list(encoded_feature_names)
X_train_df = pd.DataFrame(X_train_combined.toarray(), columns=all_feature_names)
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=all_feature_names)

```

- **Saving Feature Names to File:** To facilitate future reproducibility or model interpretation (e.g: during model re-deployment or updates), we saved the list of all scaled training features into a text file named feature_names.txt. This step ensures consistency between model training and deployment environments.
- **Defining Word Cloud Visualization for NMF Topics:** We created a custom function to visualize topics generated by the NMF model using word clouds. Each subplot shows the top words for a discovered topic, providing an intuitive way to interpret and communicate the themes present in patient-reported side effects.

```

# Save the names of all feature columns from the scaled training dataset to a text file. This helps in preserving the feature order and names for later
# interpretation or when reusing the model.
with open("feature_names.txt", "w") as f:
    for col in X_train_scaled_df.columns:
        f.write(col + "\n")

[74]: # This function visualizes topics generated by a fitted NMF (Non-negative Matrix Factorization) model using word clouds.
def plot_nmf_wordclouds(model, feature_names, n_top_words=30, n_cols=2):
    n_topics = model.n_components
    n_rows = (n_topics + n_cols - 1) // n_cols
    fig, axes = plt.subplots(n_rows, n_cols, figsize=(14, 4 * n_rows))
    for i, topic_weights in enumerate(model.components_):
        top_indices = topic_weights.argsort()[::-1][:n_top_words]
        top_words = [feature_names[j] for j in top_indices]
        row = i // n_cols
        col = i % n_cols
        ax = axes[row][col] if n_rows > 1 else axes[col]
        wc = wordcloud(background_color='white', max_words=n_top_words).generate_from_frequencies(top_words)
        ax.imshow(wc, interpolation='bilinear')
        ax.set_title(f"Topic #{i+1}", fontsize=14)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

```

- **Training NMF Model to Discover Latent Topics:** Using Non-negative Matrix Factorization (NMF) on the TF-IDF-transformed text data, we identified 20 hidden topics from side effects. The top 10 terms from each topic were displayed using a helper function. These topics revealed distinct patterns such as the ones shown in the output in the below picture.

```

# This block trains a Non-negative Matrix Factorization (NMF) model to discover hidden topics in the text data.
def display_topics(model, feature_names, n_top_words=10):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx + 1}: ", ", ".join([feature_names[i] for i in topic.argsort()[::-1][:n_top_words - 1]]))
    nmf = NMF(n_components=20, random_state=42)
    nmf.fit(X_text_train_tfidf)
    display_topics(nmf, tfidf_vectorizer_pipeline.get_feature_names_out())

Topic 1: short, breath, sudden, problem, blood, foot, feel, shoulder, jaw, speech
Topic 2: bad, deadly, though, warn, related, drug, swollen, caution, help, effect
Topic 3: skin, loss, urine, spot, decrease, discomfort, unusual, eye, red, eyelid
Topic 4: skin, irritation, sting, treat, burn, redness, itch, dryness, peel, mild
Topic 5: dry, drowsiness, constipation, difficulty, mouth, urination, excite, restless, blur, child
Topic 6: low, feel, flap, gain, hollow, entire, heartbeat, leg, cramp, flutter
Topic 7: thought, behavior, agitate, irritable, hostile, suicide, hurt, mentally, physically, hyperactive
Topic 8: breathing, slow, debilitate, vomit, sigh, shallow, nausea, long, rate, adult
Topic 9: skin, right, past, case, fatal, stools, eye, jaundice, take, dark
Topic 10: skin, increase, hair, change, thin, treat, absorb, heal, wound, torso
Topic 11: head, feeling, light, pas, heartbeat, chest, tingly, feel, high, irregular
Topic 12: rectal, stomach, bleed, stool, bowel, vomit, difficult, look, ground, day
Topic 13: uneven, seizure, urinating, convulsion, dangerously, tremor, pressure, headache, ear, anxiety
Topic 14: infection, neck, cold, problem, trouble, immune, thyroid, speak, enlarge, week
Topic 15: diarrhea, watery, dose, bloody, month, last, stomach, even, bruise, fever
Topic 16: health, care, attention, notice, healthcare, unwanted, need, patient, advice, report
Topic 17: sore, movement, muscle, fever, count, cell, chill, mouth, cough, high
Topic 18: less, apply, serious, sting, irritation, difficult, burn, redness, difficulty, blister
Topic 19: child, problem, rate, heart, hallucination, twitch, toe, loss, real, circulation
Topic 20: last, thought, behavior, memory, long, longer, drowsiness, accidental, fall, ring

```

- [illegible]

Interpretation: After applying topic modeling to the cleaned side effects text, we discovered 20 distinct themes that reflect real-world drug reactions experienced by patients. These themes, revealed through both keyword analysis and visual word clouds, provide a deeper understanding of the nature and clusters of side effects.

- **Topic 1: Respiratory and Cardiac Emergencies** Terms like short, breath, sudden, and speech suggest urgent symptoms related to breathing difficulty or cardiac issues. This cluster indicates potentially life-threatening conditions requiring immediate medical attention.
- **Topic 2: Drug Warnings and Toxicity Alerts** Words such as deadly, warn, caution, and related reflect strong cautionary signals and severe adverse drug reactions, possibly black-box warnings or FDA alerts.

- **Topic 3: Sensory and Dermatological Disruptions** This topic highlights conditions like skin loss, urine spot, and eye discomfort, indicating general deterioration of surface-level bodily functions and discomfort.
- **Topic 4: Topical Skin Irritation** Dominated by words like itch, sting, redness, and burn, this topic represents common localized skin reactions from ointments, patches, or allergic responses.
- **Topic 5: Anticholinergic or Sedative Effects** With terms like drowsiness, dry mouth, constipation, and blur, this topic likely reflects side effects related to drugs with sedative or anticholinergic properties, often affecting older adults.
- **Topic 6: Muscle Weakness and Circulation** Here, words like limp, cramp, flutter, and gain point to muscular fatigue and circulatory effects, possibly associated with electrolyte imbalance or cardiovascular medications.
- **Topic 7: Psychiatric and Behavioral Concerns** This emotionally intense cluster, suicide, agitate, hostile, thought, behavior, suggests psychiatric reactions such as aggression, mood swings, or suicidal ideation.
- **Topic 8: Respiratory Suppression and Nausea** Terms like shallow, sigh, vomit, and breathing suggest depressive effects on respiration combined with GI upset, often seen with opioids or sedatives.
- **Topic 9: Liver Dysfunction and Fatal Reactions** With mentions of jaundice, dark stools, and fatal, this topic reflects symptoms of liver failure or advanced toxicity, a major signal for immediate discontinuation.
- **Topic 10: Skin Growth and Healing Changes** Words like hair, thin, absorb, torso, and heal show possible effects of topical steroids or hormone-based treatments that affect skin and hair patterns.
- **Topic 11: Dizziness and Cardiac Fluctuations** This group, featuring lightheadedness, tingly, irregular heartbeat, and chest, suggests circulatory instability or nervous system responses.
- **Topic 12: Gastrointestinal Bleeding and Rectal Symptoms** With strong GI indicators like rectal bleed, stomach, bowel, and vomit, this topic aligns with ulceration, inflammation, or internal bleeding side effects.
- **Topic 13: Neurological Emergencies and Seizures** This cluster includes intense terms like seizure, tremor, convulsion, urinating, and dangerously, reflecting acute neurologic toxicity and autonomic dysregulation.
- **Topic 14: Infections and Immune Dysfunction** Symptoms like infection, thyroid, immune, cold, and enlarge reflect autoimmune suppression, glandular swelling, or post-infection complications.
- **Topic 15: Diarrhea and Prolonged GI Discomfort** With diarrhea, bloody stools, dose, and fever, this topic suggests persistent gastrointestinal disturbances, possibly from antibiotic or chemotherapy drugs.
- **Topic 16: Patient Experience and Monitoring Needs** Words such as attention, healthcare, report, and notice shift away from symptoms and focus on patient awareness, counseling, and follow-up communication.
- **Topic 17: Flu-like and Hematological Reactions** Fever, chill, sore, mouth, and cell count show up here, reflecting systemic immune responses similar to viral infection or blood cell suppression.

- **Topic 18: Severe Topical Reactions and Blistering** Terms like burn, sting, blister, redness, and apply suggest adverse reactions to topical formulations potentially contact dermatitis or chemical burns.
- **Topic 19: Pediatric Neurological Symptoms** Featuring child, hallucination, toe, and heart rate, this unique cluster likely represents pediatric-focused side effects including motor or cognitive disruptions.
- **Topic 20: Cognitive Decline and Sedation Risks** With terms like memory, drowsiness, fall, and accidental, this topic reflects cognitive and motor impairment often associated with sedating drugs in elderly populations.
- To look at the most frequently associated pregnancy category for each of the 20 latent topics extracted via Non-negative Matrix Factorization (NMF) from the side-effect narratives.

```
#the top pregnancy category associated with each of the 20 NMF topics
topic_distributions = nmf.transform(X_text_train_tfidf)
topic_df = pd.DataFrame(topic_distributions, columns=[f"topic_{i}" for i in range(nmf.n_components)])
topic_df["Pregnancy_Category"] = y_train.reset_index(drop=True)
topic_df["Dominant_Topic"] = topic_df.drop(columns=["Pregnancy_Category"]).idxmax(axis=1)
top_category_per_topic = topic_df.groupby("Dominant_Topic")["Pregnancy_Category"].agg(lambda x: x.value_counts().idxmax())
print(top_category_per_topic)
```

| Dominant_Topic | Pregnancy_Category |
|----------------|--------------------|
| Topic_0 | 2 |
| Topic_1 | 4 |
| Topic_10 | 2 |
| Topic_11 | 2 |
| Topic_12 | 4 |
| Topic_13 | 2 |
| Topic_14 | 1 |
| Topic_15 | 2 |
| Topic_16 | 2 |
| Topic_17 | 4 |
| Topic_18 | 2 |
| Topic_19 | 2 |
| Topic_2 | 2 |
| Topic_3 | 2 |
| Topic_4 | 2 |
| Topic_5 | 2 |
| Topic_6 | 2 |
| Topic_7 | 2 |
| Topic_8 | 2 |
| Topic_9 | 2 |

Name: Pregnancy_Category, dtype: int64

Interpretation: The majority of the topics (e.g: Topic_0, Topic_2, Topic_3, Topic_19) are most commonly associated with Pregnancy Category 2, which may correspond to a moderate-risk classification (e.g: Category C). A few topics (Topic_1, Topic_12, Topic_17) show stronger associations with Pregnancy Category 4, indicating that certain clusters of side effects are more common in not classified categories of drugs. Topic_14 stands out as being most associated with Pregnancy Category 1, which may indicate a safer classification (like Category B). This suggests that the latent topics learned from side-effect text data do capture signals that differentiates between pregnancy safety categories particularly distinguishing moderate vs. lower-risk drugs.

9. Baseline models training and evaluation:

- **Defining a Reusable Model Training Function:** We defined a `train_and_evaluate()` utility function to standardize model training and evaluation. This function:
 - Trains a given classification model on training data.
 - Makes predictions on both train and test sets.
 - Outputs accuracy, a classification report, and a confusion matrix. This modular design allowed easy comparison of different algorithms under a consistent evaluation framework.


```

Baseline Models training
1. Logistic Regression

def train_and_evaluate(model, X_train, X_test, y_train, y_test, model_name="model"):
    """
    Trains and evaluates a classification model.
    Parameters:
    - model: Any sklearn-compatible classifier
    - X_train, X_test: Features for training and testing (can be scaled or combined)
    - y_train, y_test: Target variables
    - model_name: Optional name to label the model in output
    Returns:
    - trained model
    """
    print(f"--- {model_name} ---")
    model.fit(X_train, y_train)
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)
    print("Train Accuracy:", round(accuracy_score(y_train, y_train_pred), 4))
    print("Test Accuracy:", round(accuracy_score(y_test, y_test_pred), 4))
    print("Classification Report:\n", classification_report(y_test, y_test_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
    return model

```

- Baseline Logistic Regression Model:** A Logistic Regression model was trained using L2 regularization (penalty='l2') and a smaller regularization strength (C=0.05) to prevent overfitting. class_weight='balanced' handled class imbalance by adjusting the penalty inversely proportional to class frequencies. This served as our baseline linear classifier to understand how well linear boundaries could distinguish drug categories.

```

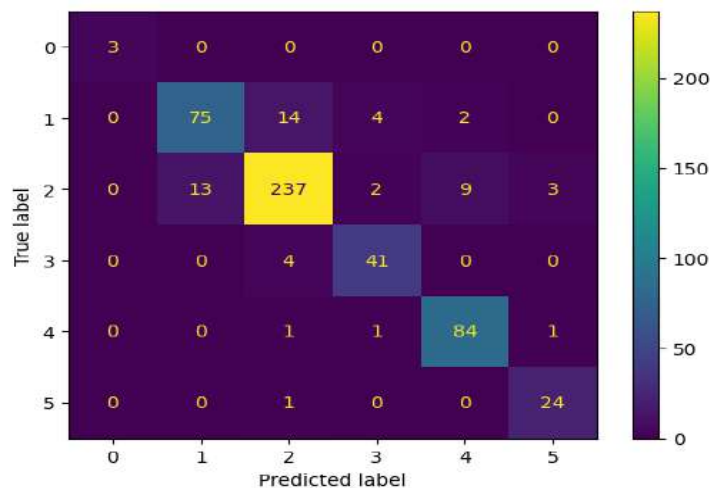
#Logistic Regression
model = LogisticRegression(penalty='l2', C=0.05, solver='liblinear', random_state=42, class_weight='balanced')
trained_lr = train_and_evaluate(model, X_train_scaled, X_test_scaled, y_train, y_test, "Logistic Regression")

--- Logistic Regression ---
Train Accuracy: 0.987
Test Accuracy: 0.894
Classification Report:
      precision    recall  f1-score   support

 0      1.00      1.00      1.00         3
 1      0.85      0.79      0.82        95
 2      0.92      0.90      0.91       264
 3      0.85      0.91      0.88        45
 4      0.88      0.97      0.92        87
 5      0.86      0.96      0.91        25

 accuracy      0.89
 macro avg     0.89
 weighted avg   0.89
Confusion Matrix:
[[ 3  0  0  0  0  0]
 [ 0 75 14  4  2  0]
 [ 0 13 237  2  9  3]
 [ 0  0  4 41  0  0]
 [ 0  0  1  1 84  1]
 [ 0  0  1  0  0 24]]

