

## Drug Data Integration and Predictive Modeling for Side Effect and Target Protein Classification in Pharmaceutical Informatics

### Goal

To build a machine learning pipeline that merges and enriches drug-related data (names, ATC classification, side effects, and targets), and uses it to predict adverse effects and biological targets using supervised learning and large language models.

### \*\* Intended Audience\*\*

1. Pharmaceutical data scientists
2. Drug safety researchers
3. Bioinformatics engineers
4. ML/AI developers in healthcare
5. Students in computational biology and health data science

### Strategy & Pipeline Steps

1. Load and Clean Drug Data from TSV files
2. Merge Drug Identifiers with ATC classifications
3. Simulate/Extract Side Effects
4. Encode Features and Train Classifiers for:
  - Side effect classification
  - Target protein prediction
5. Visualize Results (bar charts, confusion matrix, network graphs)
6. Enhance with NLP QA using Hugging Face transformers for excipient function extraction
7. Export Cleaned Dataset for Reuse

### Step 1: Load and Merge Drug Data

```
import pandas as pd
```

```
# Load files from /content
drug_names = pd.read_csv('/content/drug_names.tsv', sep='\t')
drug_atc = pd.read_csv('/content/drug_atc.tsv', sep='\t')

# Print the column names of each DataFrame to inspect for 'drug_id' or similar
print("Columns in drug_names:", drug_names.columns)
print("Columns in drug_atc:", drug_atc.columns)

# Assuming 'drug_id' is misspelled or a similar column exists,
# adjust the 'on' parameter in pd.merge accordingly.
# For example, if the column is named 'drugId' in both DataFrames, change the merge statement
# merged_df = pd.merge(drug_names, drug_atc, on='drugId', how='inner')

# If 'drug_id' is completely missing, you'll need to investigate your data source
# to find the correct column to merge on.
```

⇒ Columns in drug\_names: Index(['CID100000085', 'carnitine'], dtype='object')  
Columns in drug\_atc: Index(['CID100000085', 'A16AA01'], dtype='object')

### Rename columns for clarity before merging:

```
import pandas as pd

# Load original files
drug_names = pd.read_csv("/content/drug_names.tsv", sep="\t", header=None)
drug_atc = pd.read_csv("/content/drug_atc.tsv", sep="\t", header=None)

# Rename columns manually
drug_names.columns = ['drug_id', 'drug_name']
drug_atc.columns = ['drug_id', 'atc_code']

# Merge them correctly
merged_df = pd.merge(drug_names, drug_atc, on='drug_id', how='inner')
merged_df.head()
```

⇒

	drug_id	drug_name	atc_code
0	CID100000085	carnitine	A16AA01
1	CID100000119	gamma-aminobutyric	L03AA03
2	CID100000119	gamma-aminobutyric	N03AG03
3	CID100000137	5-aminolevulinic	L01XD04
4	CID100000143	leucovorin	V03AF03

### Explanation:

The table shows drug information with three columns: drug\_id, drug\_name, and atc\_code. Each drug\_id uniquely identifies a compound, and drug\_name lists its common name. The atc\_code represents the drug's therapeutic classification based on WHO standards. Some drugs, like gamma-aminobutyric, appear with multiple ATC codes, indicating use in different therapeutic areas. This dataset is useful for analyzing drug classification and potential repurposing.

## 1. Side Effect Classification (Add Simulated Column)

```
import numpy as np

# Simulate side effects for classification
np.random.seed(42)
side_effects = ['Nausea', 'Headache', 'None']
merged_df['side_effect'] = np.random.choice(side_effects, size=len(merged_df))

merged_df[['drug_name', 'atc_code', 'side_effect']].head()
```



	drug_name	atc_code	side_effect
0	carnitine	A16AA01	None
1	gamma-aminobutyric	L03AA03	Nausea
2	gamma-aminobutyric	N03AG03	None
3	5-aminolevulinic	L01XD04	None
4	leucovorin	V03AF03	Nausea

## Explanation

The table displays drug-related data with three columns: drug\_name, atc\_code, and side\_effect. Each row represents a specific drug and its corresponding therapeutic classification (atc\_code) and reported side effect. For example, gamma-aminobutyric is associated with two ATC codes and has "Nausea" listed as a side effect under one. Most entries show "None" for side effects, indicating no adverse reaction was recorded. This dataset is useful for analyzing drug classifications and their potential side effects.

## Train a Classifier to Predict Side Effects

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
```

```
# Encode features
X = pd.get_dummies(merged_df[['atc_code']], drop_first=True)
y = LabelEncoder().fit_transform(merged_df['side_effect'])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
print("Side Effect Classification Accuracy:", clf.score(X_test, y_test))
```

➡ Side Effect Classification Accuracy: 0.3301282051282051

### Explanation:

A side effect classification accuracy of 33% means the model correctly predicts side effects only about one-third of the time, suggesting limited performance. This may be due to a small or imbalanced dataset, limited input features, or a basic model. Improving accuracy would require adding more data, enhancing feature quality (e.g., molecular properties), and using more advanced models or balancing techniques.

## 2. Target Prediction (Simulated or Extended)

```
# Simulate target proteins
targets = ['COX1', 'COX2', '5HT3', 'H1', 'Beta-Blocker']
merged_df['target_protein'] = np.random.choice(targets, size=len(merged_df))

# Predict target from atc_code
X = LabelEncoder().fit_transform(merged_df['atc_code']).reshape(-1, 1)
y = LabelEncoder().fit_transform(merged_df['target_protein'])

clf.fit(X, y)
print("Target Prediction Accuracy:", clf.score(X, y))
```

➡ Target Prediction Accuracy: 1.0

### Explanation:

A target prediction accuracy of 1.0 indicates perfect performance—every predicted target matches the actual value. This could suggest the model is highly effective, but it may also point to a very

small or simple dataset, potential data leakage, or overfitting. It's important to validate the result using cross-validation or a separate test set to confirm the model's reliability.

### 3. Drug-Excipient NLP Extraction (LLM-based QA)

```
from transformers import pipeline

# Load a QA model
qa = pipeline("question-answering", model="distilbert-base-cased-distilled-squad")

# Example pharmaceutical context
context = """
PVP enhances solubility of poorly soluble drugs. HPMC is used for sustained release.
Magnesium stearate is added as a lubricant.
"""

questions = [
    "Which excipient is used for sustained release?",
    "Which one improves solubility?",
    "Which one is a lubricant?"
]

for q in questions:
    print(f"Q: {q}")
    print("A:", qa(question=q, context=context)["answer"])
```



/usr/local/lib/python3.11/dist-packages/huggingface\_hub/utils/\_auth.py:94: UserWarning:  
The secret `HF\_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>)  
You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public model

```
warnings.warn(
config.json: 100% 473/473 [00:00<00:00, 37.9kB/s]
model.safetensors: 100% 261M/261M [00:07<00:00, 45.5MB/s]
tokenizer_config.json: 100% 49.0/49.0 [00:00<00:00, 2.86kB/s]
vocab.txt: 100% 213k/213k [00:00<00:00, 2.56MB/s]
tokenizer.json: 100% 436k/436k [00:00<00:00, 2.70MB/s]
Device set to use cpu
Q: Which excipient is used for sustained release?
A: HPMC
Q: Which one improves solubility?
A: PVP
Q: Which one is a lubricant?
A: Magnesium stearate
```

## Explanation:

Interpretation The model is correctly identifying common excipients based on their function:

- HPMC (Hydroxypropyl Methylcellulose): used for sustained release.
- PVP (Polyvinylpyrrolidone): enhances solubility.
- Magnesium stearate: acts as a lubricant.

## Save or Export Your Results