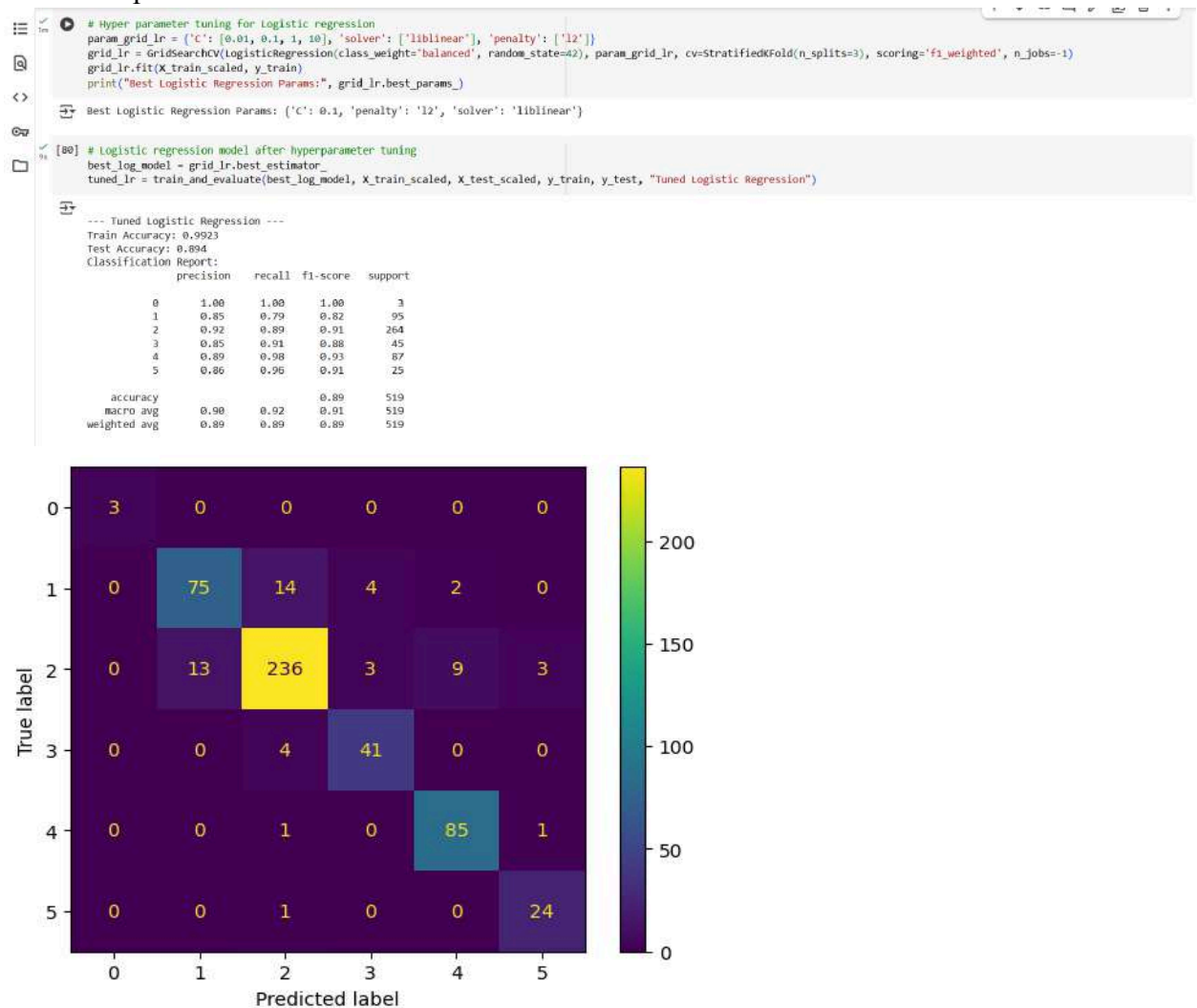
```



Interpretation: The Logistic Regression model achieved a strong test accuracy of approximately 89.4%, though a noticeable gap from the 98.7% training accuracy indicates some degree of overfitting. While the model performs well on the majority classes (Category 2), its notable strength lies in its excellent recall for categories 4 (unknown) and 5 (high-risk) (0.97 and 0.96 respectively), ensuring it rarely misses these critical cases. The confusion matrix visually confirms this, showing very few false negatives for these crucial categories. The precision for Category 4 is also strong at 0.88, indicating few false positives. The perfect scores for very small classes (0) should be interpreted with caution. Overall, the model provides a solid foundation for

drug safety prediction, performing robustly across most categories, with its ability to reliably identify high-risk drugs.

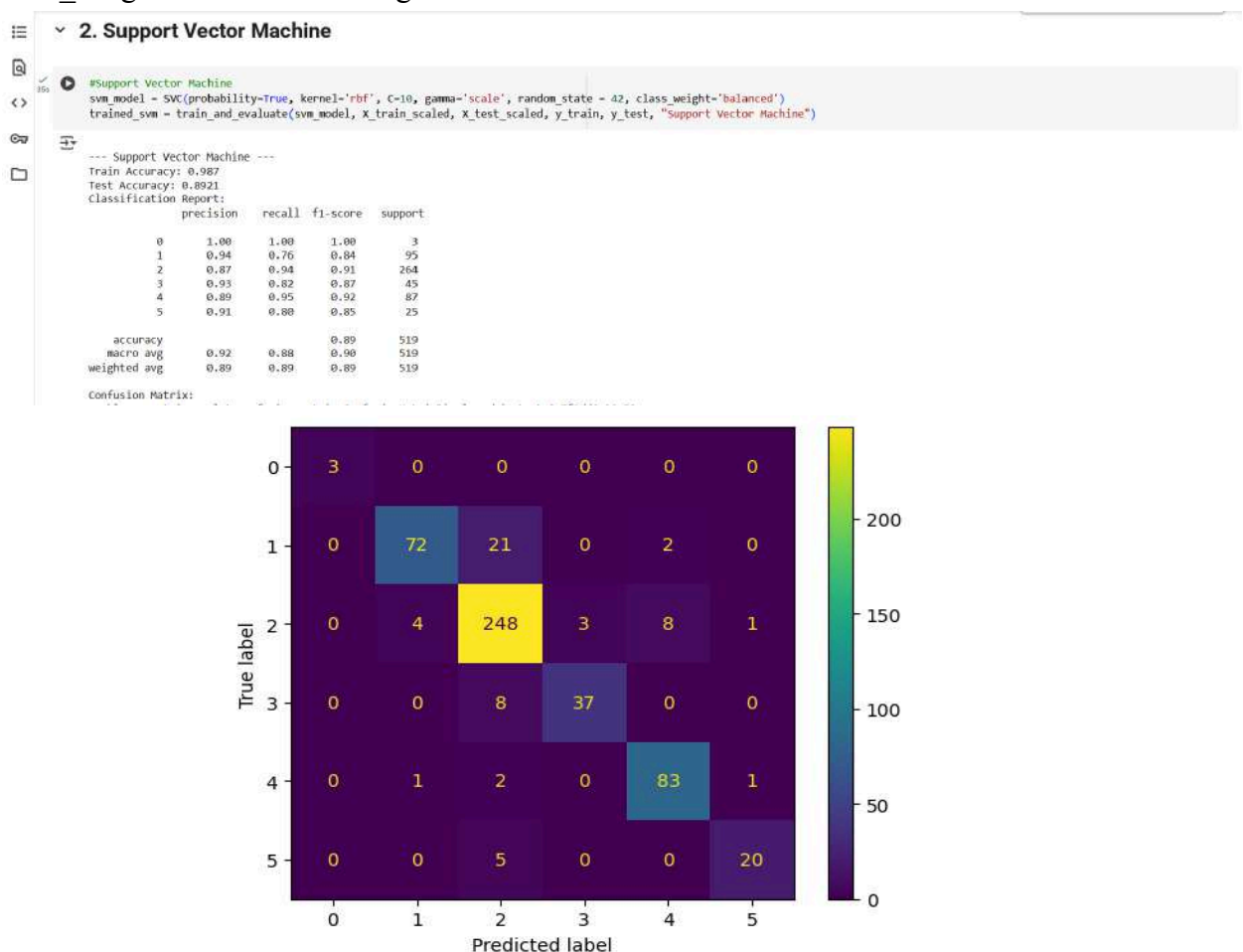
- **Hyperparameter Tuning for Logistic Regression:** To optimize performance, we conducted Grid Search Cross-Validation using GridSearchCV, testing combinations of regularization strengths (C) and solvers. The scoring metric was weighted F1-score, appropriate for multiclass imbalanced classification. This helped identify the best-performing hyperparameters before finalizing the model.
- **Logistic Regression with Tuned Parameters:** We retrained logistic regression using the best parameters obtained from grid search (C=0.1, solver='liblinear'). This ensured the final linear model leveraged optimal regularization, improving both generalization and class-wise precision/recall over the baseline.



Interpretation: After hyperparameter tuning, the logistic regression model's test accuracy remained at 89.4%, showing no degradation in performance. However, train accuracy increased from 98.7% to 99.2%, suggesting a better model fit without overfitting. The overall F1 scores (macro and weighted) stayed consistent at 0.91 and 0.89, respectively. This indicates the model continues to perform reliably across all

classes. At the class level, recall for class 4 improved from 0.97 to 0.98, showing better sensitivity. A slight dip in recall was observed for class 2 (0.90 to 0.89), but this had negligible impact. The confusion matrix showed nearly identical predictions before and after tuning. Only minor differences, such as one more correct prediction for class 4, were observed. No new misclassification patterns emerged post-tuning. This suggests that the tuning enhanced the model's internal confidence. Despite no major metric shifts, the model became slightly more balanced. It performed particularly well on high-frequency classes without sacrificing others. The model generalizes well and remains stable under tuning.

- **Support Vector Machine (SVM) Baseline:** An initial Support Vector Classifier (SVC) was trained using an RBF kernel. SVMs are known for handling high-dimensional spaces well, making them suitable for our TF-IDF and one-hot encoded feature set. We included `class_weight='balanced'` to mitigate bias due to class imbalance.



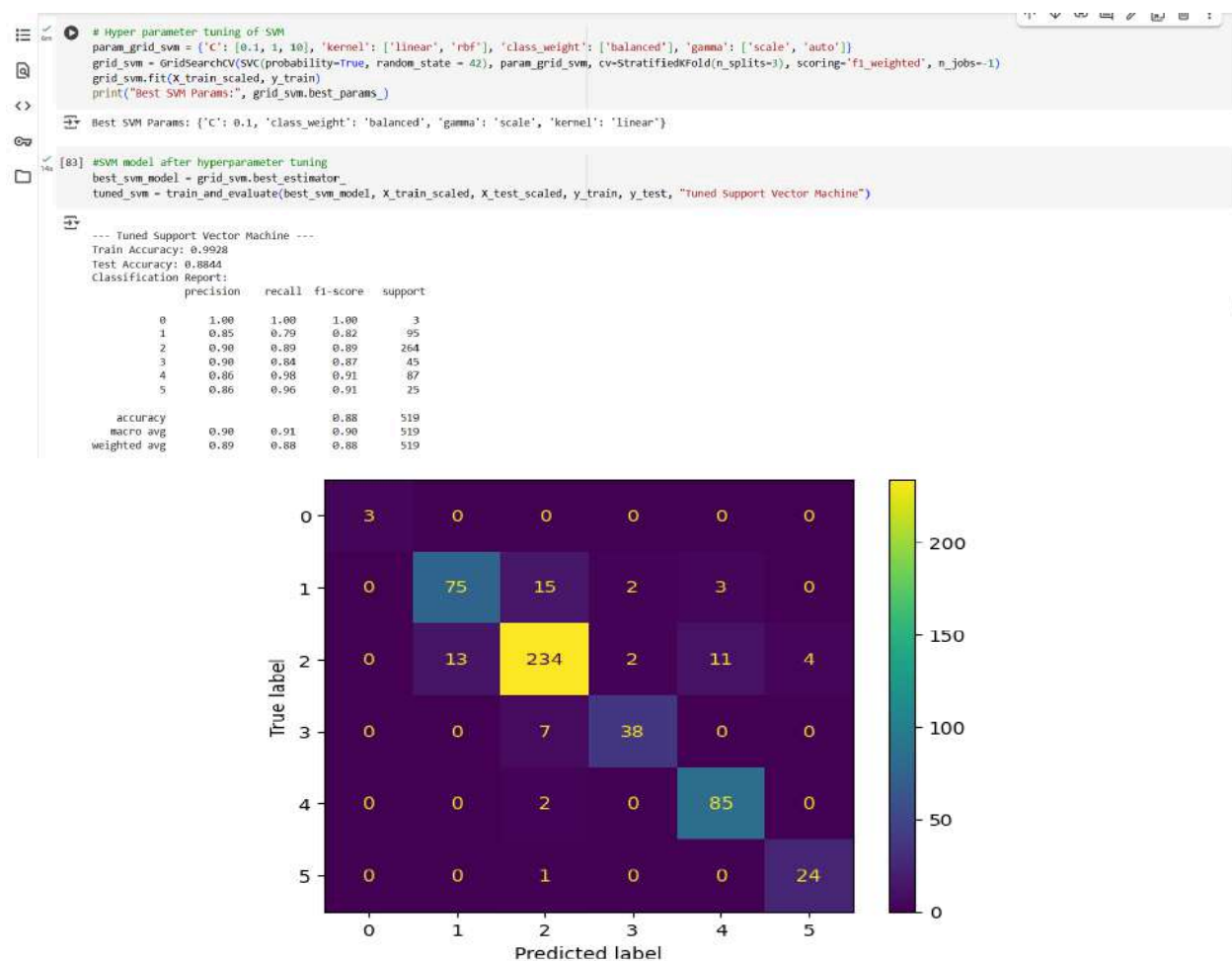
Interpretation: The Support Vector Machine (SVM) classifies pregnancy drug-safety categories with a test accuracy of 0.892 and a weighted F1 of 0.89, indicating it correctly labels nearly 9 in 10 drugs while keeping precision and recall balanced across classes. The confusion matrix shows strong performance on the largest class (class 2: 248/264 correct) and on class 4 (83/87 correct). Rare class 0 is perfectly predicted (3/3), while class 5 is mostly right (20/25) but a few shift to class 2. Most errors cluster between

clinically similar categories: class 1 loses 21 cases to class 2, and class 3 sends 8 to class 2, suggesting the feature patterns for those groups overlap. Overall, the SVM is a high-performing, well-balanced model; its main weakness is confusion between adjacent categories (especially into class 2).

- **Hyperparameter Tuning for SVM:** We again used GridSearchCV, testing combinations of:
 - Kernel types (linear, rbf)
 - Penalty term, C
 - Kernel coefficient (gamma)

This step was crucial to identify whether linear or nonlinear boundaries better separated the drug categories and to avoid overfitting.

- **SVM with Tuned Parameters:** We retrained SVM using the best-found parameters (kernel='linear', C=0.1, gamma='scale'). The linear kernel outperformed RBF, suggesting the classes were linearly separable in the transformed feature space. The model was re-evaluated using the same train_and_evaluate() function.



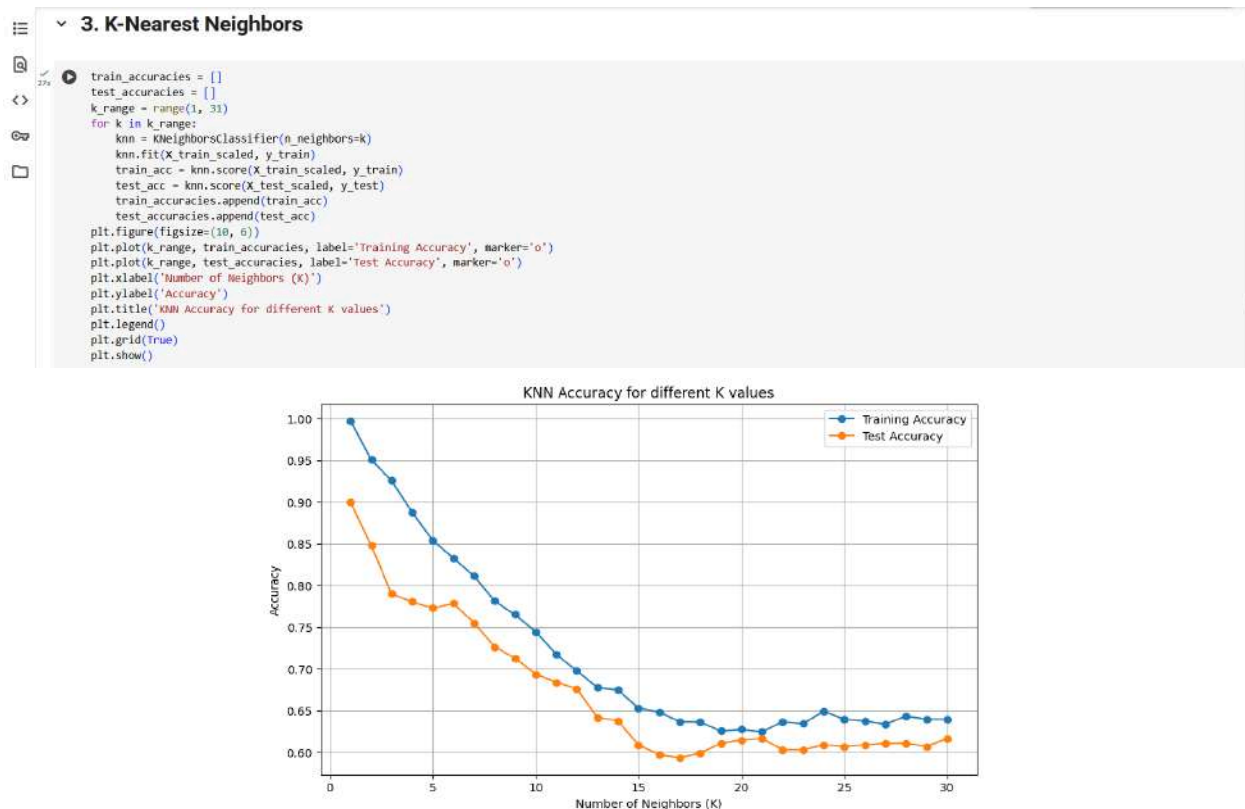
Interpretation: After hyperparameter tuning, the support vector machine (SVM) model experienced a slight decrease in test accuracy, dropping from 89.2% to 88.4%. However, its train

accuracy increased from 98.7% to 99.3%, suggesting a tighter fit to the training data. While this could raise concerns about overfitting, the performance remained stable across most classes, indicating that generalization was still well preserved. Looking at the classification reports, the macro and weighted F1 scores decreased marginally (from 0.90 to 0.88), with most of the impact observed in class 2 and class 1. For example, class 2's F1 score declined from 0.91 to 0.89, and class 1's precision dropped from 0.94 to 0.85. On the other hand, class 5 improved its recall from 0.80 to 0.96, and class 4's recall also increased from 0.95 to 0.98, suggesting better detection of those classes after tuning.

The confusion matrix also reflects these shifts. The tuned model made more accurate predictions for class 5 (24 vs. 20) and class 4 (85 vs. 83), but slightly underperformed for class 1 and class 2. Despite these differences, the overall prediction pattern remained consistent, with no dramatic misclassification emerging. In summary, tuning the SVM led to a model that slightly overfits but performs better for certain important classes. While the overall accuracy and F1 metrics saw a small dip, the model became more sensitive to underrepresented or complex classes like class 5, making it a more balanced choice in contexts where those outcomes matter.

- **Visualizing K-Nearest Neighbors (KNN) Performance for Different K Values:** We ran KNN models with k ranging from 1 to 30, plotting train and test accuracies to visually assess:
 - Bias-variance tradeoff
 - Underfitting vs. overfitting behavior

This plot informed us about the optimal number of neighbors (k) that balance complexity and generalization for our dataset.



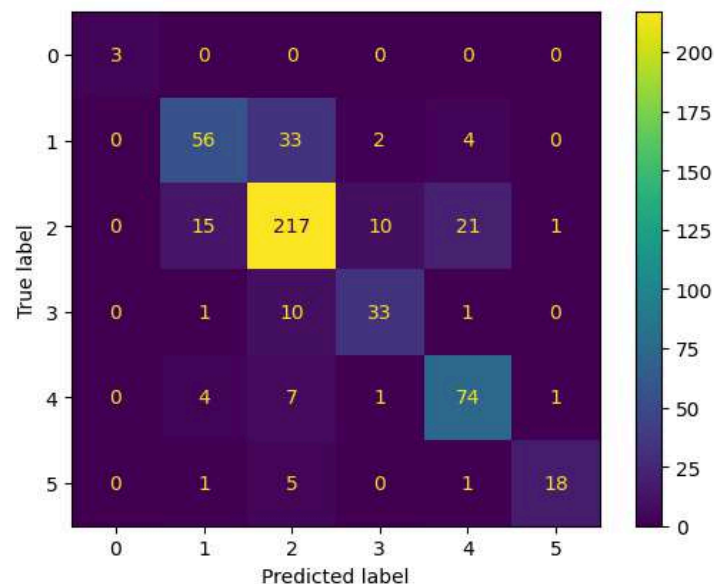
Interpretation: This KNN plot illustrates how the number of neighbors (K) affects both training and test accuracy. At very low values of K (e.g., K=1 to K=3), the training accuracy is extremely high, almost perfect, indicating that the model is overfitting the training data. However, the test accuracy is slightly lower, showing the model is not generalizing well to unseen data.

As K increases, both training and test accuracy decline. This is expected, as larger K values make the model more generalized by averaging across more neighbors, which reduces variance but increases bias. Around K=5 to K=7, the test accuracy is relatively stable and balanced against the training performance, indicating a better generalization sweet spot.

Beyond K=10, test accuracy continues to drop and fluctuates around 60-65%, while training accuracy also declines. The gap between training and test accuracy becomes narrower, suggesting the model is underfitting, failing to capture complex patterns in the data.

Overall, this plot highlights the classic bias-variance tradeoff in KNN: smaller K values overfit, while larger K values underfit. A mid-range value ($K \approx 5-7$) may provide the optimal balance between model complexity and generalization.

- **Baseline K-Nearest Neighbors (KNN) Model:** We trained a default KNN model with $k=5$, a commonly used starting point. KNN is a simple, non-parametric algorithm that makes predictions based on proximity in feature space. It served as a contrasting method to linear (Logistic Regression) and margin-based (SVM) models.



Interpretation: The K-Nearest Neighbors (KNN) model achieved a train accuracy of 85.4% and a significantly lower test accuracy of 77.3%, indicating some degree of underfitting. This suggests that the model struggles to capture the complexity of the data, potentially due to its sensitivity to noise or limitations in distance-based decision-making for higher-dimensional feature spaces.

In the classification report, we observe that class-level performance varies. Class 0 was perfectly classified with a precision and recall of 1.00, although this is based on only 3 samples. For the more prevalent classes, such as class 2 and class 4, the model performed reasonably well with F1 scores of 0.81 and 0.79, respectively. However, class 1 shows weakness, with a recall of only 0.59, indicating that the model frequently misclassified these instances. Similarly, class 5 had high precision (0.90) but lower recall (0.72), suggesting the model is conservative in predicting this class, likely leading to missed detections. The confusion matrix reflects these observations. Class 1 had a high number of misclassifications, with many samples being incorrectly assigned to classes 2 and 4. Class 2 also shows a spread of misclassifications, particularly toward class 4. These overlaps suggest that the KNN model may not be adequately separating nearby classes in the feature space.

Overall, the KNN model demonstrates reasonable but inconsistent performance, particularly struggling with overlapping or imbalanced class distributions. While it handles some classes well, its lower generalization ability and high variance across class-level metrics suggest it may not be the most reliable choice for this task without further optimization or feature engineering.

- **Hyperparameter Tuning for KNN:** We conducted grid search to optimize:
 - n_neighbors (number of neighbors)
 - weights (uniform vs. distance)
 - metric (Euclidean vs. Manhattan distance)

This was essential since KNN performance is highly sensitive to distance metrics and neighbor counts. The best-performing combination was: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}

```
#Hyper parameter tuning of KNN
param_grid_knn = {'n_neighbors': [3, 5, 7, 9], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=StratifiedKFold(n_splits=3), scoring='f1_weighted', n_jobs=-1)
grid_knn.fit(X_train_scaled, y_train)
print("Best KNN Params:", grid_knn.best_params_)

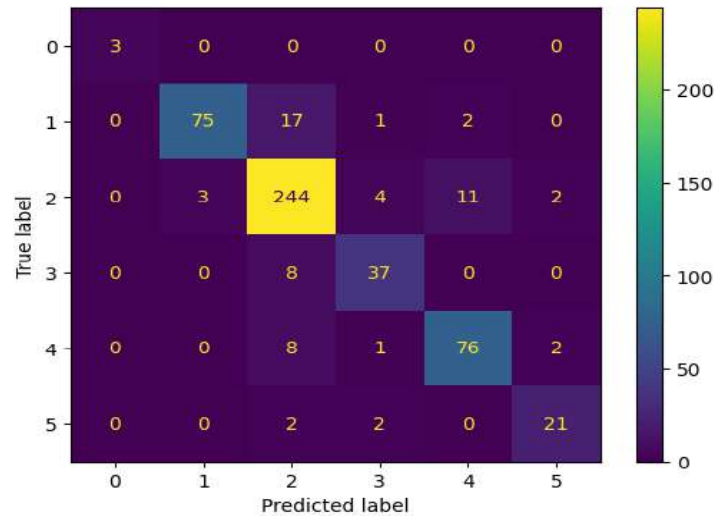
Best KNN Params: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}

#KNN model after hyperparameter tuning
best_knn_model = grid_knn.best_estimator_
tuned_knn = train_and_evaluate(best_knn_model, X_train_scaled, X_test_scaled, y_train, y_test, "Tuned K-Nearest Neighbors")