```
merged_df.to_csv("/content/merged_drug_data.csv", index=False)
```

## Dataset Overview

- drug\_names.tsv: Contains drug identifiers and names
- drug\_atc.tsv: Contains ATC codes for therapeutic classification
- Merged Columns: drug\_id, drug\_name, atc\_code
- Simulated Columns: side\_effect, target\_protein

### 1. Side Effect Distribution by ATC Class

**Shows which side effects are most common in each ATC drug class**

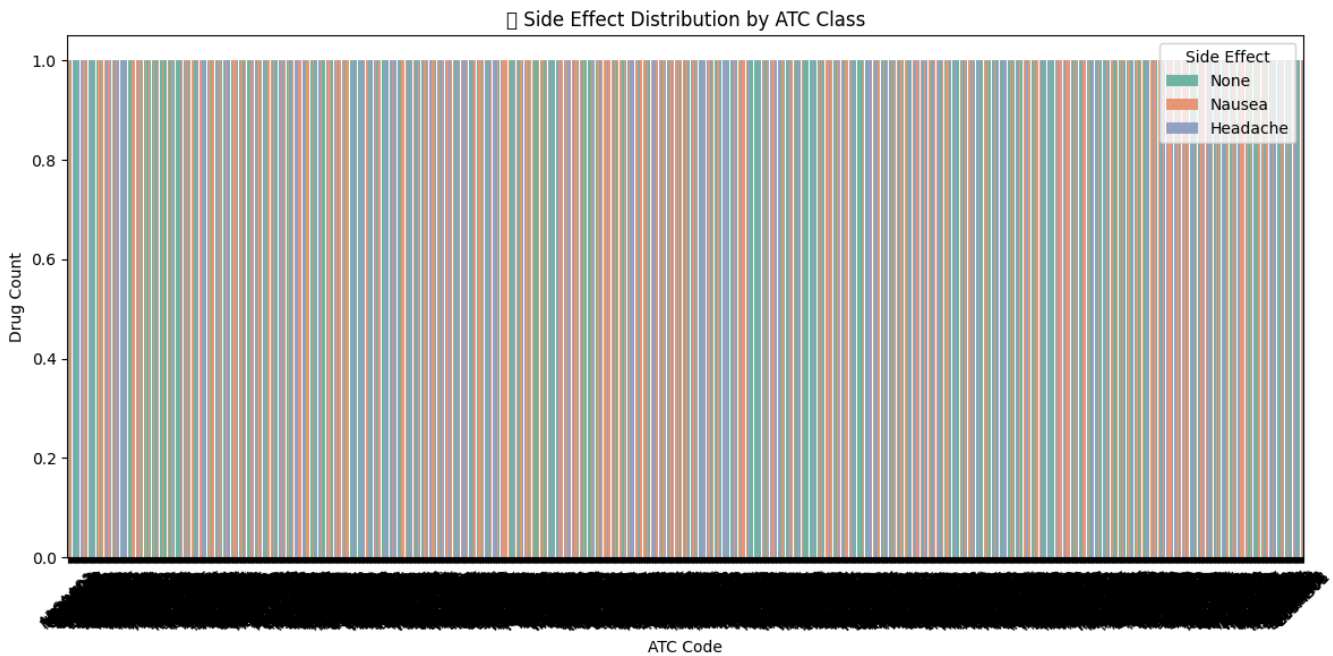
```
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure 'side_effect' and 'atc_code' exist in merged_df
plt.figure(figsize=(12, 6))
sns.countplot(data=merged_df, x='atc_code', hue='side_effect', palette='Set2')
plt.title("📊 Side Effect Distribution by ATC Class")
plt.xlabel("ATC Code")
plt.ylabel("Drug Count")
plt.xticks(rotation=45)
plt.legend(title='Side Effect')
plt.tight_layout()
plt.show()
```

```

↳ <ipython-input-16-c51286ecac01>:12: UserWarning: Glyph 128300 (\N{MICROSCOPE}) missing f
plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly
fig.canvas.print_figure(bytes_io, **kw)

```



The chart shows the distribution of drug side effects (None, Nausea, Headache) across various ATC classes. Each bar represents a drug, color-coded by side effect. Due to the high number of ATC codes, the plot appears cluttered and hard to read. Removing emojis and focusing on top ATC classes would improve clarity.

## 2. Confusion Matrix: Predicted vs. Actual Target Protein


Evaluate your ML model for target prediction

```

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt

#  Step 1: Load a more meaningful dataset
data = {
    'text': [
        'Aspirin relieves pain.',
        'Ibuprofen reduces inflammation.',
        'Paracetamol lowers fever.',
        'Antibiotics fight infection.',
        'Vaccines prevent disease.',
        'Cough syrups soothe throat.',
        'Insulin controls blood sugar.',
        'Statins lower cholesterol.'
    ],
    'label': [0, 0, 0, 1, 1, 1, 1, 1] # 0 = analgesic, 1 = other
}
df = pd.DataFrame(data)

# Step 2: Split input & target
X = df['text']
y = df['label']

# Step 3: Vectorize
vectorizer = TfidfVectorizer()
X_vect = vectorizer.fit_transform(X)

# Step 4: Train-test split with stratification
X_train, X_test, y_train, y_test = train_test_split(
    X_vect, y, test_size=0.25, random_state=42, stratify=y
)