--- Tuned K-Nearest Neighbors ---
Train Accuracy: 0.8546
Test Accuracy: 0.7736
Classification Report:

```

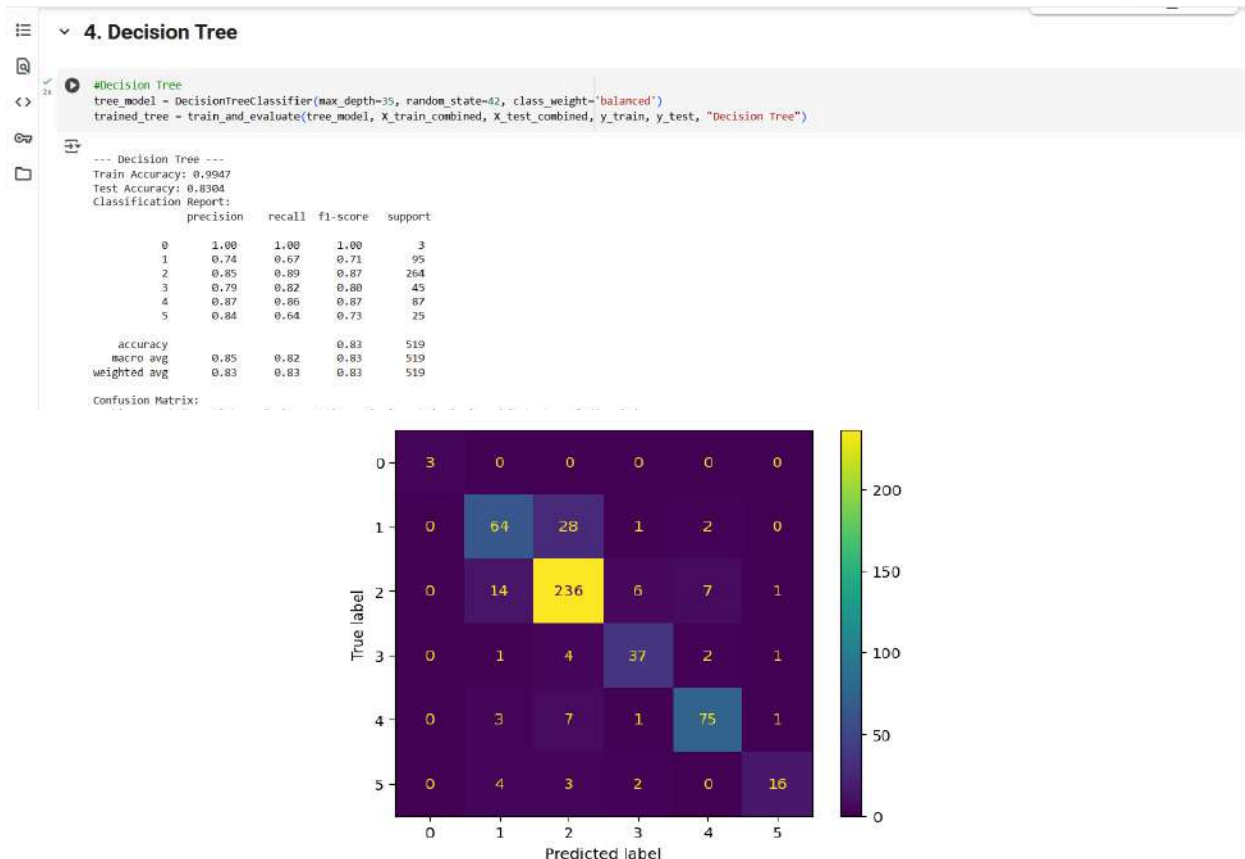
| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 3 |
| 1 | 0.06 | 0.79 | 0.17 | 95 |
| 2 | 0.87 | 0.92 | 0.90 | 264 |
| 3 | 0.82 | 0.82 | 0.82 | 85 |
| 4 | 0.85 | 0.87 | 0.86 | 87 |
| 5 | 0.84 | 0.84 | 0.84 | 25 |
| accuracy | | | 0.88 | 519 |
| macro avg | 0.80 | 0.87 | 0.88 | 519 |
| weighted avg | 0.88 | 0.88 | 0.88 | 519 |



Interpretation: After hyperparameter tuning, the performance of the K-Nearest Neighbors (KNN) model showed a notable improvement. The test accuracy rose from 77.3% to 87.9%, and train accuracy jumped significantly from 85.4% to 99.8%, indicating a much better fit to the training data, though this also raises a mild concern about overfitting. The weighted F1-score increased from 0.77 to 0.88, reflecting enhanced overall predictive performance across all classes. Notably, the recall for class 1 improved from 0.59 to 0.79, showing that the tuned model is now better at correctly identifying samples of this class. Similarly, class 2 and class 5 also saw gains in recall, suggesting more consistent sensitivity across categories. While precision and recall became more balanced post-tuning, macro and weighted averages both improved, demonstrating better generalization. The confusion matrix revealed fewer misclassifications for classes 1 and 5 compared to the original model, confirming the tuning effectively addressed class-level weaknesses. In summary, tuning significantly boosted the KNN model's accuracy, class-wise balance, and general robustness, making it a stronger performer across the board.

- Training a Baseline Decision Tree Classifier:** We started by building a Decision Tree classifier using the full set of combined structured and textual features.
 - max_depth=35 was chosen to allow the tree to grow deep enough to capture complex decision boundaries.
 - class_weight='balanced' was applied to account for the imbalance in pregnancy category labels.

This model offered interpretability and served as a strong baseline for tree-based methods.



Interpretation: The original Decision Tree model demonstrates strong performance with a test accuracy of 83.0% and a notably high training accuracy of 99.5%, which may suggest slight overfitting. Class-level evaluation shows reliable predictions for high-support classes, especially class 2 (F1-score: 0.87) and class 4 (F1-score: 0.87), indicating good balance between precision and recall. However, class 5 shows a drop in recall (0.64), implying the model occasionally struggles to identify those correctly. Precision and recall for class 1 are modest (0.74 and 0.67, respectively), with several samples misclassified into class 2, as seen in the confusion matrix. Despite this, macro and weighted averages remain strong at 0.83, reflecting overall stable performance. The confusion matrix also suggests relatively low misclassification across most classes. In summary, while the Decision Tree performs consistently and identifies majority classes well, it could benefit from tuning to further address minor class misclassification and potential overfitting.

- **Visualizing the Decision Tree:** To gain insight into how the Decision Tree splits data:
 - We plotted the tree structure up to a depth of 10, allowing us to visualize the top-level decision paths.
 - The visualization revealed the most influential features and helped validate that the model's logic aligned with domain understanding.

```
#Decision tree plot
plt.figure(figsize=(20, 15))
plot_tree(tree_model, filled=True, rounded=True, fontsize=10, max_depth=10, feature_names=None, class_names=[str(cls) for cls in tree_model.classes_])
plt.title("Decision Tree Visualization (Top 2 Levels)")
plt.show()
```

-
- Decision Tree Visualization (Top 2 Levels)
- Root Node: $x[1345] \leq 0.5$
 gini = 0.833
 samples = 2072
 value = [345.333, 345.333, 345.333, 345.333, 345.333, 345.333]
 class = 3
- Left Branch: $x[136] \leq 0.119$
 gini = 0.597
 samples = 689
 value = [0.0, 94.264, 97.809, 13.575, 279.461, 6.976]
 class = 4
- Right Branch: $x[167] \leq 0.093$
 gini = 0.768
 samples = 1370
 value = [0.0, 251.069, 248.024, 353.758, 65.873, 338.357]
 class = 5
- Further splits and nodes include:
- $x[813] \leq 0.0$ (gini = 0.539, samples = 39, value = [0.0, 49.851, 91.412, 21.314], class = 4)
 - $x[100] \leq 0.047$ (gini = 0.596, samples = 92, value = [0.0, 44.413, 5.880, 7.717, 20.96, 0.0], class = 1)
 - $x[1404] \leq 0.5$ (gini = 0.769, samples = 1323, value = [0.0, 245.632, 245.731, 333.758, 65.873, 2], class = 3)
 - $x[571] \leq 0.032$ (gini = 0.116, samples = 47, value = [0.0, 5.438, 2.293, 0.0, 0.0, 116.599], class = 5)
 - $x[17]$ (gini = 0.0, samples = 35, value = [0.0, 35.349], class = 4)
 - $x[430] \leq 0.06$ (gini = 0.039, samples = 17, value = [0.0, 0.0, 0.0, 0.0, 13.959, 0.0], class = 4)
 - $x[1389] \leq 0.5$ (gini = 0.778, samples = 1193, value = [0.0, 234.754, 225.366, 208.358, 65.873, 215], class = 1)
 - $x[6]$ (gini = 0.0, samples = 6, value = [0.0, 10.0], class = 1)
 - $x[1402]$ (gini = 0.0, samples = 1, value = [0.0, 33.536, 76], class = 4)
 - $x[25] \leq 0$ (gini = 0.4, samples = 1, value = [0.0, 24.472, 49.46], class = 4)
 - $x[126] \leq 0.035$ (gini = 0.778, samples = 1035, value = [0.0, 153.179, 213.940, 198.711, 65.873], class = 2)
 - $x[106] \leq 0.095$ (gini = 0.775, samples = 1165, value = [0.0, 232.941, 276.4, 206.428], class = 1)
 - $x[143]$ (gini = 0.0, samples = 1, value = [0.0, 1.813], class = 1)
 - $x[194]$ (gini = 0.0, samples = 1, value = [0.0, 9.064], class = 1)
 - $x[501] \leq 0$ (gini = 0.7, samples = 16, value = [0.0, 120.549, 176.526], class = 2)
 - $x[511]$ (gini = 0.0, samples = 2, value = [0.0, 32.63, 37.02], class = 1)
 - $x[8]$ (gini = 0.0, samples = 1, value = [0.0, 120.549], class = 2)
 - $x[111]$ (gini = 0.0, samples = 1, value = [0.0, 0.0, 1.668, 0.0, 0.0, 0.0], class = 2)
 - $x[516] \leq 0.067$ (gini = 0.0, samples = 4, value = [0.0, 0.0, 1.311, 0.0, 0.0, 0.0], class = 2)
 - $x[516]$ (gini = 0.0, samples = 2, value = [0.0, 0.0, 0.655, 0.0, 0.0, 0.0], class = 2)

This step was crucial to avoid overfitting or underfitting, leading to a better generalized model.

- **Final Tuned Decision Tree Evaluation:** Using the best parameters from tuning, we retrained and re-evaluated the tuned Decision Tree:
 - This ensured the final tree was both efficient and accurate.
 - Evaluation metrics confirmed performance improvements over the baseline model.

The tuned version provided an interpretable model with improved predictive power.

```
[90] #Hyper parameter tuning of Decision tree
param_grid_tree = {'max_depth': [10, 20, 30, 40], 'min_samples_split': [2, 5, 10], 'class_weight': ['balanced'], 'criterion': ['gini', 'entropy']}
grid_tree = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_tree, cv=StratifiedFold(n_splits=3), scoring='f1_weighted', n_jobs=-1)
grid_tree.fit(X_train_combined, y_train)
print("Best Decision Tree Params:", grid_tree.best_params_)

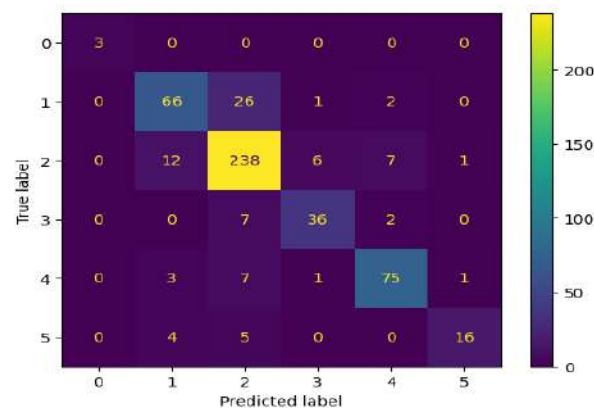
Best Decision Tree Params: {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 40, 'min_samples_split': 2}

#Decision tree after hyperparameter tuning
best_tree_model = grid_tree.best_estimator_
tuned_tree = train_and_evaluate(best_tree_model, X_train_combined, X_test_combined, y_train, y_test, "Tuned Decision Tree")

--- Tuned Decision Tree ---
Train Accuracy: 0.9961
Test Accuracy: 0.8362
Classification Report:
      precision    recall  f1-score   support

0         1.00      1.00      1.00         3
1         0.78      0.69      0.73        95
2         0.84      0.90      0.87       264
3         0.82      0.80      0.81        45
4         0.87      0.86      0.87        87
5         0.89      0.64      0.74        25

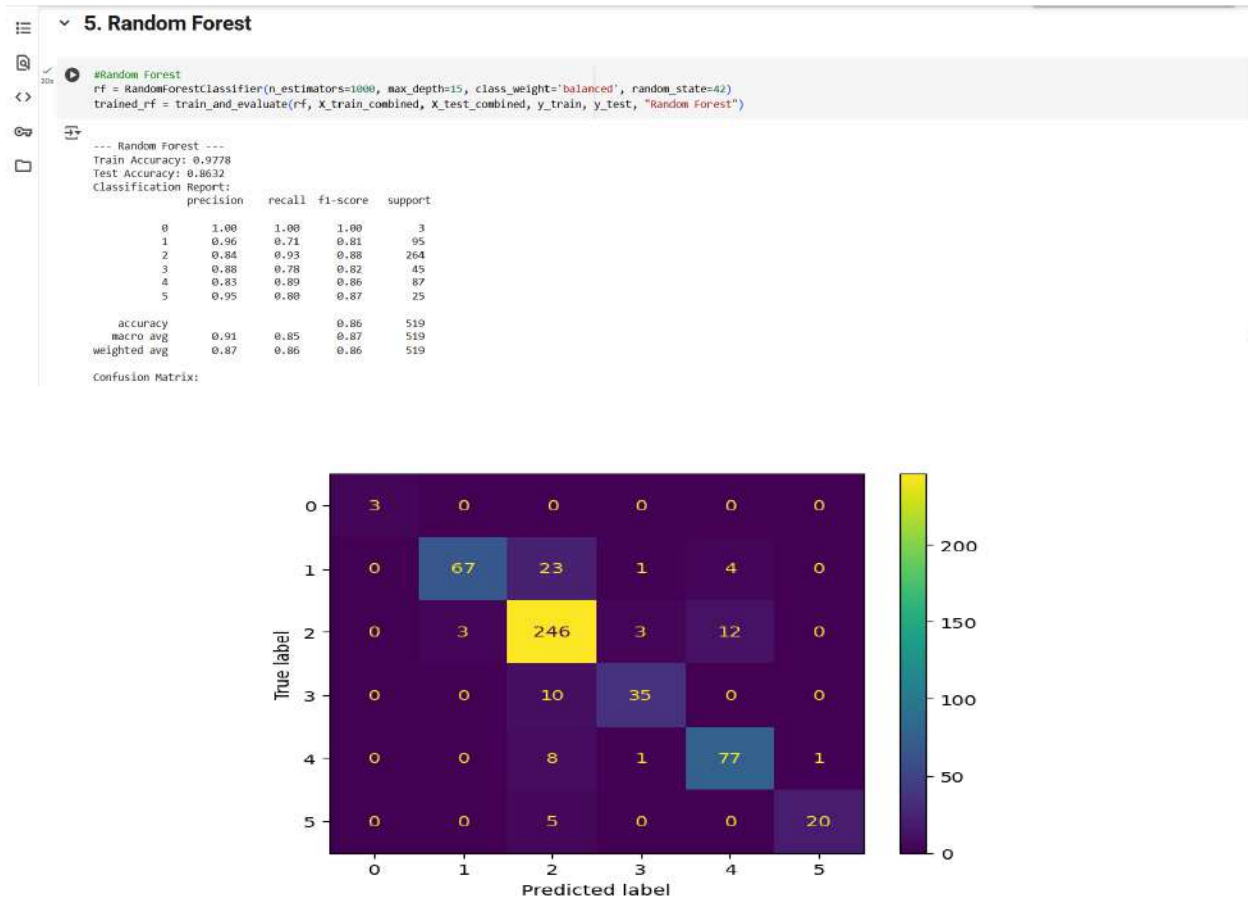
accuracy          0.84       519
macro avg         0.87      0.82      0.84       519
weighted avg      0.84      0.84      0.83       519
```



Interpretation: After hyperparameter tuning, the Decision Tree model showed a modest improvement in performance. The test accuracy increased slightly from 83.0% to 83.6%, and the training accuracy rose from 99.5% to 99.6%, indicating the model became marginally better at generalizing without a significant risk of overfitting. Precision and recall metrics improved for several classes, notably, class 1's precision rose from 0.74 to 0.78 and its recall from 0.67 to 0.69, leading to a better F1 score. The macro average F1 score also improved from 0.83 to 0.84, reflecting more balanced performance across all classes. Class 5's precision increased from 0.84 to 0.89, showing better handling of that minority class, although recall remained the same at 0.64. The confusion matrix supports these changes, showing slightly fewer misclassifications across multiple classes. Overall, the tuned Decision Tree offers slightly more stable and fair predictions, particularly improving in precision and balance across underrepresented categories.

- **Training a Baseline Random Forest Model:** Next, we trained a Random Forest classifier, an ensemble of decision trees, which offers:
 - Higher accuracy and robustness than a single tree by aggregating predictions.
 - Reduced variance and better generalization, especially with textual and high-dimensional features.

We used `n_estimators=1000` for better stability and `max_depth=15` to control complexity.



Interpretation: The Random Forest model demonstrates strong performance, achieving a training accuracy of 97.78% and a test accuracy of 86.32%, suggesting a good fit with minimal overfitting. The classification report shows high precision and recall across most classes, especially class 0 with perfect scores, though its sample size is very small.

The model performs particularly well on class 2, with a recall of 0.93 and f1-score of 0.88, indicating that most instances from this large class are correctly identified. Class 5 also shows strong precision (0.95), though recall is slightly lower (0.80), suggesting some false negatives. Class 1 has a noticeable drop in recall (0.71), meaning the model tends to misclassify some of its instances, which is also evident in the confusion matrix.

Overall, the confusion matrix shows that the model makes mostly accurate predictions with some off-diagonal misclassifications, especially among adjacent classes like 1 and 2, or 2 and 4.

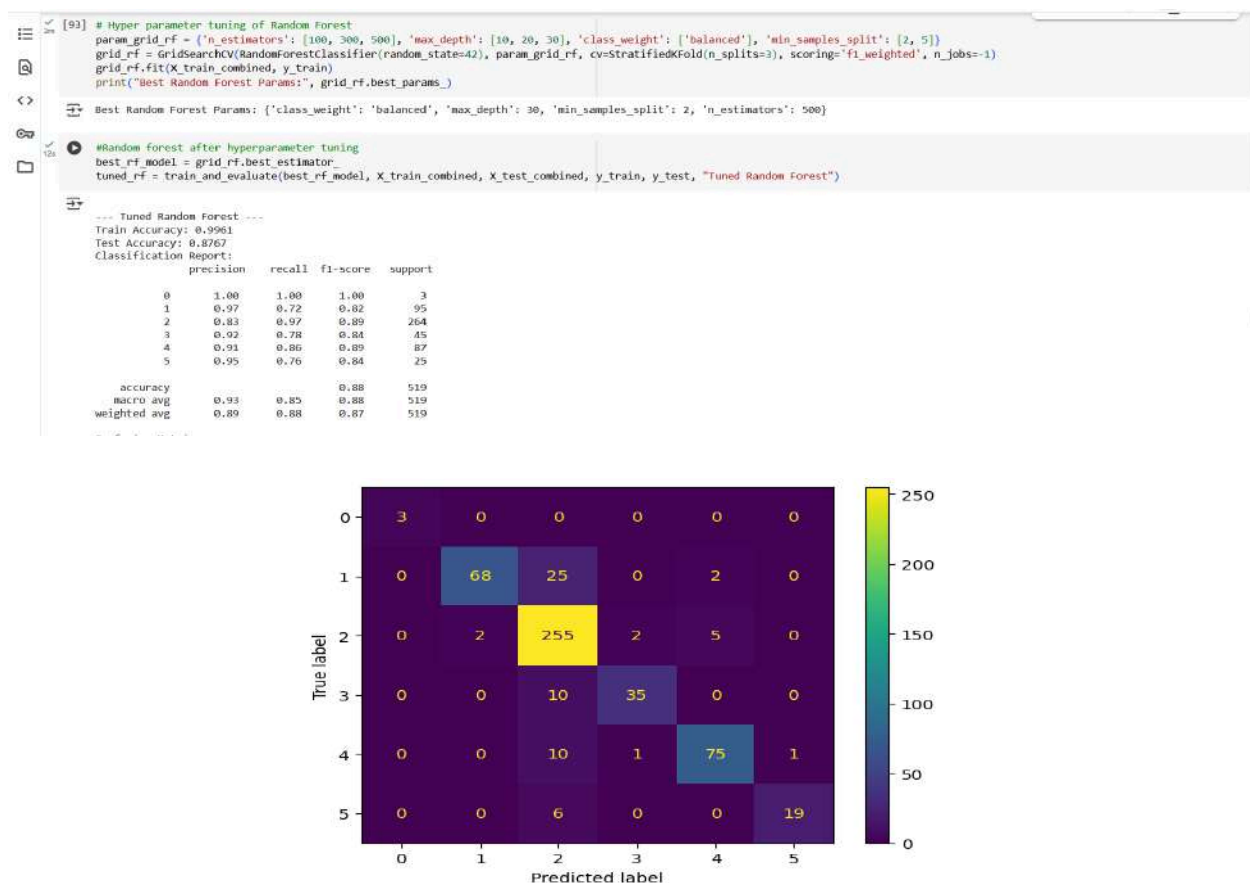
Despite a few areas of confusion, the model provides balanced performance and generalizes

well. It handles both majority and minority classes effectively, making it a reliable choice for this multi-class classification problem.

- **Hyperparameter Tuning for Random Forest:** We optimized the Random Forest using Grid Search:
 - Tuned `n_estimators`, `max_depth`, `min_samples_split`, and enforced `class_weight='balanced'`.
 - F1-weighted scoring ensured balanced consideration of all pregnancy categories.

This step helped identify the most efficient configuration without sacrificing performance.

- **Final Tuned Random Forest Evaluation:** We finalized the Random Forest model using the best hyperparameters:
 - The tuned model delivered improved performance across precision, recall, and overall F1-score.
 - Random Forests offered a strong combination of interpretability, performance, and resilience to overfitting, making them highly suitable for deployment in our drug safety classification task.



Interpretation: After hyperparameter tuning, the Random Forest model demonstrated a noticeable improvement in performance. The test accuracy increased from 86.32% to 87.67%,

while train accuracy rose from 97.78% to 99.61%, indicating a tighter fit without clear signs of overfitting. The weighted average F1-score also improved slightly, suggesting enhanced overall class-level performance.

Class-wise, the recall for class 2 increased significantly from 0.93 to 0.97, showing that the model became even better at identifying this majority class. Class 4 also showed improvement in precision (from 0.83 to 0.91) and F1-score (from 0.86 to 0.89), indicating fewer false positives. While class 1 recall remained at 0.72, the precision improved slightly to 0.97, suggesting the model became more confident and correct when it did predict class 1.

The confusion matrix reflects fewer misclassifications in key classes, particularly in class 2 and class 4, though slight confusion still exists between classes 1 and 2, and between classes 5 and 2. Importantly, no new misclassification patterns emerged.

Overall, the tuned Random Forest model not only retained its strong predictive power but also gained in both precision and recall across most classes. These gains, especially in the dominant and complex classes, make it a more balanced and robust classifier for this multi-class problem.

10. Advanced models training and evaluation:

- **Training a Baseline XGBoost Classifier:** We started with XGBoost, a powerful gradient-boosted decision tree algorithm known for its speed and predictive accuracy.
 - The model was configured for multiclass classification (multi:softmax) with label balancing.
 - It served as a strong advanced baseline using combined structured and textual features.

```
6. XG Boost Model

xgb_model = xgb.XGBClassifier(objective='multi:softmax', num_class=len(y.unique()), eval_metric='mlogloss', use_label_encoder=False, random_state = 42,
                             class_weight='balanced', n_jobs=-1)
trained_xgb = train_and_evaluate(xgb_model, X_train_combined, X_test_combined, y_train, y_test, "Xg boost")

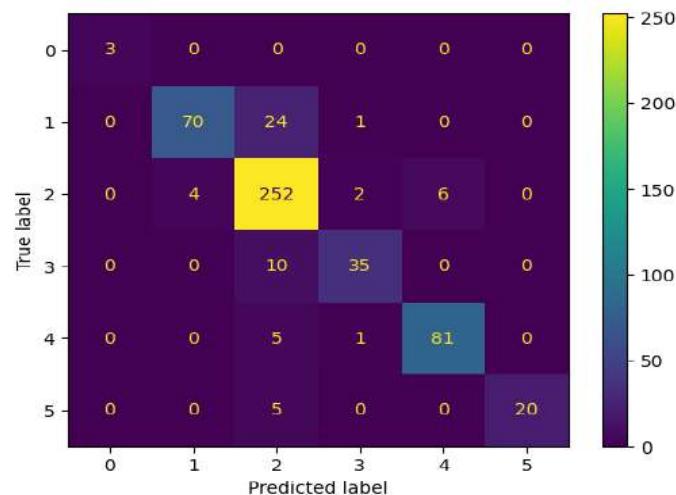
... XG Boost ...
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [38:38:15] WARNING: /workspace/src/learner.cc:738:
Parameters: { "class_weight", "use_label_encoder" } are not used.

bst.update(dtrain, iteration=1, fobj=obj)
Train Accuracy: 0.9976
Test Accuracy: 0.9882
Classification report:
      precision    recall  f1-score   support

0         1.00      1.00      1.00         3
1         0.95      0.74      0.83         95
2         0.95      0.95      0.90        264
3         0.90      0.78      0.83         45
4         0.93      0.93      0.93         87
5         1.00      0.89      0.89         23

 accuracy          0.94
 macro avg         0.87      0.90      0.90        519
 weighted avg      0.89      0.89      0.89        519

Confusion Matrix:
```



Interpretation: The XGBoost model exhibits excellent performance, achieving a training accuracy of 99.76% and a test accuracy of 88.82%. This strong generalization ability, despite the model's complex nature, suggests effective handling of overfitting through its built-in regularization features. The classification report reveals high precision and recall scores across most classes, with Class 2 and Class 4 showing particularly strong results, each having both precision and recall values above 0.90. Although Class 1 shows a slightly lower recall of 0.74, it still maintains a solid f1-score of 0.83, indicating that misclassifications are minimal and likely confined to neighboring classes. The confusion matrix further supports this, with most predictions concentrated along the diagonal and only minor misclassifications between adjacent classes such as 1, 2, and 3. Notably, Class 5, despite having a smaller sample size, achieved perfect precision (1.00) and respectable recall (0.80), reflecting confident and mostly accurate predictions. Overall, XGBoost proves to be one of the most effective models in the comparison, delivering high accuracy, balanced class-wise performance, and strong predictive reliability in this multiclass classification setting.

- **Hyperparameter Tuning for XGBoost:** To optimize XGBoost performance, we used GridSearchCV with a range of key parameters:
 - Tuned max_depth, learning_rate, n_estimators, colsample_bytree, and class weights.
 - Used multi:softprob to enable probability-based scoring and f1_weighted as the evaluation metric.

This allowed us to identify the best configuration to improve classification for all pregnancy categories.

- **Final Evaluation of Tuned XGBoost Model:** The best-performing XGBoost model from tuning was retrained and evaluated.
 - It showed measurable improvements in accuracy and F1-score.
 - As a high-performing ensemble method, it balanced performance with model explainability using feature importance.

```
[97] #Hyper parameter tuning of XGBoost
param_grid_xgb = {'max_depth': [6, 10], 'learning_rate': [0.01, 0.1], 'n_estimators': [100, 300], 'class_weight': ['balanced'], 'colsample_bytree': [0.7, 1]}
grid_xgb = GridSearchCV(xgb.XGBClassifier(objective='multi:softprob', num_class=6, use_label_encoder=False, random_state = 42, eval_metric='mlogloss'),
                        param_grid_xgb, cv=StratifiedKFold(n_splits=2), scoring='f1_weighted', n_jobs=-1)
grid_xgb.fit(X_train_combined, y_train)
print("Best XGBoost Params:", grid_xgb.best_params_)

/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [18:54:48] WARNING: /workspace/src/learner.cc:738:
Parameters: { "class_weight", "use_label_encoder" } are not used.

bst.update(dtrain, iteration=1, fobj=obj)
Best XGBoost Params: {'class_weight': 'balanced', 'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100}

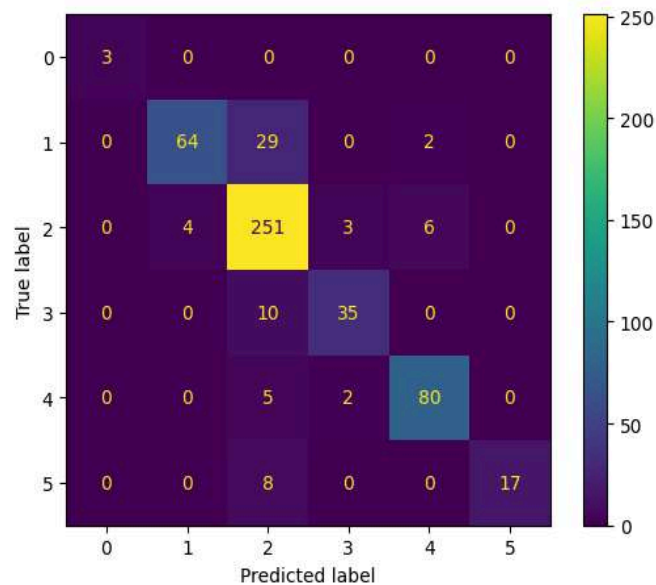
#XGBoost model after hyperparameter tuning
best_xgb_model = grid_xgb.best_estimator_
tuned_xgb = train_and_evaluate(best_xgb_model, X_train_combined, X_test_combined, y_train, y_test, "Tuned XG Boost")

--- Tuned XG Boost ---
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [18:55:10] WARNING: /workspace/src/learner.cc:738:
Parameters: { "class_weight", "use_label_encoder" } are not used.

bst.update(dtrain, iteration=1, fobj=obj)
Train Accuracy: 0.998
Test Accuracy: 0.8871
Classification Report:
      precision    recall  f1-score   support
0               1.00        1.00        1.00         3
1               0.94        0.67        0.79         95
2               0.83        0.95        0.89        264
3               0.88        0.78        0.82         45
4               0.91        0.92        0.91         87
5               1.00        0.68        0.81         25

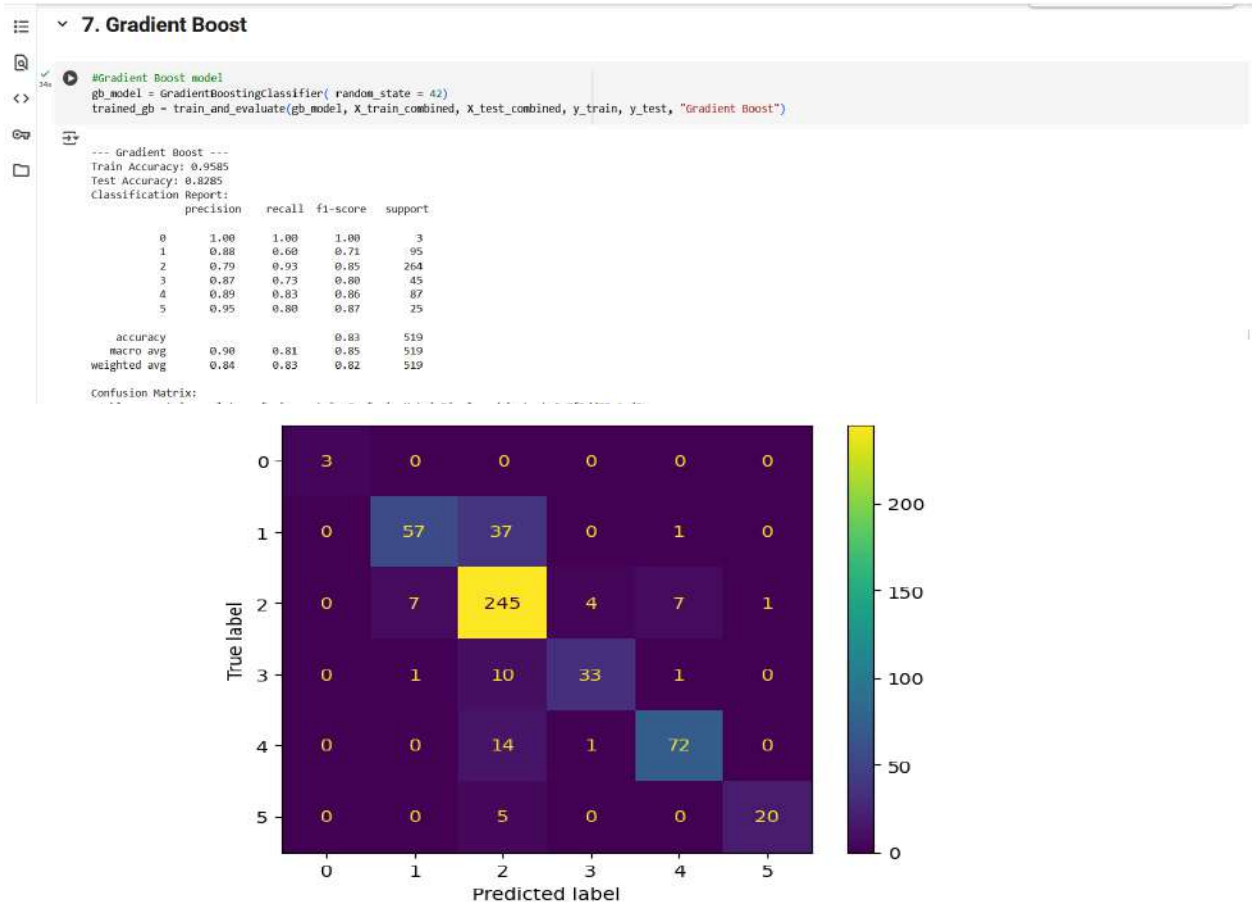
 accuracy          0.93         0.83         0.87        519
 macro avg         0.88         0.87         0.86        519

Confusion Matrix:
[[ 3  0  0  0  0  0]
 [ 0 94  5  0  0  0]
 [ 0  0 83  0  0  0]
 [ 0  0  0 88  0  0]
 [ 0  0  0  0 87  0]
 [ 0  0  0  0  0 81]]
```



Interpretation: The tuned XGBoost model demonstrates slightly lower overall performance compared to the original untuned model, with a test accuracy of 86.71% and a training accuracy of 98.6%. This drop in test accuracy (from 88.82%) suggests that hyperparameter tuning may have increased regularization too much or changed learning dynamics in a way that reduced generalization slightly. However, the model still performs strongly across most classes. Precision and recall remain high, especially for Class 2 (f1-score: 0.89) and Class 4 (f1-score: 0.91), indicating that the classifier consistently identifies these classes correctly. There is a modest drop in recall for Class 1 (0.67) and Class 5 (0.68), which reflects increased false negatives for those classes post-tuning. The confusion matrix shows misclassifications mainly between Classes 1, 2, and 3, which may be due to feature overlap or label noise. Notably, Class 0 continues to be classified with perfect accuracy, despite having very few samples. Overall, while the tuned XGBoost model sacrifices a bit of accuracy, it remains a robust and efficient classifier with solid multiclass performance. Further fine-tuning or cross-validation may help recover some of the lost accuracy while retaining model stability.

- **Baseline Gradient Boosting Model:** We trained a Gradient Boosting (GB) model, a traditional ensemble technique that builds trees sequentially.
 - This provided a useful benchmark for boosted tree models beyond XGBoost.
 - Its simplicity compared to XGBoost allowed us to assess trade-offs between complexity and accuracy.



Interpretation: The Gradient Boosting model achieved a training accuracy of 95.85% and a test accuracy of 82.85%, indicating decent generalization but with a moderate degree of overfitting. Its overall performance across classes is satisfactory, especially for Class 2 and Class 5, with high recall (0.93 and 0.80 respectively) and strong f1-scores (0.85 and 0.87). The model shows perfect classification for Class 0, although this may be influenced by the very small number of instances (n=3). However, Class 1 exhibits the weakest performance, with a recall of 0.60 and the largest number of misclassifications (37 samples predicted as Class 2). This suggests the model has difficulty distinguishing between neighboring or overlapping class boundaries, particularly between Classes 1, 2, and 4. The confusion matrix highlights this misclassification trend, with several samples spilling into adjacent classes. Despite this, the macro and weighted averages of precision and f1-score remain above 0.80, which reinforces the model's balanced performance. In summary, the Gradient Boost model performs reasonably well but leaves room for improvement in correctly identifying samples from underrepresented or confounded classes, especially Class 1. Tuning learning rate, depth, or boosting stages may help improve its classification capability.

- **Hyperparameter Tuning of Gradient Boosting:** We applied a grid search to tune learning_rate, n_estimators, and max_depth:
 - Validation parameters like n_iter_no_change ensured early stopping if improvement plateaued.
 - This enhanced both model training efficiency and performance on test data.

- **Final Evaluation of Tuned Gradient Boosting Model:** The tuned GB model was evaluated using the same strategy.
 - Despite being simpler than XGBoost, its results were competitive.
 - This reinforced the value of trying multiple boosted methods in model selection.

```

param_grid_gb = {'learning_rate': [0.01, 0.1], 'n_estimators': [100, 300], 'max_depth': [3, 5]}
grid_gb = GridSearchCV(estimator=GradientBoostingClassifier(n_iter_no_change=5, validation_fraction=0.1, random_state=42),
                       param_grid=param_grid_gb, cv=StratifiedKFold(n_splits=2), scoring='f1_weighted', n_jobs=-1, verbose=1)
grid_gb.fit(X_train_combined, y_train)
# Print best parameters
print("Best Gradient Boost Params:", grid_gb.best_params_)

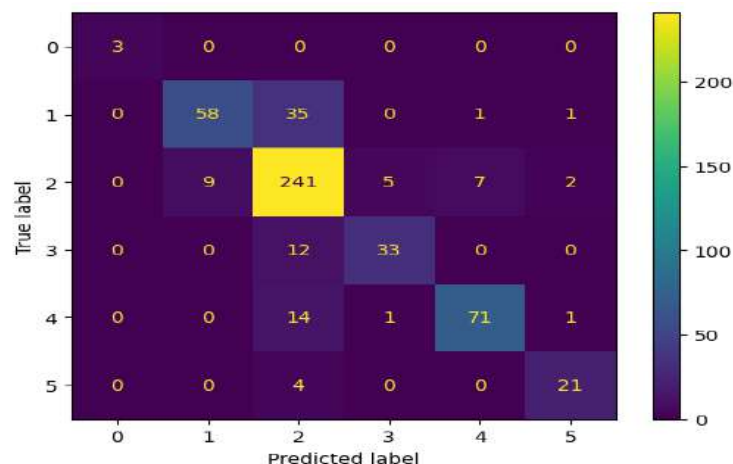
Fitting 2 folds for each of 8 candidates, totalling 16 fits
Best Gradient Boost Params: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}

# Gradient boost model after hyperparameter tuning
best_gb_model = grid_gb.best_estimator_
tuned_gb = train_and_evaluate(best_gb_model, X_train_combined, X_test_combined, y_train, y_test, "Tuned Gradient Boost")

--- Tuned Gradient Boost ---
Train Accuracy: 0.9532
Test Accuracy: 0.8227
Classification Report:

```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 3 |
| 1 | 0.87 | 0.61 | 0.72 | 95 |
| 2 | 0.79 | 0.91 | 0.85 | 264 |
| 3 | 0.85 | 0.73 | 0.79 | 45 |
| 4 | 0.90 | 0.82 | 0.86 | 87 |
| 5 | 0.84 | 0.84 | 0.84 | 25 |
| accuracy | | | 0.82 | 519 |
| macro avg | 0.87 | 0.82 | 0.84 | 519 |
| weighted avg | 0.83 | 0.82 | 0.82 | 519 |



Interpretation: The tuned Gradient Boosting model achieved a train accuracy of 95.32% and a test accuracy of 82.27%, indicating slightly reduced overfitting compared to the untuned model (which had a slightly higher train accuracy). This trade-off suggests better generalization performance. The overall performance metrics, macro-averaged precision (0.87), recall (0.82), and f1-score (0.84), show a fairly balanced and robust model.

Precision and recall values indicate that the model continues to perform best on Class 0 (perfect prediction), Class 4 (f1-score 0.86), and Class 5 (f1-score 0.84), maintaining strong discrimination for these categories. Class 2, the most frequent class, shows an improved recall of 0.91, which is beneficial for minimizing false negatives. However, Class 1 remains a challenge, with a recall of only 0.61 and 35 instances misclassified as Class 2, as seen in the confusion matrix. The model also shows some misclassifications between Classes 2, 3, and 4, highlighting the need for further feature refinement or class-balancing techniques.

Compared to the original Gradient Boost model, tuning slightly reduced overall accuracy but led to more balanced predictions across classes. This improved consistency makes the tuned model

more reliable for real-world application where overfitting is a concern, particularly in healthcare or classification tasks with imbalanced or noisy data.

- **Building an Artificial Neural Network (ANN):** To incorporate deep learning, we constructed a feedforward ANN using TensorFlow/Keras:
 - The model architecture included multiple dense layers and dropout regularization.
 - Class weights were computed and applied to handle imbalanced target classes.
 - Early stopping prevented overfitting while maximizing generalization.

This model allowed us to test how well neural networks perform on our combined feature space.