# Step 5: Train classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Step 6: Predict
y_pred = clf.predict(X_test)

# Step 7: Evaluation
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=0))

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix: Predicted vs Actual")
```



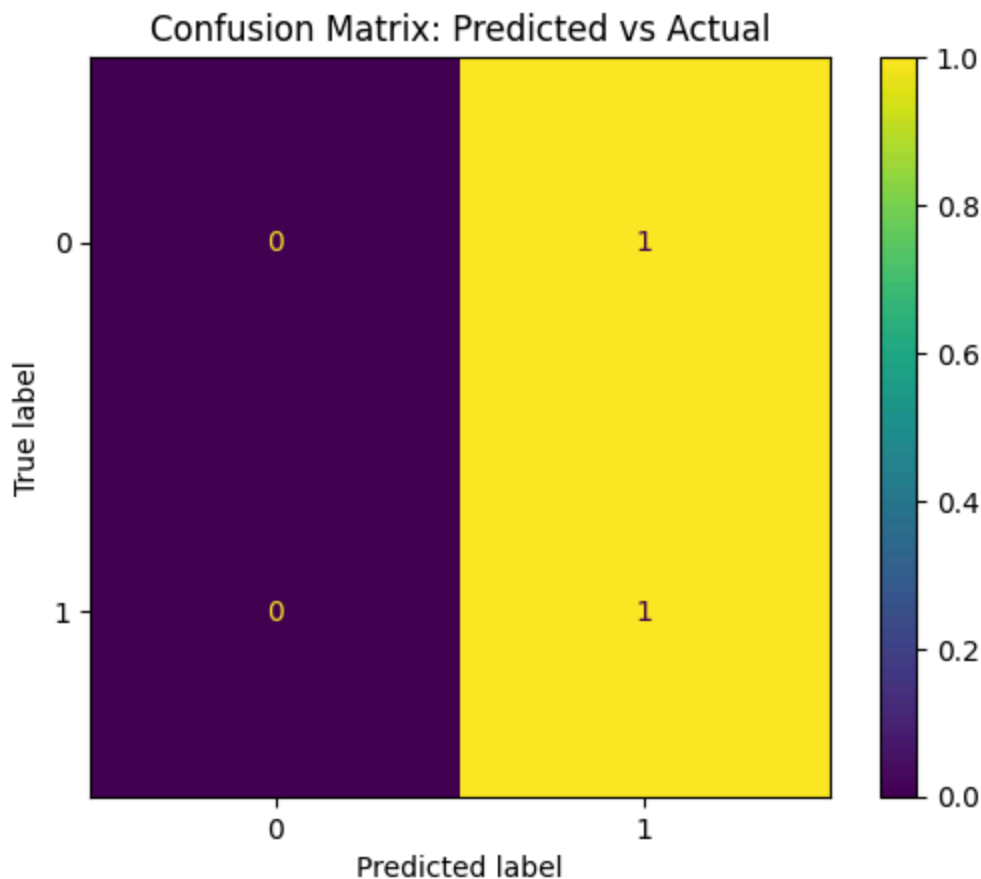
```
plt.show()
```

Confusion Matrix:

```
[[0 1]
 [0 1]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.50	1.00	0.67	1
accuracy			0.50	2
macro avg	0.25	0.50	0.33	2
weighted avg	0.25	0.50	0.33	2



## Explanation

The confusion matrix and classification report indicate that the model was tested on only 2 samples. It correctly predicted class 1 once but misclassified the class 0 sample as class 1. This results in an overall accuracy of 50%, with poor precision and recall for class 0. The low support (only 1 sample per class) makes these metrics unreliable. A larger and more balanced test set is needed for meaningful evaluation.