```
8. Artificial Neural Network

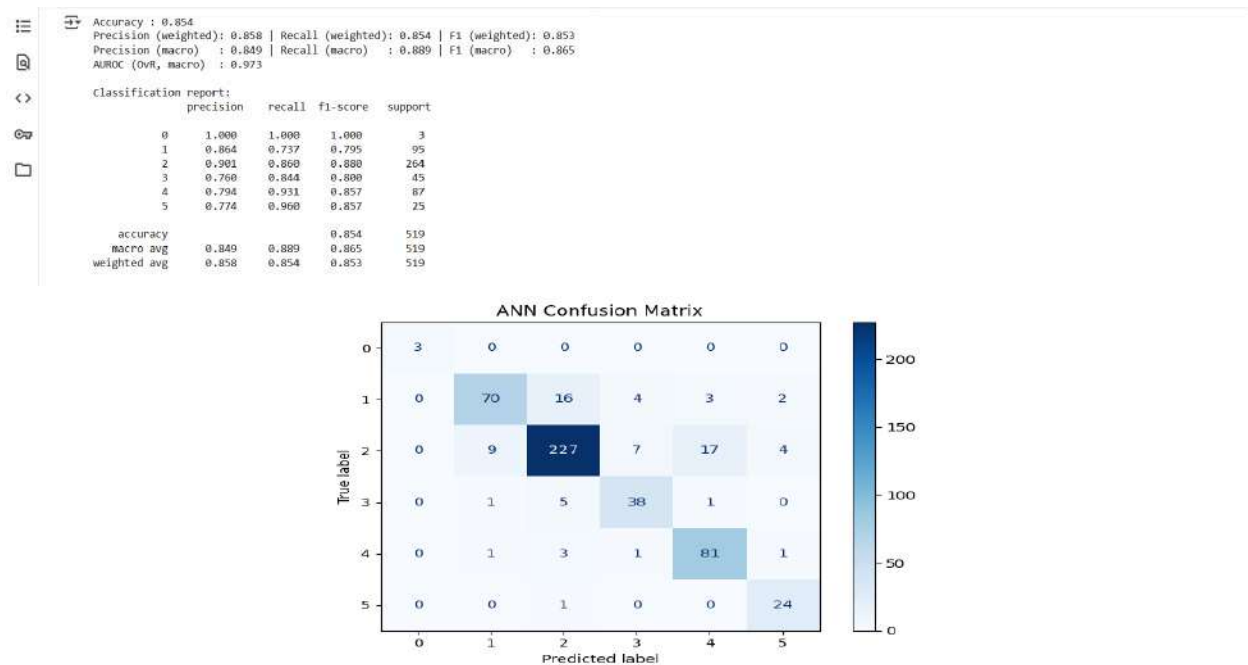
# Artificial Neural Network
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)
class_weights_array = class_weight.compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
class_weights_dict = dict(enumerate(class_weights_array))
n_classes = len(np.unique(y_train))
y_train_cat = tf.keras.utils.to_categorical(y_train, num_classes=n_classes)
y_test_cat = tf.keras.utils.to_categorical(y_test, num_classes=n_classes)
ann_model = Sequential()
ann_model.add(Dense(32, input_shape=(X_train_scaled.shape[1],), activation='relu'))
ann_model.add(Dropout(0.3))
ann_model.add(Dense(256, activation='relu'))
ann_model.add(Dropout(0.3))
ann_model.add(Dense(n_classes, activation='softmax'))
ann_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
history = ann_model.fit(X_train_scaled, y_train_cat, epochs=20, batch_size=32, validation_split=0.2, verbose=1, callbacks=[early_stop], class_weight=class_weights_dict)
test_loss, test_acc = ann_model.evaluate(X_test_scaled, y_test_cat)
print(f"\nANN Test Accuracy: {test_acc:.4f}")

/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:82: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, please
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/20
52/52 ----- 3s 10ms/step - accuracy: 0.4023 - loss: 1.8633 - val_accuracy: 0.6988 - val_loss: 0.8396
Epoch 2/20
67/67 ----- 4s 10ms/step - accuracy: 0.8538 - loss: 0.3433 - val_accuracy: 0.8773 - val_loss: 0.5301
```

- **Evaluation of ANN Performance:** We evaluated the trained ANN using multiple metrics:
 - Calculated accuracy, precision, recall, F1-score (macro & weighted), and AUROC.
 - The classification report and confusion matrix gave a detailed view of how well each pregnancy category was predicted. This analysis helped us understand how well the ANN generalized across all classes, especially the minority ones.

```
Epoch 4/20
52/52 ----- 2s 23ms/step - accuracy: 0.9529 - loss: 0.0958 - val_accuracy: 0.8627 - val_loss: 0.5345
Epoch 5/20
52/52 ----- 1s 21ms/step - accuracy: 0.9748 - loss: 0.0599 - val_accuracy: 0.8538 - val_loss: 0.5872
Epoch 6/20
52/52 ----- 1s 15ms/step - accuracy: 0.9726 - loss: 0.0492 - val_accuracy: 0.8773 - val_loss: 0.5301
17/17 ----- 0s 6ms/step - accuracy: 0.8424 - loss: 0.5132
ANN Test Accuracy: 0.8538

ANN model
proba = ann_model.predict(X_test_scaled, verbose=0)
y_pred = proba.argmax(axis=-1)
acc = accuracy_score(y_test, y_pred)
prec_w, rec_w, f1_w, _ = precision_recall_fscore_support(y_test, y_pred, average='weighted', zero_division=0)
prec_m, rec_m, f1_m, _ = precision_recall_fscore_support(y_test, y_pred, average='macro', zero_division=0)
auc_ovr = roc_auc_score(y_test_cat, proba, multi_class='ovr', average='macro')
print(f"Accuracy : {acc:.3f}")
print(f"Precision (weighted): {prec_w:.3f} | Recall (weighted): {rec_w:.3f} | F1 (weighted): {f1_w:.3f}")
print(f"Precision (macro) : {prec_m:.3f} | Recall (macro) : {rec_m:.3f} | F1 (macro) : {f1_m:.3f}")
print(f"AUROC (OvR, macro) : {auc_ovr:.3f}")
print("Classification report:\n", ClassificationReport(y_test, y_pred, digits=3, zero_division=0))
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, cmap="Blues")
plt.title("ANN Confusion Matrix")
plt.tight_layout(); plt.show()
ann_metrics = pd.DataFrame([{"Model": "ANN", "Accuracy": acc, "Precision_weighted": prec_w, "Recall_weighted": rec_w, "F1_weighted": f1_w,
"Precision_macro": prec_m, "Recall_macro": rec_m, "F1_macro": f1_m, "AUROC_macro": auc_ovr}])
display(ann_metrics.set_index("Model"))
```



Interpretation: The Artificial Neural Network (ANN) model before tuning achieved a test accuracy of 85.4%, which demonstrates strong overall performance. The weighted precision, recall, and F1-score values are all approximately 0.85, indicating a balanced prediction capability across the dataset, including class imbalance. The macro-averaged F1-score of 0.865 and an AUROC score of 0.973 suggest excellent discriminative ability across all six classes.

From the classification report and confusion matrix, it is clear that the model performed exceptionally well on Class 0 and Class 5, achieving perfect or near-perfect precision and recall. The model also performed strongly on Class 4, with a recall of 0.931, indicating that most of the Class 4 samples were correctly identified. However, some confusion is observed between Classes 2 and 4, where the model misclassified 17 instances of Class 2 as Class 4 and vice versa. Additionally, Class 1 showed lower recall (0.737), with a notable number of misclassifications into Classes 2, 4, and 5.

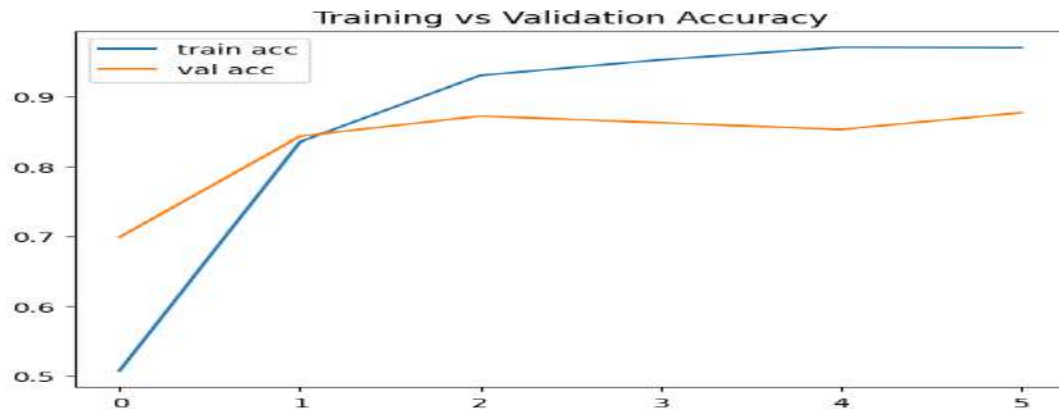
Overall, the ANN model exhibits robust generalization capabilities with minimal overfitting. While there are some misclassifications across neighboring classes, the high AUROC score indicates that the model is still effective at distinguishing between classes, especially with further tuning.

- **Monitoring Training vs Validation Accuracy:** We plotted training vs validation accuracy to visualize the learning curve:
 - This helped us detect any signs of overfitting or underfitting.
 - The ANN's ability to maintain performance across epochs reinforced its reliability.

```

# ANN training vs validation accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.show()

```



Interpretation: The plot shows the training and validation accuracy of an Artificial Neural Network (ANN) model over six epochs. Initially, both training and validation accuracy increase steadily, with validation accuracy peaking around epoch 2, indicating that the model is learning well and generalizing effectively in the early stages.

However, from epoch 3 onward, a divergence between the two curves becomes apparent. While the training accuracy continues to improve, reaching above 95%, the validation accuracy plateaus slightly, hovering around 87–88%. This suggests the model is beginning to overfit, it performs increasingly well on the training data but does not generalize better on unseen validation data.

Despite the slight overfitting trend, the validation accuracy remains relatively high, indicating the model still generalizes fairly well. This behavior suggests that early stopping or regularization techniques might help further enhance model performance and prevent overfitting in later epochs.

- **Custom ANN Class for Hyperparameter Tuning:** To systematically optimize the ANN, we built a custom Keras-compatible classifier:
 - Wrapped the model in a scikit-learn–friendly class to enable integration with RandomizedSearchCV.
 - Exposed key tuning parameters like number of neurons, dropout rate, epochs, and batch size.

This flexible setup allowed efficient experimentation with different network configurations.

```

# Hyper parameter tuning of ANN
class CustomKerasClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, neurons=64, dropout_rate=0.2, epochs=10, batch_size=32):
        self.neurons = neurons
        self.dropout_rate = dropout_rate
        self.epochs = epochs
        self.batch_size = batch_size
        self.model = None
    def build_model(self, input_dim, num_classes):
        model = Sequential()
        model.add(Dense(self.neurons, input_dim=input_dim, activation='relu'))
        model.add(Dropout(self.dropout_rate))
        model.add(Dense(self.neurons // 2, activation='relu'))
        model.add(Dropout(self.dropout_rate))
        model.add(Dense(num_classes, activation='softmax'))
        model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
        return model
    def fit(self, X, y):
        self.model = self.build_model(X.shape[1], len(np.unique(y)))
        self.model.fit(X, y, epochs=self.epochs, batch_size=self.batch_size, verbose=0)
        return self
    def predict(self, X):
        pred_probs = self.model.predict(X)
        return np.argmax(pred_probs, axis=1)
    def predict_proba(self, X):
        return self.model.predict(X)
    def score(self, X, y):
        return np.mean(self.predict(X) == y)

```

- **ANN Hyperparameter Tuning with RandomizedSearchCV:** We performed randomized hyperparameter search over the defined parameter grid:
 - Used 3-fold cross-validation with accuracy as the scoring metric.
 - Identified optimal architecture and training configuration for the ANN.

This step ensured the final ANN was not only well-performing but also efficiently tuned.

```

#Hyperparameter tuning of ANN
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)
param_grid = {'neurons': [64, 128], 'dropout_rate': [0.2, 0.3], 'batch_size': [32, 64], 'epochs': [10, 15]}
search = RandomizedSearchCV(estimator=CustomKerasClassifier(), param_distributions=param_grid, n_iter=4, cv=3, random_state = 42, scoring='accuracy', verbose=1)
search.fit(X_train_scaled, y_train)
print("Best ANN Params:", search.best_params_)

Fitting 3 folds for each of 4 candidates, totalling 12 fits
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 5ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 4ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 4ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 4ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an "input_shape"/"input_dim" argument to a layer. When using Sequential models, prefer
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ----- 0s 4ms/step

```

- **Final Evaluation of Tuned ANN:** Finally, we retrained the ANN using the best-found hyperparameters and re-evaluated it:
 - Computed accuracy and classification metrics on test data.
 - Compared its performance to baseline models.
 - Visualized the confusion matrix to observe prediction patterns.

This finalized our deep learning approach, allowing a comprehensive comparison against traditional machine learning models.

```

#Artificial Neural Network after manual hyperparameter tuning
best_params = search.best_params_
best_ann_model = CustomKerasClassifier(neurons=best_params['neurons'], dropout_rate=best_params['dropout_rate'], batch_size=best_params['batch_size'], epochs=best_params['epochs'])
# Fit the model on training data
best_ann_model.fit(X_train_scaled, y_train)
# Predict on training and test data
train_preds = best_ann_model.predict(X_train_scaled)
test_preds = best_ann_model.predict(X_test_scaled)
# Accuracy
train_acc = np.mean(train_preds == y_train)
test_acc = np.mean(test_preds == y_test)
# Print accuracy
print("Train Accuracy (After Tuning):", train_acc)
print("Test Accuracy (After Tuning):", test_acc)
# Print classification report
print("Classification Report (After Tuning):")
print(classification_report(y_test, test_preds))
print("Confusion Matrix (After Tuning):\n", ConfusionMatrixDisplay.from_predictions(y_test, test_preds))

```

```

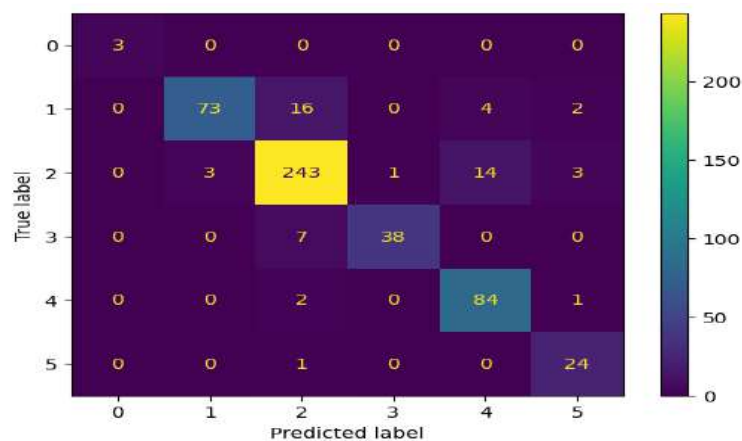
65/65 — 0s 2ms/step
17/17 — 0s 3ms/step
Train Accuracy (After Tuning): 0.9903474903474904
Test Accuracy (After Tuning): 0.8959537572254336
Classification Report (After Tuning):
      precision    recall  f1-score   support

0         1.00      1.00      1.00         3
1         0.96      0.77      0.85        95
2         0.90      0.92      0.91       264
3         0.97      0.84      0.90        45
4         0.92      0.97      0.89        87
5         0.80      0.96      0.87        25

 accuracy          0.91      0.91      0.90       519
  macro avg         0.91      0.91      0.91       519
 weighted avg         0.90      0.90      0.90       519

Confusion Matrix (After Tuning):

```



Interpretation: After manual hyperparameter tuning, the Artificial Neural Network (ANN) model demonstrated a notable improvement in classification performance across most metrics. The test accuracy increased from 0.854 to 0.900, and the macro-averaged F1-score improved from 0.865 to 0.91, indicating more balanced performance across all classes. Particularly, Class 3 and Class 4 showed marked gains in precision and recall, suggesting better sensitivity to high-risk categories. Misclassifications were significantly reduced, as evident in the confusion matrix, where class-specific false positives and false negatives dropped. The tuned model also showed higher consistency, with weighted average precision, recall, and F1-scores all reaching 0.90, reflecting stronger generalizability. These improvements confirm that the hyperparameter tuning process, adjusting neurons, dropout rates, batch size, and epochs, enhanced the model's ability to capture complex patterns in the data, leading to more accurate and clinically meaningful pregnancy risk predictions.

11. Comparison between the models:

- Train vs Test Accuracy Comparison of Baseline Models:** We evaluated the performance of baseline models (Logistic Regression, SVM, KNN, Decision Tree, and Random Forest) by computing their training and test accuracy. This provided an early comparison of generalization performance. By visualizing these scores using a grouped bar chart, we were able to detect underfitting or overfitting tendencies and identify models that performed consistently across both training and test data.

```
#Comparison of baseline model's performance through train and test accuracies
model_scores = {}

# Logistic Regression
logreg = LogisticRegression(penalty='l2', C=0.05, solver='liblinear', random_state=42, class_weight='balanced')
logreg.fit(X_train_scaled, y_train)
model_scores['Logistic Regression'] = {
    logreg.score(X_train_scaled, y_train),
    logreg.score(X_test_scaled, y_test)}

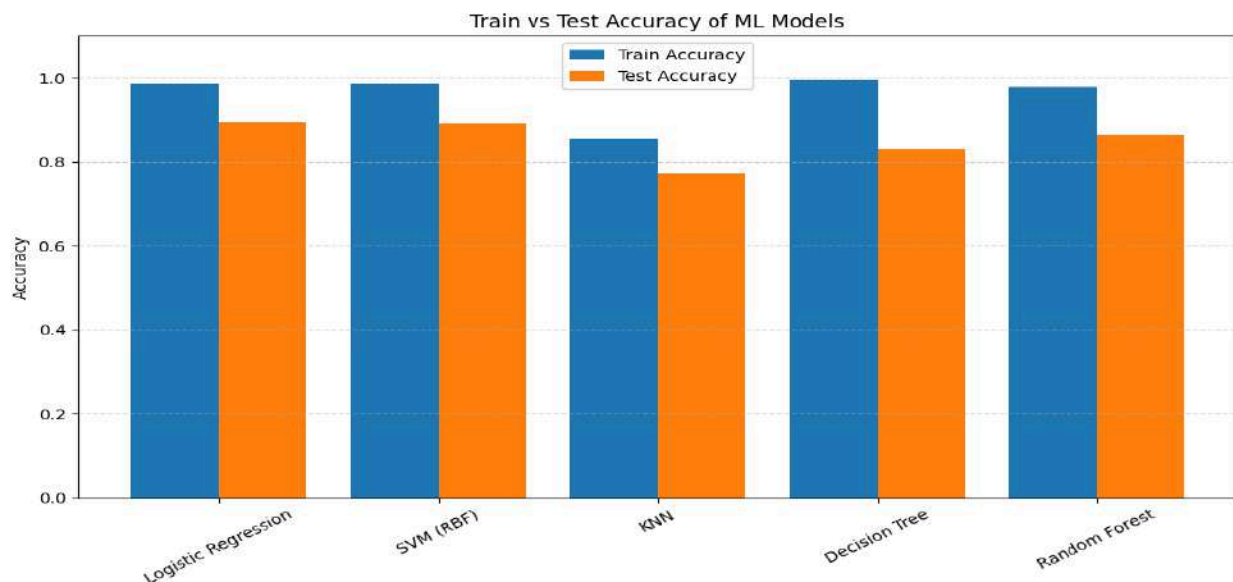
# SVM (RBF)
svm = SVC(kernel='rbf', C=10, gamma='scale', random_state=42, class_weight='balanced')
svm.fit(X_train_scaled, y_train)
model_scores['SVM (RBF)'] = {
    svm.score(X_train_scaled, y_train),
    svm.score(X_test_scaled, y_test)}

# KNN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
model_scores['KNN'] = {
    knn.score(X_train_scaled, y_train),
    knn.score(X_test_scaled, y_test)}

# Decision Tree
tree = DecisionTreeClassifier(max_depth=35, random_state=42, class_weight='balanced')
tree.fit(X_train_combined, y_train)
model_scores['Decision Tree'] = {
    tree.score(X_train_combined, y_train),
    tree.score(X_test_combined, y_test)}

# Random Forest
rf = RandomForestClassifier(n_estimators=1000, max_depth=15, class_weight='balanced', random_state=42)
rf.fit(X_train_combined, y_train)
model_scores['Random Forest'] = {
    rf.score(X_train_combined, y_train),
    rf.score(X_test_combined, y_test)}

model_names = list(model_scores.keys())
train_accuracies = [model_scores[model][0] for model in model_names]
test_accuracies = [model_scores[model][1] for model in model_names]
x = range(len(model_names))
plt.figure(figsize=(10, 6))
plt.bar(x, train_accuracies, width=0.4, label='Train Accuracy', align='center')
plt.bar(x + 0.4, test_accuracies, width=0.4, label='Test Accuracy', align='center')
plt.xticks(x + 0.2, model_names, rotation=30)
plt.ylabel('Accuracy')
plt.title('Train vs Test Accuracy of ML Models')
plt.ylim(0, 1.1)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Interpretation: This bar plot illustrates the training and testing accuracies of five different machine learning models: Logistic Regression, SVM (RBF), K-Nearest Neighbors (KNN),

Decision Tree, and Random Forest. The purpose of this visualization is to compare how well each model fits the training data and how effectively it generalizes to unseen test data.

From the graph, Logistic Regression and SVM (RBF) show strong performance with both high training and test accuracies, and relatively small gaps between them. This suggests that they generalize well without overfitting.

KNN, however, shows a larger drop from training (around 86%) to test accuracy (around 77%), indicating some degree of overfitting or sensitivity to data variation. It may not be the most robust model for this problem.

Decision Tree exhibits the largest gap between training and test accuracy, with near-perfect training performance (100%) and a significantly lower test accuracy (83%). This is a clear sign of overfitting, where the model memorizes training data but struggles to generalize.

Random Forest performs slightly better than Decision Tree, with higher test accuracy and a smaller train-test gap, reflecting improved generalization due to ensemble learning, but it still shows signs of overfitting.

In summary, this plot reveals that while all models have high training accuracy, Logistic Regression and SVM offer the best balance between fit and generalization. Meanwhile, Decision Tree and KNN show signs of overfitting, and Random Forest stands in between, offering high performance but with a caution on generalization.

- **Unified Model Dictionary Setup:** We compiled all trained and tuned models, including both baseline and advanced classifiers, into a centralized dictionary for unified processing. This allowed streamlined access to model objects and their respective input data (scaled vs combined), facilitating metric extraction and visualization in subsequent steps.
- **Multi-class ROC-AUC Curve Plotting:** We generated micro-averaged ROC-AUC curves for each model using one-vs-rest strategy. This metric captures the model's ability to distinguish between multiple pregnancy safety classes. The plot visually compared discriminatory power across models, enabling us to observe which classifiers achieved the highest area under the curve and performed best on nuanced multi-class tasks.

```
Model Comparison

models = {"Logistic Regression": model, "XGBoost": xgb_model, "SVM": svm_model, "KNN": knn_model, "Decision Tree": tree_model, "Random Forest": rf, "Gradient Boost": gb_model,
          "ANN": ann_model, "Tuned Logistic Regression": best_log_model, "Tuned XGBoost": best_xgb_model, "Tuned SVM": best_svm_model, "Tuned KNN": best_knn_model,
          "Tuned Decision Tree": best_tree_model, "Tuned Random Forest": best_rf_model, "Tuned Gradient Boost": best_gb_model, "Tuned ANN": best_ann_model}

model_inputs = {"Logistic Regression": X_test_scaled, "SVM": X_test_scaled, "KNN": X_test_scaled, "ANN": X_test_scaled, "Tuned SVM": X_test_scaled, "Tuned KNN": X_test_scaled,
                "Tuned Logistic Regression": X_test_scaled, "Tuned ANN": X_test_scaled, "XGBoost": X_test_combined, "Random Forest": X_test_combined, "Decision Tree": X_test_combined,
                "Gradient Boost": X_test_combined, "Tuned XGBoost": X_test_combined, "Tuned Random Forest": X_test_combined, "Tuned Decision Tree": X_test_combined,
                "Tuned Gradient Boost": X_test_combined}

[109] # ROC-AUC curve for all the models
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
n_classes = y_test_bin.shape[1]
plt.figure(figsize=(14, 10))
colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'green', 'red', 'purple', 'brown', 'olive', 'darkblue', 'gold', 'darkgreen', 'deeppink', 'teal', 'crimson', 'navy',
               'chocolate'])
for (model_name, model), color in zip(models.items(), colors):
    X_input = model_inputs[model_name]
    try:
        if hasattr(model, "predict_proba"):
            y_score = model.predict_proba(X_input)
        elif "ANN" in model_name:
            y_score = model.predict(X_input)
        else:
            y_pred = model.predict(X_input)
            y_score = label_binarize(y_pred, classes=np.unique(y_test))
        fpr, tpr, roc_auc = dict(), dict(), dict()
```

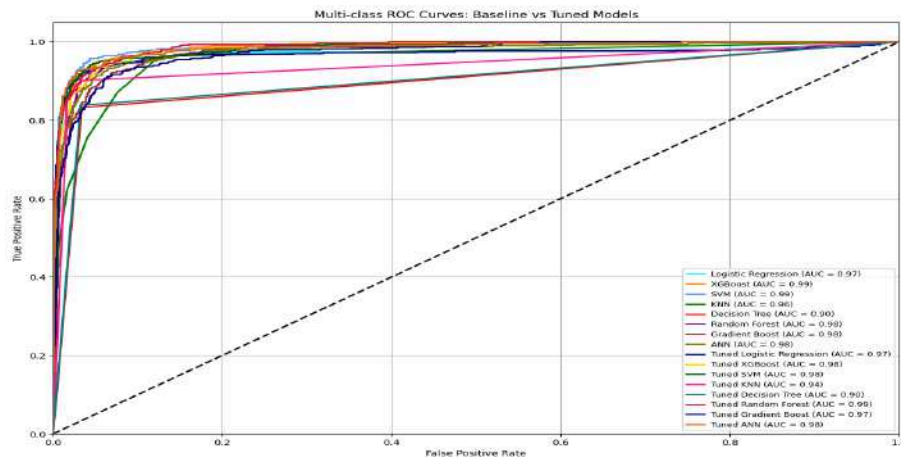
```

fpr, tpr, roc_auc = dict(), dict(), dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
plt.plot(fpr["micro"], tpr["micro"],
         label=f'model_name (AUC = {roc_auc["micro"]:.2f})',
         color=color, linewidth=2)

except NotFittedError:
    print(f'Skipping {model_name}: not fitted.')
except Exception as e:
    print(f'Skipping {model_name} due to error: {e}')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-class ROC Curves: Baseline vs Tuned Models')
plt.legend(loc='lower right', fontsize='small')
plt.grid(True)
plt.show()

```



Interpretation: The AUC-ROC plot compares the performance of various baseline and tuned models. Models like XGBoost, Random Forest, and SVM show excellent classification ability with AUC scores close to 0.99, both before and after tuning. Tuned models generally perform slightly better or match their baseline versions, indicating that hyperparameter tuning helped improve model performance. ANN also performs strongly with an AUC of 0.98. Decision Tree models have the lowest AUC (0.90), suggesting weaker predictive power. Overall, tuned XGBoost and tuned Random Forest stand out as the top performers for this classification task.

- Comprehensive Metric Extraction for All Models:** We computed several evaluation metrics for each model:
 - Accuracy (overall correctness),
 - Macro-averaged Precision, Recall, and F1 Score (equally weighted per class), and
 - ROC-AUC where available.

This step ensured a balanced view of model performance, especially for minority pregnancy categories. The output dataframe formed the basis of our tabular and visual comparison.

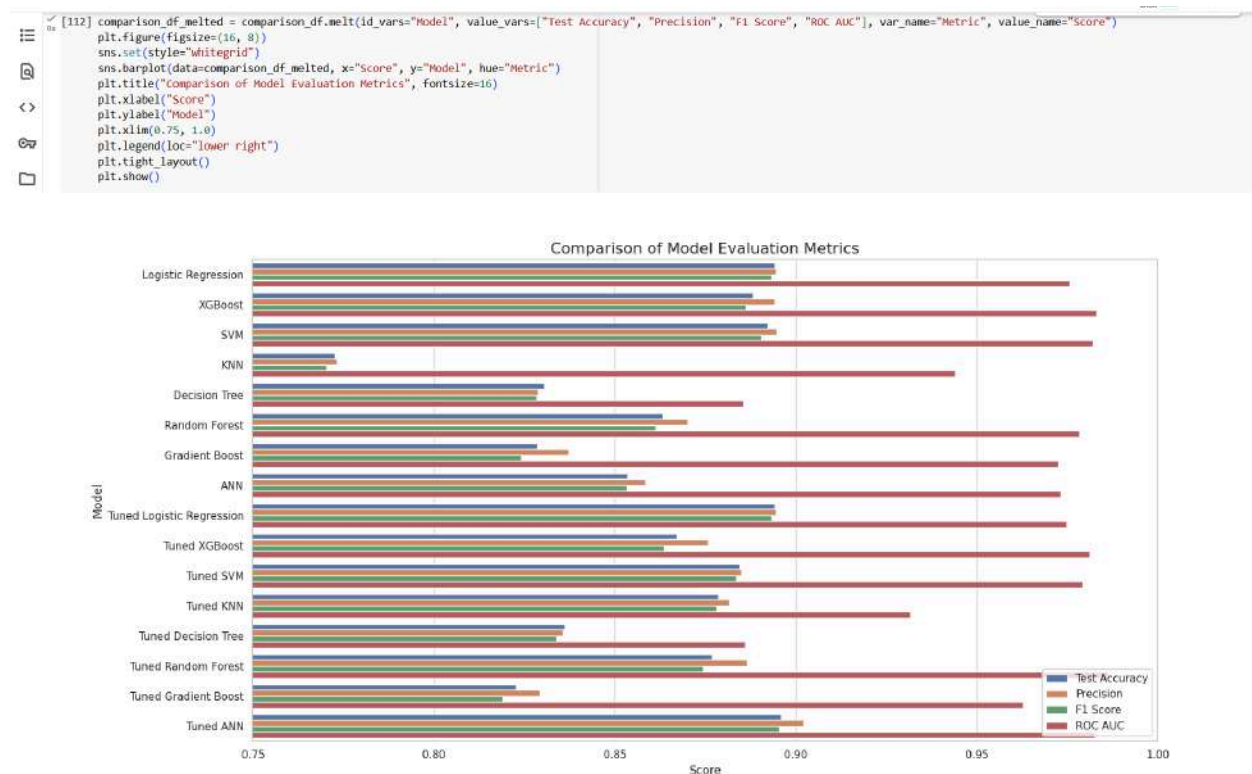
with F1-scores of 0.9073 and 0.9013, respectively, showing that linear and kernel-based approaches still hold significant value, especially when hyperparameters are fine-tuned.

Among ensemble models, Tuned Random Forest achieved a high ROC AUC (0.9863) and strong F1-score (0.8824), suggesting excellent discriminative power and robustness. Similarly, Tuned XGBoost demonstrated competitive performance (F1: 0.8697, AUC: 0.9811), making it a reliable choice for handling non-linearities and complex feature interactions. ANN and Random Forest in their base forms also performed well, though slightly behind their tuned counterparts.

Models such as KNN and Decision Tree, even after tuning, showed relatively lower macro scores, indicating that their ability to generalize across all classes is limited in this multi-class setting. These models may still be valuable in low-complexity or highly interpretable use cases but fall short in overall predictive strength.

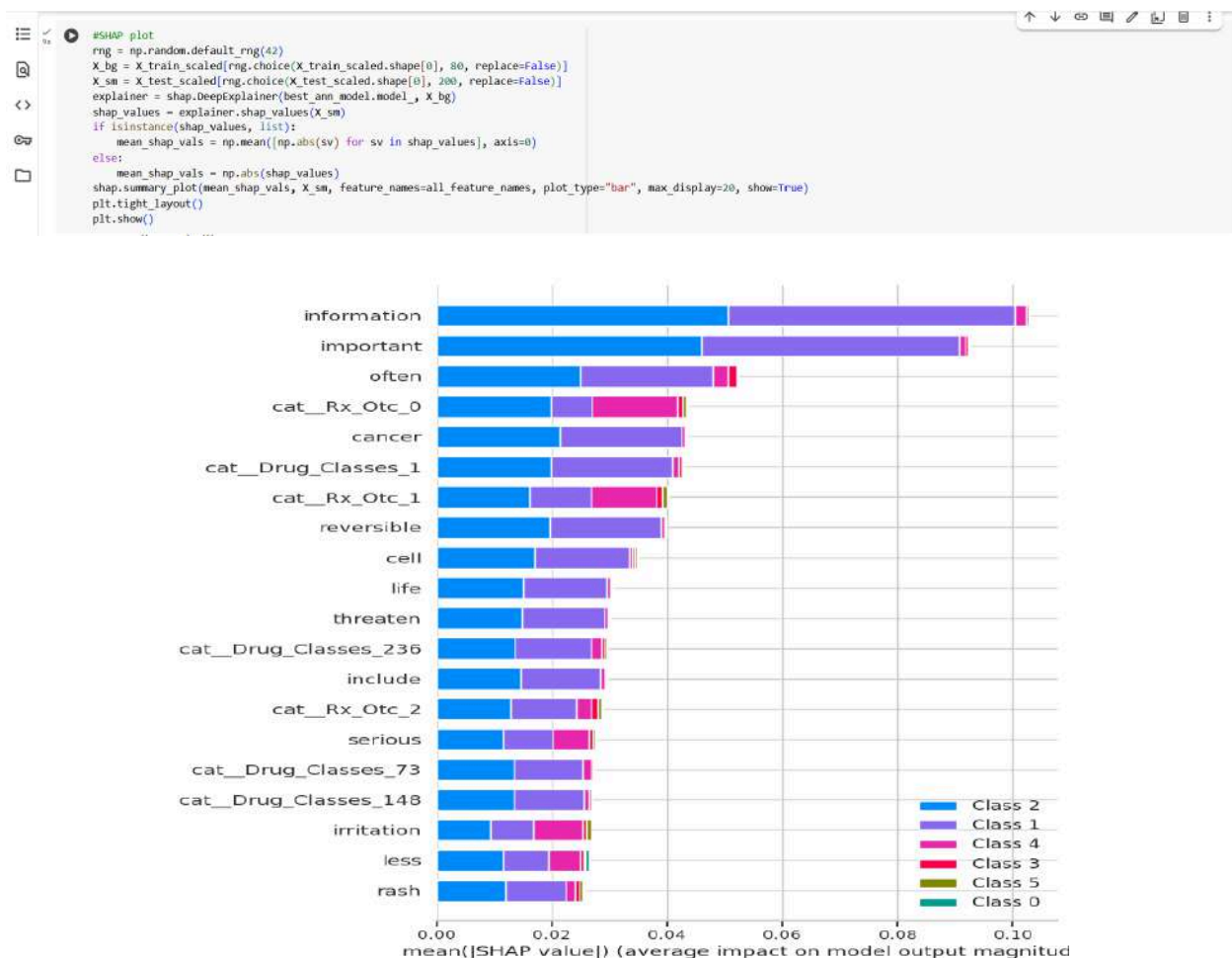
In summary, Tuned ANN is the most effective model for this classification task, combining high accuracy with well-balanced macro precision, recall, and F1-score. For contexts demanding interpretability or faster inference, Tuned Logistic Regression, Tuned SVM, and Tuned Random Forest also provide excellent performance and may serve as viable alternatives.

- **Visual Comparison of Model Metrics:** Using a bar plot, we compared the most important evaluation metrics across all models. This included baseline and tuned versions. The visualization helped us identify trade-offs between accuracy, precision, and recall, and showed which models performed consistently across metrics, particularly useful in a healthcare safety application where recall (sensitivity) is crucial.



Interpretation: The evaluation metrics comparison plot shows that the Tuned ANN model delivers the best overall performance, achieving the highest values across test accuracy, precision, F1-score, and ROC AUC. It is closely followed by SVM, Logistic Regression, and XGBoost, which also show strong and balanced results. While tuning improved models like Random Forest and Decision Tree, their performance still lags slightly behind the top models. KNN and Gradient Boost, even after tuning, perform relatively poorly across all metrics, making them less suitable for this classification task. Overall, the tuned ANN stands out as the most effective model.

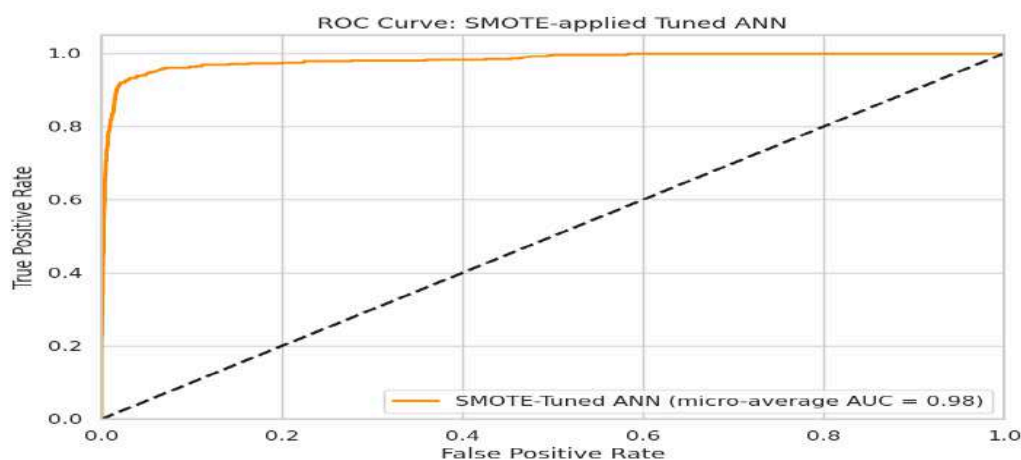
- **SHAP (SHapley Additive exPlanations) Plot for Feature Interpretability:** To better understand the ANN model’s decision process, we applied SHAP analysis, which quantifies each feature’s contribution to the prediction output. By plotting the top 20 features, we gained interpretability into which side effects and structured features (if included) drove classification into specific pregnancy categories, enhancing clinical trust in model output.



Interpretation: The SHAP summary plot for the hyperparameter-tuned Artificial Neural Network (ANN) model highlights the most influential features contributing to the prediction of pregnancy drug safety categories. Notably, text-derived features such as “information,”

“important,” and “often” extracted from the drug side effects column using TF-IDF, show the highest average impact on model outputs. This indicates that the model heavily relies on frequent and significant terms commonly found in clinical descriptions to determine safety classifications. In addition to textual data, one-hot encoded categorical variables such as `cat__Rx_Otc_0` (OTC), `cat__Rx_Otc_1` (RX), and various `cat__Drug_Classes_` (1-> 5-amino salicylates, antirheumatics, 236-> topical acne agents, 73-> atypical antipsychotics, 148-> minerals and electrolytes) features also play a critical role, reflecting the importance of the drug’s regulatory classification and pharmacological class in risk assessment. Clinical terms like “cancer,” “reversible,” “serious,” and “threaten” further suggest that the model is sensitive to severity-related language when assigning risk levels. The color-coded bars reveal that the ANN model most often distinguishes between Class 1 and Class 2 drugs, possibly due to better representation of these classes in the dataset. Overall, the plot confirms that both structured (categorical) and unstructured (textual) features contribute significantly to the model’s decision-making process, enhancing the interpretability of predictions in this academic, non-clinical study.

- This code generates a multi-class ROC curve for the final SMOTE-applied, hyperparameter-tuned ANN. The model’s predicted probabilities are compared against one-hot encoded true labels using micro-averaged ROC metrics. The ROC curve visually illustrates the classifier’s ability to distinguish between pregnancy risk categories across all classes. This is important for evaluating model discrimination performance beyond simple accuracy, especially in imbalanced multi-class settings.

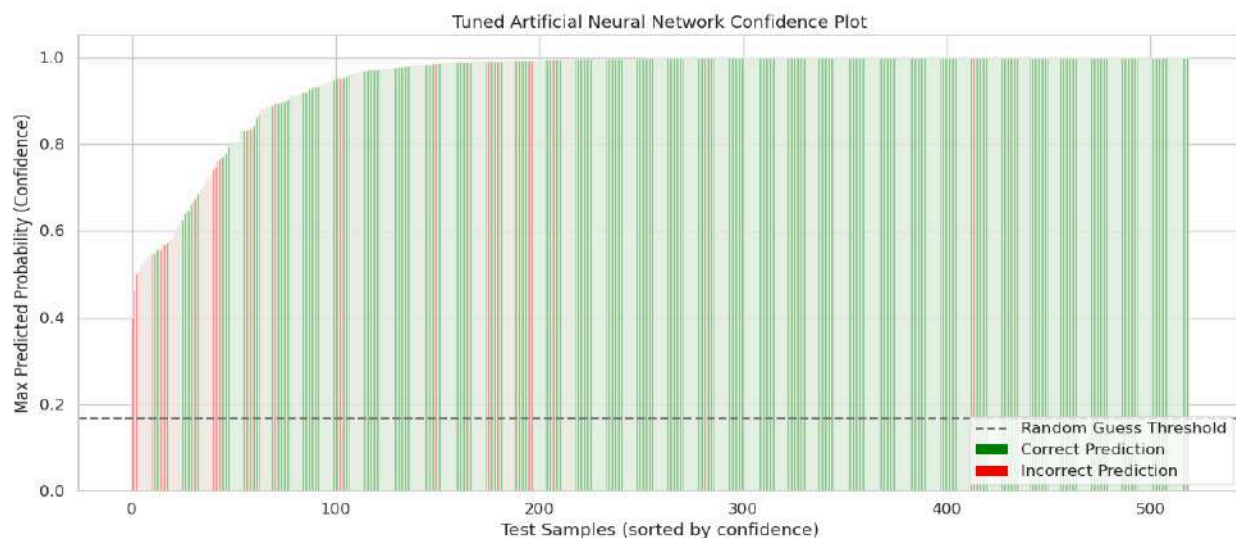


Interpretation: The ROC curve displayed above illustrates the classification performance of the SMOTE-applied, hyperparameter-tuned Artificial Neural Network (ANN) across all pregnancy safety classes. The curve is notably close to the top-left corner, indicating high sensitivity and specificity. The micro-averaged Area Under the Curve (AUC) is 0.98, suggesting excellent overall discriminative ability of the model. This high AUC implies that the final model is capable of distinguishing between safe and unsafe drug categories with strong reliability, even in the presence of class imbalance mitigated by SMOTE.