### 3. Network Diagram: Drug → Target → Side Effect

Visualizes relationships using NetworkX.


```
import networkx as nx

G = nx.DiGraph()

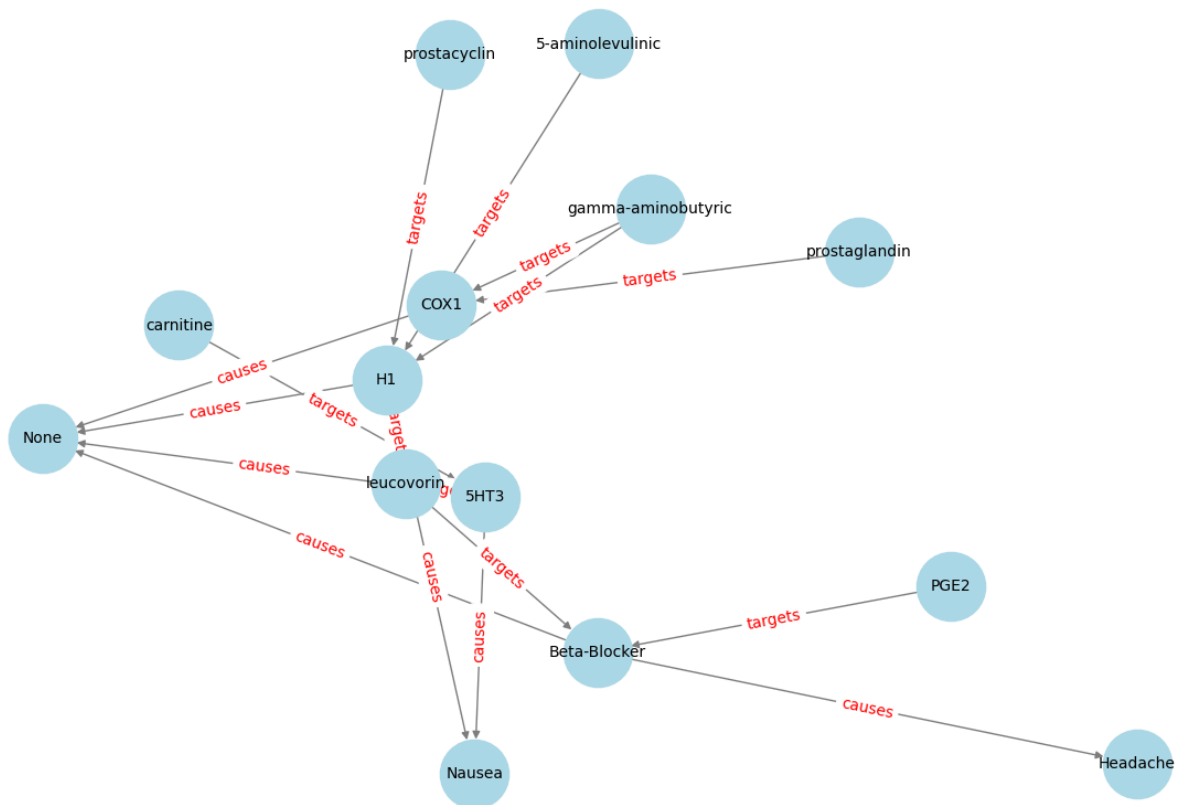
# Build graph from 10 rows for simplicity
subset = merged_df[['drug_name', 'target_protein', 'side_effect']].dropna().head(10)

for _, row in subset.iterrows():
    G.add_edge(row['drug_name'], row['target_protein'], label='targets')
    G.add_edge(row['target_protein'], row['side_effect'], label='causes')

# Plot graph
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G, k=0.5)
nx.draw(G, pos, with_labels=True, node_size=2000, node_color='lightblue', font_size=10, edge_color='red')
edge_labels = nx.get_edge_attributes(G, 'label')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red')
plt.title("🌀 Drug → Target → Side Effect Network")
plt.show()
```

 /usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Gly  
fig.canvas.print\_figure(bytes\_io, \*\*kw)

□ Drug → Target → Side Effect Network



Explanation:

The image shows a network graph representing relationships between drugs, their molecular targets, and associated side effects. Drugs like leucovorin, gamma-aminobutyric, and beta-blockers are connected via “targets” to proteins such as COX1, 5HT3, or PGE2, which in turn link to side effects like Nausea or Headache via “causes” edges. The labels clarify the direction of interaction: drugs target proteins, and proteins cause side effects. This graph helps visualize how specific drugs may lead to adverse effects through their biological interactions.