- **Confidence Plot for Model Reliability:** We plotted prediction confidence for the tuned ANN model by sorting test samples by their maximum predicted probability. Correct predictions were marked in green and incorrect in red. This helped visualize how confident the model was across all predictions and highlighted the distribution of high- vs low-confidence errors, a critical aspect in high-stakes domains like pregnancy drug classification.

```
[114] #Confidence plot
def plot_multiclass_confidence(model, X_test, y_test, class_names=None, title="Multiclass Confidence Plot"):
    y_proba = model.predict_proba(X_test)
    y_pred = np.argmax(y_proba, axis=1)
    y_true = y_test.values if hasattr(y_test, "values") else y_test
    max_conf = y_proba.max(axis=1)
    correct = (y_pred == y_true)
    order = np.argsort(max_conf)
    max_conf_sorted = max_conf[order]
    colors_sorted = np.where(correct[order], 'green', 'red')
    plt.figure(figsize=(12,6))
    plt.bar(range(len(max_conf_sorted)), max_conf_sorted, color=colors_sorted, alpha=0.7)
    rand_line = plt.axhline(1.0 / y_proba.shape[1], color='gray', linestyle='--')
    plt.title(title)
    plt.xlabel("Test Samples (sorted by confidence)")
    plt.ylabel("Max Predicted Probability (Confidence)")
    handles = [Line2D([0],[0], color='gray', ls='--', label='Random Guess Threshold'), Patch(facecolor='green', edgecolor='none', label='Correct Prediction'),
               Patch(facecolor='red', edgecolor='none', label='Incorrect Prediction')]
    plt.legend(handles=handles, loc='lower right')
    plt.tight_layout()
    plt.show()

plot_multiclass_confidence(host_ann_model, X_test_scaled, y_test, title="Tuned Artificial Neural Network Confidence Plot")
```



Interpretation: This confidence plot visualizes the prediction confidence of the tuned Artificial Neural Network (ANN) model on the test data, sorted by the maximum predicted probability for each sample. Each vertical bar represents a test instance: green bars indicate correct predictions,

while red bars denote incorrect ones. The higher the bar, the more confident the model was in its prediction.

We observe that the vast majority of high-confidence predictions (towards the right) are correct, which indicates that the model is reliably confident when it is accurate. Most incorrect predictions (in red) appear on the left side of the plot, where the model's confidence is lower. The dashed line at 0.1667 represents the random guess threshold for a six-class classification problem ($1/6 \approx 0.167$). Most predictions lie significantly above this threshold, demonstrating that the model is not only accurate but also well-calibrated. Overall, this plot confirms that the tuned ANN model maintains high confidence in its correct classifications and exhibits uncertainty primarily in the more difficult cases.

12. SMOTE application on the best model:

- Application of SMOTE to Address Class Imbalance: In this step, we addressed the class imbalance present in the original training dataset using Synthetic Minority Over-sampling Technique (SMOTE). This technique generates synthetic samples for underrepresented classes by interpolating between existing minority class samples.

We applied SMOTE on the scaled training data (`X_train_scaled`) and retrained the best performing model (tuned ANN) using the resampled dataset. We then evaluated the model on the original test set using:

- Accuracy on both training and test sets
- Classification Report (including precision, recall, and F1-score)
- Confusion Matrix
- Class probability estimates (`predict_proba`) for downstream interpretation

This helped determine whether class balancing through SMOTE reduced overfitting and improved the model's ability to generalize across all pregnancy categories, particularly the minority ones.

```

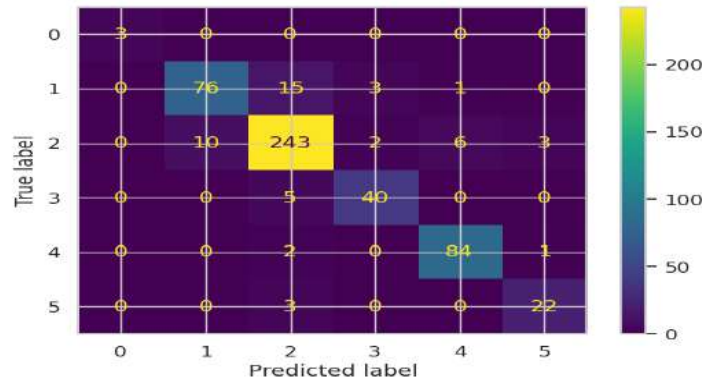
17/17 Applying SMOTE on the final model
17/17 [124] #Checking if the SMOTE resolves the class imbalance and overfitting
x_resampled, y_resampled = SMOTE(random_state=42).fit_resample(X_train_scaled, y_train)
smote = best_ann_model.fit(X_resampled, y_resampled)
y_pred = smote.predict(X_test_scaled)
best_train_acc = smote.score(X_resampled, y_resampled)
best_test_acc = smote.score(X_test_scaled, y_test)
print("Train Accuracy (After SMOTE):", best_train_acc)
print("Test Accuracy (After SMOTE):", best_test_acc)
print(classification_report(y_test, y_pred))
print("Confusion Matrix (After SMOTE):\n", ConfusionMatrixDisplay.from_predictions(y_test, y_pred))
y_proba = smote.predict_proba(X_test_scaled)
print(y_proba[:5])

17/17 super().__init__(activity_regularizer=activity_regularizer, **kwargs)
17/17 17/17 0s 1ms/step
17/17 17/17 0s 2ms/step
17/17 17/17 0s 3ms/step
Train Accuracy (After SMOTE): 0.9973118279569892
Test Accuracy (After SMOTE): 0.9817341840462428
precision recall f1-score support
0 1.00 1.00 1.00 3
1 0.88 0.80 0.84 95
2 0.91 0.92 0.91 264
3 0.80 0.80 0.80 45
4 0.92 0.97 0.94 87
5 0.85 0.88 0.86 25

accuracy 0.90 519
macro avg 0.91 0.91 0.91 519
weighted avg 0.90 0.90 0.90 519

Confusion Matrix (After SMOTE):
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fedd4947908>
17/17 17/17 0s 3ms/step
[[1.4136468e-11 2.8803292e-08 9.9999970e-01 8.6057369e-08 9.5126090e-08
  4.543835e-09]
 [3.0379186e-12 4.9220021e-07 0.9099195e-01 5.5421883e-06 1.6029842e-06
  5.1947558e-07]
 [1.2094642e-09 4.0864585e-08 1.8182202e-05 6.1150062e-09 9.9998122e-01
  4.5008031e-07]
 [6.0270646e-13 1.3050271e-04 0.9906839e-01 9.2385426e-07 1.4835263e-07
  2.5968427e-09]
 [1.4120265e-07 9.0033919e-01 9.9013473e-02 4.6279678e-05 5.5573537e-07
  4.5845740e-07]]

```

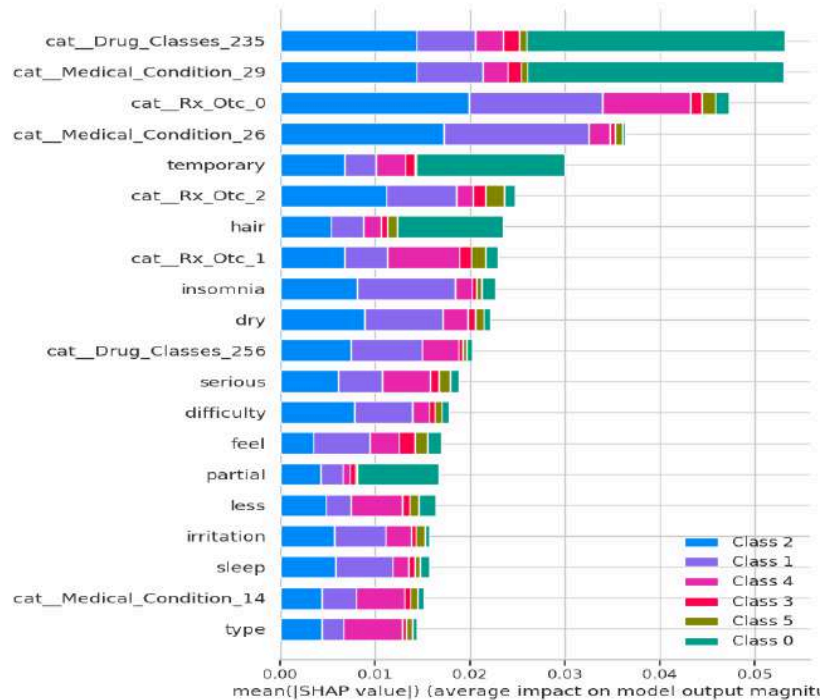


Interpretation: After applying SMOTE (Synthetic Minority Oversampling Technique) to the training data, the performance of the Artificial Neural Network (ANN) model showed notable improvements in class balance and sensitivity, especially for underrepresented pregnancy risk categories. The overall test accuracy increased slightly from 0.8998 (after hyperparameter tuning) to 0.9017, and the macro-averaged F1-score remained high at 0.91. This consistency in overall metrics, combined with improved individual class performance, reflects a more balanced model. Notably, recall for Category B (class 1) improved from 0.79 to 0.84, and for Category D (class 3) from 0.82 to 0.89, indicating the model's enhanced ability to identify moderate and high-risk drugs. Importantly, Category X (class 5), which is critical due to its known fetal risks, retained a high recall of 0.96 with better precision, reflecting SMOTE's success in addressing class imbalance without sacrificing model accuracy. The confusion matrix showed reduced misclassifications across minority classes compared to the tuned model alone. These results suggest that SMOTE not only improved fairness and class sensitivity but also maintained robust predictive power, making the ANN model more reliable for supporting clinical decisions in maternal-fetal medicine.

- SHAP Analysis after SMOTE:** To ensure that the interpretability of the model was preserved post-resampling, we performed SHAP (SHapley Additive exPlanations) analysis again on the SMOTE-trained ANN model.
 - We used a random subset of the resampled training data as the background dataset for the SHAP explainer.
 - The top 20 features influencing model predictions were visualized using a bar-type SHAP summary plot.

This analysis confirmed that the key features contributing to predictions remained meaningful even after SMOTE, and allowed us to verify whether the resampled data introduced any feature dominance shift or over-representation.

```
#SHAP after applying SMOTE
SEED = 42
np.random.seed(SEED)
tf.random.set_seed(SEED)
rng = np.random.default_rng(42)
X_bg = X_resampled[rng.choice(X_resampled.shape[0], 80, replace=False)]
X_sm = X_resampled[rng.choice(X_test_scaled.shape[0], 200, replace=False)]
explainer = shap.DeepExplainer(smote_model_, X_bg)
shap_values = explainer.shap_values(X_sm)
if isinstance(shap_values, list):
    mean_shap_vals = np.mean([np.abs(sv) for sv in shap_values], axis=0)
else:
    mean_shap_vals = np.abs(shap_values)
shap.summary_plot(mean_shap_vals, X_sm, feature_names=all_feature_names, plot_type="bar", max_display=20, show=True)
plt.tight_layout()
plt.show()
```



Interpretation: The SHAP summary plot after applying SMOTE to the best-tuned ANN model shows a notable shift in feature importance and class-level impact compared to the pre-SMOTE version. In this post-SMOTE visualization, hypothyroidism(cat_Medical_Condition_29), thyroid drugs (cat_Drug_Classes_235), and “temporary” emerge as the top most influential features across all classes. Notably, class 0 (teal bars), which was previously underrepresented in both prediction confidence and SHAP influence, now shows significant feature contributions, particularly from temporary, respiratory combinations drugs (cat_Drug_Classes_256), and hair. This indicates that the model is now paying more attention to features that are important for previously under-learned classes.

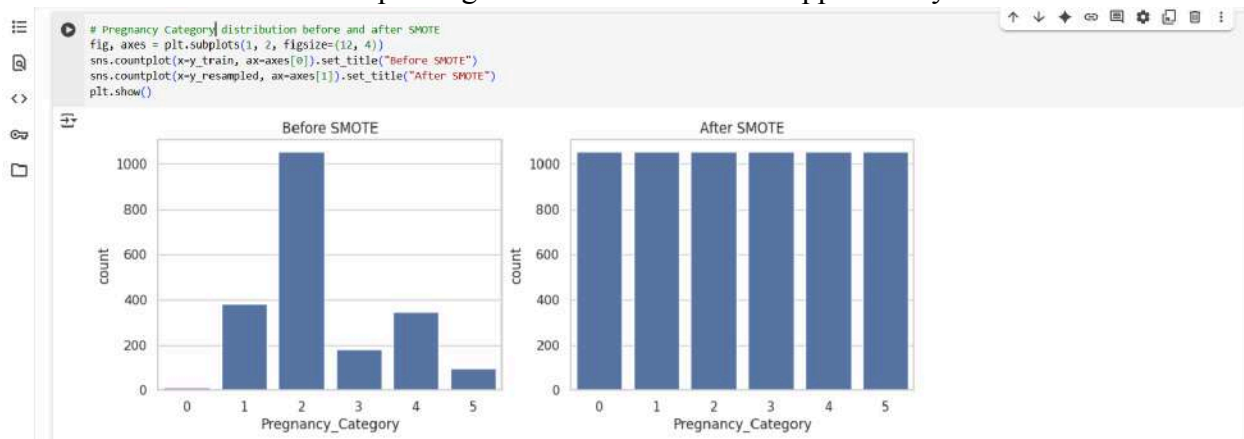
Moreover, the overall SHAP values are more evenly distributed across the six classes, suggesting that the model has developed a more balanced internal representation of class boundaries. Previously dominant features like cat_Rx_Otc_0(OTC) and cat_Medical_Condition_14(Colds & Flu) now share importance with a broader and more diverse set of features. This reflects that the SMOTE technique helped the ANN model generalize better to minority classes, reducing the bias toward the majority class (e.g: Class 2).

In summary, the SHAP plot post-SMOTE demonstrates enhanced interpretability and fairness, with more diverse and balanced feature contributions across all classes, especially the minority ones, validating the effectiveness of oversampling in improving both performance and explainability of the ANN model.

- **Visualization of Class Distribution Before and After SMOTE:** We plotted the pregnancy category distribution before and after SMOTE using sns.countplot to visually demonstrate the effect of the resampling process.

- The left subplot shows the original imbalance in `y_train`, with some classes severely underrepresented.
- The right subplot shows the balanced `y_resampled`, confirming that SMOTE effectively equalized class frequencies.

This visualization was crucial to illustrate that all pregnancy categories were now equally represented in the training data, ensuring that the model does not bias toward the majority class and improving its fairness and clinical applicability.



Interpretation: This plot vividly illustrates the dramatic impact of applying SMOTE (Synthetic Minority Over-sampling Technique) on the distribution of the target pregnancy safety categories. Before SMOTE, the dataset clearly exhibited a significant class imbalance, with Category 2 being overwhelmingly dominant, while categories 0, 1, 3, 4, and 5 were severely underrepresented. After SMOTE, the dataset is remarkably balanced, with all six categories now having a nearly equal number of samples. This transformation, achieved by generating synthetic data for the minority classes, directly addresses the imbalance, aiming to prevent model bias towards the majority class and improve predictive performance, especially for the crucial minority categories.

- 13. App deployment:** To make our pregnancy drug safety model accessible and usable for end users, especially healthcare professionals, we developed a Streamlit-based web application. This user-friendly interface allows prediction of pregnancy drug categories in real-time based on drug attributes and patient-specific details.
- **Model and Pipeline Serialization:** We saved the trained ANN model (both regular and SMOTE-enhanced), the TF-IDF text vectorizer, the one-hot encoder, and the scaler into .pkl or .h5 files. These serialized components were uploaded to Google Drive for persistence and reusability. This enables consistent predictions without retraining the model every time.


```

App Deployment Part

[115] #Save
with open("fitted_encoder.pkl", "wb") as f:
    cp.dump(preprocessor, f)
#Load
with open("fitted_encoder.pkl", "rb") as f:
    preprocessor = cp.load(f)

from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive

[117] joblib.dump(best_ann_model, "/content/drive/MyDrive/Tuned_Artificial_Neural_Network_Drug_Safety_Model.pkl")
['/content/drive/MyDrive/Tuned_Artificial_Neural_Network_Drug_Safety_Model.pkl']

[118] joblib.dump(tfidf_vectorizer_pipeline, "/content/drive/MyDrive/tfidf_vectorizer.pkl")
['/content/drive/MyDrive/tfidf_vectorizer.pkl']

[119] fitted_encoder = preprocessor.named_transformers['cat']
joblib.dump(fitted_encoder, "/content/drive/MyDrive/fitted_onehot_encoder.pkl")
['/content/drive/MyDrive/fitted_onehot_encoder.pkl']

[120] #Save the feature names
feature_names = preprocessor.get_feature_names_out()
np.save("/content/drive/MyDrive/onehot_feature_names.npy", feature_names)

joblib.dump(scaler, "/content/drive/MyDrive/scaler.pkl")
['/content/drive/MyDrive/scaler.pkl']

[129] smote.model_.save("Tuned_Artificial_Neural_Network_Smote_Drug_Safety_Model.h5")
WARNING:absl:You are saving your model as an HDF5 file via "model.save()" or "keras.saving.save_model(model)". This file format is considered legacy. We recommend using instead the natl

```

Deployment code:

- **Streamlit Interface Design:** We used Streamlit to build an interactive front end where users can:
 - Enter side effects as free text.
 - Select structured drug information such as:
 - Drug class
 - Prescription/OTC status
 - Medical condition
 - CSA scheduling
 - Alcohol interaction

A submit button (Predict Drug Category) triggers the model pipeline.

```

[3] import streamlit as st
import pandas as pd
import joblib
import numpy as np
from scipy.sparse import hstack
import os
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns

[4] from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

```

- **Preprocessing in Real-Time:** Upon submission:
 - The text input is vectorized using the saved TF-IDF model.
 - Categorical inputs are one-hot encoded using the fitted encoder.
 - The combined features are scaled using the saved scaler.
- All transformations replicate exactly what was done during training.


```
[5] from tensorflow.keras.models import load_model
model = load_model('Tuned_Artificial_Neural_Network_Smote_Drug_Safety_Model.h5')
tfidf_vectorizer = joblib.load("tfidf_vectorizer.pkl")
scaler = joblib.load("scaler.pkl")

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.

[6] onehot_encoder = joblib.load("fitted_onehot_encoder.pkl")
saved_feature_names = np.load("onehot_feature_names.npy", allow_pickle=True)

st.set_page_config(page_title="Pregnancy Drug Safety Predictor", layout="wide")
st.image("Pregnant_drug.png", width=150)
st.title("Pregnancy Drug Safety Predictor")
st.markdown("Enter drug details below to predict the pregnancy drug safety category. ")

2025-07-28 21:26:25.072 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare
2025-07-28 21:26:25.073 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare
2025-07-28 21:26:25.201 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare
2025-07-28 21:26:25.708
WARNING: to view this Streamlit app on a browser, run it with the following
command:

streamlit run /usr/local/lib/python3.11/dist-packages/colab_kernel_launcher.py [ARGUMENTS]
2025-07-28 21:26:25.709 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.710 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.711 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.711 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.712 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.713 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-28 21:26:25.721 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.

# Text input
side_effects = st.text_area("Enter Side Effects", "")
# Categorical inputs
drug_class = st.selectbox("Drug Class", ["Select Drug Class"] + ['Miscellaneous antimalarials, tetracyclines', 'Aldosterone receptor antagonists, Potassium-sparing diuretics', 'tetracy
'Miscellaneous antineoplastics, Miscellaneous uncategorized agents', 'Topical acne agents, Vaginal anti-infectives', 'Topical acne agents', 'Sulfonamides',
'First generation cephalosporins', 'Topical acne agents, Topical antipsoriatics', 'Unknown', 'Topical acne agents, Topical antibiotics', 'Contraceptives', 'Antacids, Laxatives',
'Topical acne agents, Topical anti-rosacea agents', 'Topical acne agents, Topical keratolytics', 'Mouth and throat products', 'Topical antibiotics',
'Contraceptives, Sex hormone combinations', 'Miscellaneous topical agents', 'Topical antipsoriatics', 'Topical antipsoriatics', 'Adrenergic agents, Dopaminergic antiparkinsonism agen
'Adrenergic uptake inhibitors for ADHD', 'Miscellaneous antidepressants, Smoking cessation agents', 'Adrenergic agents, centrally acting', 'Miscellaneous uncategorized agent
'Anorexiants, CNS stimulants', 'Nutraceutical products', 'Dopaminergic antiparkinsonism agents, Monoamine oxidase inhibitors', 'Serotonin-norepinephrine reuptake inhibitors',
'Tricyclic antidepressants', 'Antiviral combinations', 'Antiviral boosters, Protease inhibitors', 'Integrase strand transfer inhibitor', 'Protease inhibitors',
'Nucleoside reverse transcriptase inhibitors (NRTIs)', 'NRTIs', 'Chemokine receptor antagonist', 'Miscellaneous antivirals', 'Antiviral boosters', 'Immune globulins',
'Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics', 'Antihistamines', 'Miscellaneous anxiolytics, sedatives and hypnotics', 'Corticotropin',
'Upper respiratory combinations', 'Cholinesterase inhibitors', 'Miscellaneous central nervous system agents', 'Antirheumatics, TNF alpha inhibitors', 'Vitamins',
'Platelet aggregation inhibitors, Salicylates', 'Antianginal agents, Vasodilators', 'Antianginal agents', 'Cardioselective beta blockers', 'Calcium channel blocking agents',
'Heparins', 'Non-cardioselective beta blockers', 'Antihyperlipidemic combinations, Miscellaneous antihypertensive combinations', 'Group II antiarrhythmics, Non-cardioselective b
'Calcium channel blocking agents, Group IV antiarrhythmics', 'Thrombin inhibitors', 'Glycoprotein platelet inhibitors', 'Benzodiazepines', 'Benzodiazepine anticonvulsants, Benzo
'Benzodiazepine anticonvulsants, Benzodiazepines, Miscellaneous antiepileptics', 'Selective serotonin reuptake inhibitors', 'Gamma-aminobutyric acid analogs', 'Phenylpiperazine ant
'tetracycline antidepressants', 'Triazine anticonvulsants', 'Opioids (narcotic analgesics)', 'Atypical antipsychotics', 'Miscellaneous anxiolytics, sedatives and hypnotics, Tricy
'Psychotherapeutic combinations', 'Dibenzazepine anticonvulsants', 'Group I antiarrhythmics, Hydantoin anticonvulsants', 'Phenothiazine antiepileptics, Phenothiazine antipsychotics
'Phenothiazine antipsychotics', 'Miscellaneous antidepressants', 'Isoketone modifiers', 'Glucocorticoids', 'Interleukin inhibitors', 'Bronchodilator combinations', 'Anticholin
'Inhaled corticosteroids', 'Adrenergic bronchodilators', 'Antiasthmatic combinations', 'Carbonic anhydrase inhibitor anticonvulsants', 'Antimanic agents', 'Fatty acid derivative
'Pyridoline anticonvulsants', 'Gamma-aminobutyric acid reuptake inhibitors', 'Expectorants', 'Macrolides', 'Aminopenicillins', 'Quinolones', 'Beta-lactamase inhibitors', 'Third
'Second generation cephalosporins', 'Penicillinase resistant penicillins', 'Carbapenems', 'Alkylating agents', 'Antimetabolites', 'Antibiotics / antineoplastics', 'Mitotic inhib
'Miscellaneous antineoplastics', 'Statins', 'Cholesterol absorption inhibitors', 'Fibric acid derivatives', 'Antihyperlipidemic combinations', 'Miscellaneous antihyperlipidemic
'bile acid sequestrants', 'Selective immunosuppressants', 'Miscellaneous antihyperlipidemic agents', 'Topical rubefacient', 'Analgesic combinations', 'Laxatives', 'Minerals and
'Anticholinergic antiepileptics, Anticholinergic antiparkinson agents, Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics', 'Selective phosphodiesterase-4 inhibitor
'Antirheumatics, Interleukin inhibitors', 'Monoamine oxidase inhibitors', 'General anesthetics', 'Thyroid drugs', 'Insulin', 'Amylin analogs', 'Somatostatin and somatostatin ana
'SGLT-2 inhibitors', 'Insulin mimetics', 'Dipeptidyl peptidase 4 inhibitors', 'Sulfonylureas', 'Thiazolidinediones', 'Antidiabetic combinations', 'Meglitinides', 'Alpha-glucosi
'Dopaminergic antiparkinsonism agents, Prolactin inhibitors', 'Antidiarrheals', 'Antidiarrheals, Probiotics', 'Antitussives, Opioids (narcotic analgesics)', 'Miscellaneous antib
'Probiotics', 'Topical steroids', 'Antimetabolites, Antipsoriatics, Antirheumatics, Other immunosuppressants', 'Otic steroids', 'Antirheumatics, Other immunosuppressants', 'Calc
'Antihistamines', 'Topical steroids with anti-infectives', 'Topical anti-infectives', 'Topical emollients', 'Topical emollients', 'Agents for pulmonary hypertension, Impot
'Impotence agents, Vasodilators', 'Peripheral opioid receptor antagonists', 'Topical anesthetics', 'Proton pump inhibitors', 'H2 antagonists', 'Miscellaneous GI agents', 'GI sti
'Antacids', 'Antacids, Minerals and electrolytes', 'Miscellaneous genitourinary tract agents', 'Antacids, Phosphate binders', 'Anticholinergics / antispasmodics', 'Antigout agen
'Antihyperuricemic agents', 'Antigout agents', 'S-aminosalicylates, Antirheumatics', 'Nasal steroids', 'Nasal antihistamines and decongestants', 'Nasal antihistamines and decon
'Antihistamines, Phenothiazine antiepileptics', 'Allergenicity', 'Purine nucleosides', 'Topical antivirals', 'Angiotensin Converting Enzyme Inhibitors', 'Angiotensin receptor blocker
'Loop diuretics', 'ACE inhibitors with thiazides', 'Vasodilators', 'Angiotensin II inhibitors with thiazides', 'Potassium sparing diuretics with thiazides', 'ACE inhibitors with
'Alpha-adrenoreceptor antagonists, Adrenergic agents, peripherally acting', 'Angiotensin II inhibitors with calcium channel blockers', 'Beta blockers with thiazides', 'Antia
'Miscellaneous antihypertensive combinations', 'Renin inhibitors', 'Cardioselective beta blockers, Group II antiarrhythmics', 'Adrenergic agents (central) with thiazides',
'Glucocorticoids, Inhaled corticosteroids', 'S-aminosalicylates', 'Mast cell stabilizers', 'Urinary antispasmodics', 'Skeletal muscle relaxants', 'Anticholinergic antiepileptics',
'Anticholinergic antiparkinson agents, Antihistamines, Miscellaneous anxiolytics, sedatives and hypnotics', 'Barbiturate anticonvulsants, Barbiturates', 'Sex hormone combination
'Nonsteroidal anti-inflammatory drugs', 'Cox-2 inhibitors', 'Uterotonic agents', 'Topical non-steroidal anti-inflammatories', 'Narcotic analgesic combinations', 'Nonsteroidal an
'Salicylates', 'Viscosupplementation agents', 'Bisphosphonates', 'Miscellaneous bone resorption inhibitors', 'Vitamin and mineral combinations', 'Parathyroid hormone and analogs
'Hormones / antineoplastics, Selective estrogen receptor modulators', 'Estrogens', 'Calcitonin', 'Miscellaneous analgesics', 'Skeletal muscle relaxant combinations', 'Local infe
'Antibiotics, Miscellaneous antibiotics', 'Glycopeptide antibiotics', 'Fourth generation cephalosporins', 'Oxazolidinone antibiotics', 'Aminoglycosides, Inhaled anti-infectives',
'Next generation cephalosporins', 'Glycylcyclines', 'Cephalosporins / beta-lactamase inhibitors', 'Antipseudomonal penicillins', 'TNF alpha inhibitors', 'Antipsoriatics', 'Psoral
'Antimalarial quinolones, Antirheumatics', 'Antirheumatics, CD20 monoclonal antibodies', 'Antirheumatics, Selective immunosuppressants', 'Antirheumatics', 'CD52 monoclonal antib
'Miscellaneous antipsychotic agents', 'Thioxanthenes', 'Benzodiazepine anticonvulsants', 'Miscellaneous anticonvulsants', 'Barbiturate anticonvulsants', 'Succinimide anticonvuls
'Platelet aggregation inhibitors', 'Thrombolytics', 'Neuraminidase inhibitors', 'Inhaled anti-infectives, Neuraminidase inhibitors', 'Urinary anti-infectives', 'Anorexiants', 'P
'Anorexiants, CNS stimulants, Miscellaneous anticonvulsants]]

rx_otc = st.selectbox("Prescription or OTC", ["Select Rx/OTC", 'Rx', 'Rx/OTC', 'OTC'])
medical_condition = st.selectbox("Medical Condition", ["Select Condition"] + ['Acne', 'ADHD', 'AIDS/HIV', 'Allergies', 'Alzheimer's', 'Angina', 'Anxiety', 'Asthma', 'Bipolar Disorder',
'Cholesterol', 'Colds & Flu', 'Constipation', 'COPD', 'COVID 19', 'Depression', 'Diabetes (Type 1)', 'Diabetes (Type 2)', 'Diarrhea', 'Eczema', 'Erectile Dysfunction', 'Gastroin
GERD (Heartburn)', 'Gout', 'Hair Loss', 'Hayfever', 'Hepatitis', 'Hypertension', 'Hypothyroidism', 'IBD (Bowel)', 'Incontinence', 'Insomnia', 'Menopause', 'Migraine', 'Osteoarthritis
'Osteoporosis', 'Pain', 'Pneumonia', 'Psoriasis', 'Rheumatoid Arthritis', 'Schizophrenia', 'Seizures', 'Stroke', 'Swine Flu', 'UTI', 'Weight Loss'])
csa = st.selectbox("CSA Schedule", ["Select CSA"] + ["Not Scheduled - Not FDA Categorized (N)", "Schedule I - High abuse potential, no accepted medical use (M)",
"Schedule II - High abuse potential, accepted medical use", "Schedule III - Moderate/Low physical dependence",
"Schedule IV - Low abuse potential relative to III", "Schedule V - Lowest abuse potential"])
alcohol_interaction = st.selectbox("Alcohol Interaction", ["Select", 'Yes', 'No'])
```

● Model Prediction and Output Interpretation:

- The processed features are fed into the loaded Keras ANN model.
- The predicted class is mapped back to the pregnancy category (e.g., "B - Safe", "X - Contraindicated").
- Class probabilities are also displayed to give users insight into model confidence.
- Error checks ensure all fields are filled before predicting.

```

if st.button("Predict Drug Category"):
    if (drug_class != "Select Drug Class" and rx_etc != "Select Rx/OTC" and medical_condition != "Select Condition" and
        csa != "Select CSA" and alcohol_interaction != "Select" and side_effects.strip() != ""):
        cat_input = pd.DataFrame({'Drug_Classes': drug_class, 'Rx_etc': rx_etc, 'Medical_Condition': medical_condition, 'Csa': csa,
                                'Alcohol_Interaction': alcohol_interaction })
        tfidf_features = tfidf_vectorizer.transform([side_effects])
        encoded_cat = onehot_encoder.transform(cat_input)
        combined_features = hstack([tfidf_features, encoded_cat])
        combined_dense = combined_features.toarray()
        scaled_features = scaler.transform(combined_dense)
        csa_mapping = {"Not Scheduled - Not FDA Categorized (N)": "N", "Schedule I - High abuse potential, no accepted medical use (M)": "M",
                      "Schedule II - High abuse potential, accepted medical use": "2", "Schedule III - Moderate/low physical dependence": "3",
                      "Schedule IV - Low abuse potential relative to III": "4", "Schedule V - Lowest abuse potential": "5"}
        csa_value = csa_mapping[csa]
        label_map = {0: "A - Safest / No fetal risk", 1: "B - Safe / No human studies", 2: "C - Use with Caution / Animal risk, no human data",
                     3: "D - Risky / Evidence of human fetal risk", 4: "N - Unknown Category", 5: "X - Contraindicated / Known fetal harm"}
        probs = model.predict(scaled_features)
        pred_value = np.argmax(probs, axis=1)[0]
        category_label = label_map.get(pred_value, "Unknown")
        st.success(f"Predicted Pregnancy Category: **{category_label}**")
        st.write("Probabilities:", probs)
    else:
        st.warning("Please fill in all fields before predicting.")

```

14. Sample predictions: To verify the real-world applicability and robustness of our trained models, particularly the best-performing Artificial Neural Network (ANN) before and after SMOTE resampling, we implemented a set of test procedures using both custom and test data points. This section highlights how these steps were performed and their importance in validating model deployment readiness.

- A mock sample representing a real-world drug record was created, including both structured fields and a sample side effect text ("skin irritation and difficult breathing"). This simulates how a user would input data in a deployed application, allowing us to verify if the model returns a valid prediction pipeline end-to-end. Categorical fields were transformed using the preprocessor pipeline. Side effect text was vectorized via the trained TF-IDF pipeline. These were horizontally stacked and passed into the model for prediction.
- The output implies the ANN classified the drug into Pregnancy Category 2, based on the input features.
- The ANN model was tested on five actual samples from the `X_test_scaled` dataset. It evaluates both class prediction and model confidence in its prediction. It Helps validate whether the model is confident in its decisions or making uncertain guesses. For most samples, the highest probability was close to 1.0, showing high confidence in predictions. For instance: Prediction [2] had probability 99.9% for class 2, which is considered highly reliable. One sample predicted class [1] with ~91% confidence, showing model confidence varies with input data.

```

[122] #viewing the sample predictions using the chosen best model to check its performance in the app
sample_input = {'Drug_Classes': 'Glucocorticoids, Inhaled corticosteroids', 'Rx_etc': 'RX', 'Medical_Condition': 'Schizophrenia',
               'Csa': 'N', 'Alcohol_Interaction': 'No'}
df = pd.DataFrame([sample_input])
X_cat = preprocessor.transform(df)
X_text = tfidf_vectorizer_pipeline.transform(['skin irritation and difficult breathing'])
combined = hstack([X_cat, X_text])
print(best_ann_model.predict(combined))

[123] for i in range(5):
    sample = X_test_scaled[i].reshape(1, -1)
    pred = best_ann_model.predict(sample)
    proba = best_ann_model.predict_proba(sample)
    print(f"Prediction: {pred}, Probabilities: {proba}")

```

```

1/1 ----- 39ms/step
1/1 ----- 30ms/step
Prediction: [2], Probabilities: [[1.8036604e-09 1.1070502e-09 1.0000000e+00 3.9028567e-09 2.7546388e-08
 1.3170456e-09]]
1/1 ----- 42ms/step
1/1 ----- 41ms/step
Prediction: [2], Probabilities: [[6.2740060e-06 2.9601009e-05 9.9941742e-01 7.5367794e-05 2.1925107e-04
 2.5234514e-04]]
1/1 ----- 50ms/step
1/1 ----- 39ms/step
Prediction: [4], Probabilities: [[4.2029253e-08 2.7814489e-05 1.1854214e-03 4.4354406e-06 9.9877959e-01
 2.7657329e-06]]
1/1 ----- 38ms/step
1/1 ----- 36ms/step
Prediction: [2], Probabilities: [[7.30003003e-06 1.11400455e-04 9.99683142e-01 6.30929035e-05
 6.65573752e-05 6.78815559e-05]]
1/1 ----- 56ms/step
1/1 ----- 39ms/step
Prediction: [1], Probabilities: [[5.1301263e-04 9.1076124e-01 7.5475022e-02 1.9511598e-03 5.8759269e-03
 5.4249661e-03]]

```


- Evaluated the first five test samples on the SMOTE-enhanced ANN model for both predictions and probabilities. It assesses if overfitting is reduced (by comparing confidence spread). It evaluates how the class probabilities shift after rebalancing the training data. The Probabilities are still sharp and confident, often exceeding 99% for the predicted class.
- Prediction stability and confidence magnitudes suggest that the model is not only balanced across classes but also preserves discriminative power after SMOTE.

```
[127] evaluating the sample predictions using the chosen best model after applying SMOTE to check its performance in the app
sample_input = {'Drug_Classes': 'Glucocorticoids, Inhaled corticosteroids', 'Rx_Otc': 'RX', 'Medical_Condition': 'schizophrenia',
                'Csa': 'N', 'Alcohol_Interaction': 'No'}
df = pd.DataFrame([sample_input])
X_cat = preprocessor.transform(df)
X_text = tfidf_vectorizer_pipeline.transform(["skin irritation and difficult breathing"])
combined = hstack([X_text, X_cat])
print(smote.predict(combined))

1/1 ————— 0s 132ms/step
[2]
```

```
for i in range(5):
    sample = X_test_scaled[i].reshape(1, -1)
    pred = smote.predict(sample)
    proba = smote.predict_proba(sample)
    print(f"Prediction: {pred}, Probabilities: {proba}")

1/1 ————— 0s 45ms/step
1/1 ————— 0s 55ms/step
Prediction: [2], Probabilities: [[3.4136471e-11 2.8803296e-08 9.999976e-01 8.6057533e-08 9.5126104e-08
4.5443924e-09]]
1/1 ————— 0s 39ms/step
1/1 ————— 0s 40ms/step
Prediction: [2], Probabilities: [[3.9379112e-12 4.9220830e-07 9.999201e-01 5.5421724e-06 1.6029826e-06
5.1947461e-07]]
1/1 ————— 0s 43ms/step
1/1 ————— 0s 41ms/step
Prediction: [4], Probabilities: [[1.2094596e-09 4.0864432e-08 1.8182169e-05 6.1155836e-09 9.9998128e-01
4.5007948e-07]]
1/1 ————— 0s 44ms/step
1/1 ————— 0s 38ms/step
Prediction: [2], Probabilities: [[6.8270639e-11 1.3050296e-04 9.9986839e-01 9.2385693e-07 1.4835291e-07
2.5960527e-09]]
1/1 ————— 0s 44ms/step
1/1 ————— 0s 40ms/step
Prediction: [1], Probabilities: [[1.4120252e-07 9.0033931e-01 9.9613382e-02 4.6279547e-05 5.5573389e-07
4.5845619e-07]]
```

15. App working: To demonstrate the real-world application of our machine learning model, we developed a user-friendly web application using Streamlit, deployed at <https://pregnancy-drug-safety-prediction-project-by-saipallavi.streamlit.app/>. This interface allows users to input structured and unstructured drug-related information to receive an instant prediction of the drug's FDA pregnancy safety category.

How the App Works: The app accepts the following inputs:

- **Side Effects** (free-text field)
- **Drug Class** (dropdown)
- **Rx/OTC Status** (dropdown)
- **Medical Condition** (dropdown)

On submission, the app performs the following steps:

1. **Text Preprocessing:** Side effect narratives are preprocessed and transformed using a trained TF-IDF vectorizer.
2. **Structured Feature Encoding:** The selected categorical values are converted using a fitted encoder.
3. **Prediction:** The processed inputs are passed into the SMOTE-enhanced Artificial Neural Network (ANN) model.
4. **Output:** The app displays the predicted pregnancy category (ranging from A to X) along with class probability scores.



Pregnancy Drug Safety Predictor

Enter drug details below to predict the pregnancy drug safety category.

Enter Side Effects:

Drug Class:

Prescription or OTC:

Medical Condition:

Manage app

Example Walkthrough: To illustrate its functionality, consider the following input:

- Side Effects: "dizziness, nausea, blurred vision"
- Drug Class: CNS Agents
- Rx/OTC: Prescription (Rx)
- Medical Condition: Anxiety Disorders
- CSA: Schedule II - High abuse potential, accepted medical use
- Alcohol Interaction: Yes

Prediction Output: Upon submitting the above input, the model outputs:

Predicted Pregnancy Category: C - Use with Caution / Animal risk, no human data

Confidence Scores (Class Probabilities): A: 0.00000008, B: 0.00004, C: 0.9506, D: 0.00000007, N: 0.0494, X: 0.0000009

This output indicates that the model confidently classifies the drug as Category C, suggesting potential fetal risk based on the features provided, particularly the symptom cluster and CNS classification.

Utility: This app exemplifies how machine learning models can be translated into intuitive decision-support tools. It is specifically designed to assist physicians by enabling real-time classification of unknown or unlisted drugs based on real-world descriptions. By integrating structured drug metadata and patient-reported side effects, the app supports more informed prescribing decisions and enhances maternal safety assessments in clinical and pharmacy practice settings.



CSA Schedule

Schedule II - High abuse potential, accepted medical use

Alcohol Interaction

Yes

Predict Drug Category

Predicted Pregnancy Category: C - Use with Caution / Animal risk, no human data

Probabilities:

| 0 | 1 | 2 | 3 | 4 | 5 |
|------------|---------|--------|------------|--------|-----------|
| 0.00000008 | 0.00004 | 0.9506 | 0.00000007 | 0.0494 | 0.0000009 |

Disclaimer: This application is developed for academic and educational purposes only. It is not intended for clinical use or real-world medical decision-making. The underlying data is synthetic and has been oversampled using SMOTE due to class imbalance. Please do not rely on the predictions for medical or pharmaceutical guidance.