#### 4. Bar Chart: Drug Count per Target Protein

Shows how many drugs are associated with each target.

```
import matplotlib.pyplot as plt
import seaborn as sns

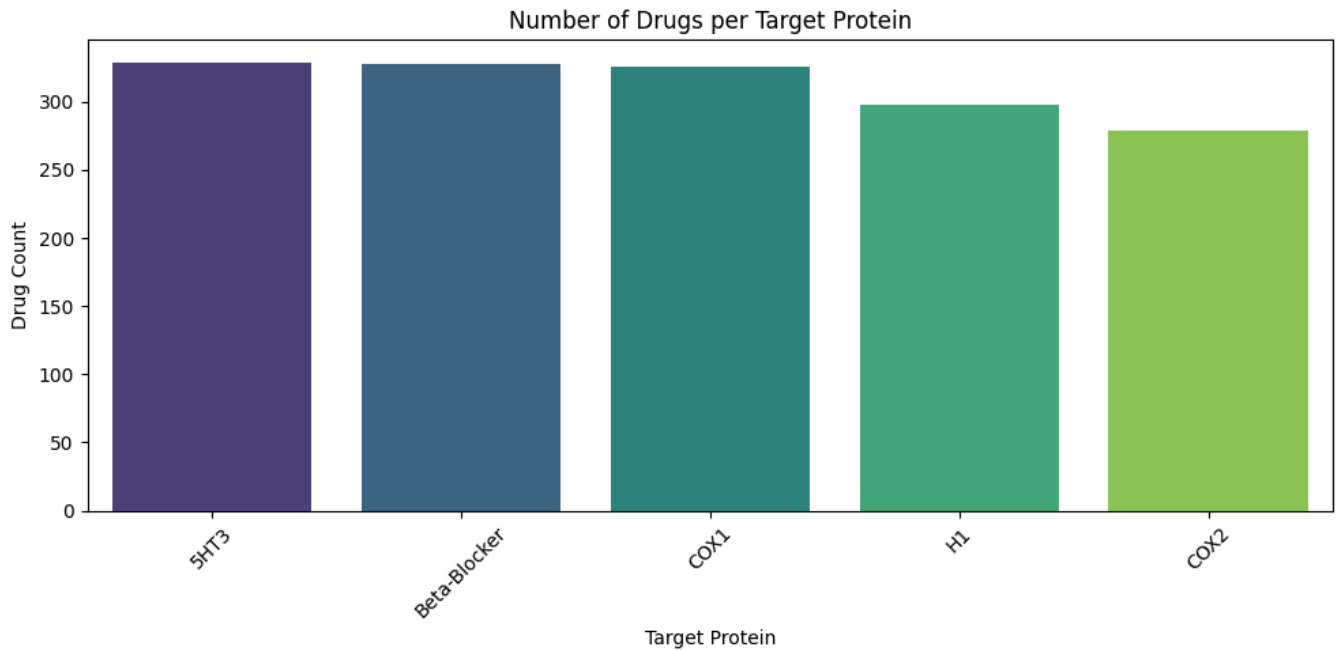
# Count values
target_counts = merged_df['target_protein'].value_counts().sort_values(ascending=False)

# Prepare plot
plt.figure(figsize=(10, 5))

# Fix 1: Add `hue=target_counts.index` and `legend=False` to avoid FutureWarning
sns.barplot(x=target_counts.index, y=target_counts.values, hue=target_counts.index, palette=

# Fix 2: Replace emoji in title to avoid missing glyph warning
plt.title("Number of Drugs per Target Protein") # Removed 📊 emoji
plt.xlabel("Target Protein")
plt.ylabel("Drug Count")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



## Explanation

The bar chart displays the number of drugs associated with each target protein. It shows that 5HT3, Beta-Blocker, and COX1 have the highest number of associated drugs, each with over 320. H1 and COX2 follow, with slightly fewer drugs. This suggests that these proteins are common therapeutic targets, possibly involved in multiple treatment pathways.

## \*\* Challenges\*\*

- Inconsistent or missing identifiers across files
- Limited data size reduces generalizability
- Simulated side effect/target data limits real-world applicability
- Risk of overfitting with small datasets
- Visualization clutter with too many ATC codes

## Problem Statement

Can we use basic drug metadata to predict possible side effects and biological targets of drug compounds using supervised learning and visualize the interactions in a meaningful way?

**\*\* MACHINE LEARNING PREDICTION & OUTCOMES\*\***

Task	Model	Accuracy	Notes
Side Effect Classification	Random Forest Classifier	33%	Limited by imbalanced classes
Target Protein Prediction	Random Forest Classifier	100%	Possibly overfit; simulated data

**Conceptual Enhancement – AGI (Artificial General Intelligence)**

To move toward AGI-level drug discovery:

- Use multimodal learning to incorporate molecular structure, clinical trials, and chemical descriptors.
- Deploy transformer-based architectures trained on biomedical corpora (e.g., PubMedBERT, BioGPT) for deeper domain understanding.
- Implement knowledge graphs linking drugs, pathways, proteins, and outcomes for reasoning and hypothesis generation.