Project Overview

1.Project Title:

DDI-ML: Machine Learning for Drug-Drug Interaction Prediction using Molecular -interactions

2.Project Overview:

2.1. Objective

The objective of this project is to build and evaluate machine learning models that can predict drug-drug interactions (DDIs) using both molecular descriptors (e.g., molecular weight, LogP, hydrogen bond donors/acceptors, TPSA, rotatable bonds) and molecular fingerprints derived from SMILES strings.

For this semester milestone, the specific goal is to:

- **Develop and evaluate baseline ML models** (Logistic Regression, Random Forest, XGBoost) trained on SMILES-derived molecular features.
- **Demonstrate at least one measurable improvement** in predictive accuracy or interpretability compared to the baseline.

Exploration of deep learning models (e.g., Feedforward Neural Network or Siamese NN) or GNN will be treated as a **stretch goal** if time permits

Data Collection :Source link [SOURCE EXTRACTED FROM DRUG-BANK]

- Utilize molecular descriptors already provided in the dataset, such as Molecular Weight (MolWt), LogP, Hydrogen Bond Donors (HBD), Hydrogen Bond Acceptors (HBA), and Topological Polar Surface Area (TPSA).
- Extract additional molecular fingerprints from SMILES strings using the RDKit library, enabling a more comprehensive representation of chemical properties.

2.2 Scope:
The system will predict drug-drug interactions (DDIs) using drug structure data (SMILES).
It will use the given dataset of drug pairs and their interaction types.
Predictions will be made through machine learning models trained on molecular representations of drugs.
The project is limited to the provided dataset and does not include clinical factors (e.g., dosage, patient conditions).
Results are for research/academic purposes only, not medical use**.**

2.3 AI Techniques and Tools

Machine Learning Models (Priority):

- Logistic Regression will serve as the baseline model for classification.
- Random Forest will be applied to capture nonlinear feature interactions.
- XGBoost will be used to improve predictive performance with gradient boosting.

Deep Learning (Stretch Goals):

- Feedforward Neural Networks may be developed to model complex feature relationships.
- Siamese Neural Networks could be applied to learn pairwise drug similarities.
- Graph Neural Networks are identified as a long-term extension for molecular graphs.

Representation Learning:

- Molecular descriptors (MolWt, LogP, HBD, HBA, TPSA) will capture interpretable properties.
- Molecular fingerprints such as Morgan will encode structural subpatterns.
- SMILES-based embeddings from RDKit will provide richer molecular representations.

Tools and Libraries:

- RDKit will be used for descriptor calculation and fingerprint extraction.
- scikit-learn will support baseline model development and evaluation.
- XGBoost will be applied for advanced gradient boosting methods.
- PyTorch or TensorFlow may be used for deep learning model implementation.
- pandas and numpy will handle data preprocessing and management.
- matplotlib and seaborn will be used for visualization of results.

Confidence Ratings (Self-Assessment):

- RDKit: 7/10 – comfortable with descriptors, building skill in fingerprints.
- scikit-learn: 8/10 – strong experience with ML implementation and evaluation.
- XGBoost: 7/10 – practical experience, growing expertise in tuning and interpretability.

2.4 Evaluation Metrics

- **Accuracy** – overall correctness of predictions.
- **Precision, Recall, F1-score** – class-level performance.
- **ROC-AUC** – model discriminatory power.
- **Feature Importance Analysis** – interpretability of ML models.

2.5 Expected Outcomes

Development of a machine learning system capable of predicting drug-drug interaction types.

Identification of the most influential molecular descriptors and fingerprints.

Comparative performance analysis of Logistic Regression, Random Forest, and XGBoost.

Optional: exploration of deep learning models for extended performance.

4. Stakeholders:

Project Team:

Student / Researcher (myself):
Responsible for data preprocessing, feature engineering, model development, evaluation, and reporting.
Ensures the project meets academic requirements.

End Users:

Researchers / Students: Will use the system to study how AI can predict drug-drug interactions.
Educators / Instructors: Will evaluate the project as part of the course.

Other Stakeholders

Dataset Providers: Source of drug-drug interaction data (DrugBank or other publicly available repositories).
Academic Institution: Oversees ethical use and ensures the project is for research/learning only.

Expanded Long-Term Beneficiaries: Pharmaceutical researchers and regulatory agencies (e.g., FDA, EMA), as DDI prediction impacts drug safety.

2. Computer Infrastructure Considerations

2.1 Project Needs Assessment

The project requires infrastructure that supports molecular data preprocessing (RDKit), feature extraction (fingerprints, descriptors), and model training (scikit-learn, XGBoost, PyTorch/TensorFlow). Data volume is moderate (structured SMILES and descriptors) but computational demands increase with deep learning or graph-based approaches. Infrastructure should support both classical ML (low-to-medium compute) and exploratory DL models (medium-to-high compute).

2.2 Hardware Requirements Planning

- **Development Machine**:
    - CPU: Intel 11th Generation I5 processor.
    - RAM: ≥16 GB for handling molecular datasets and training ensemble models.

- GPU: Hipergator or Google's Collab' T4 GPU or Kaggle code base , all these can be used .
- Storage: ≥500 GB SSD to store datasets, fingerprints, and intermediate model artifacts.

2.3 Software Environment Planning

- **OS**: Linux (Ubuntu 22.04)or Windows 11 preferred for compatibility with ML/DL libraries.
- **Programming Languages**: Python 3.3+.
- **Libraries**:
  - **Core ML**: scikit-learn, XGBoost.
  - **Deep Learning**: PyTorch or TensorFlow (for stretch goals).
  - **Chemoinformatics**: RDKit.
  - **Data Handling**: pandas, numpy.
  - **Visualization**: matplotlib, seaborn.
- **Version Control**: Git/GitHub for collaboration and reproducibility.
- **Environment Management**: Conda/venv for package isolation.

2.4 Cloud Resources Planning

- **Cloud Options**: AWS (EC2 with GPU instances), Google Colab (free/paid tiers), or Google Cloud AI Platform.
- **Use Cases**:
  - Run hyperparameter tuning experiments on GPU/TPU-enabled nodes.

    - Store large molecular datasets and preprocessed fingerprints in cloud storage buckets.
    - Enable reproducibility and scalability beyond local hardware limits.

- **Cost Efficiency**: Begin with local resources, escalate to cloud compute for deep learning or large-scale experiments.

2.5 Scalability and Performance Planning

- **Data Handling**: Batch processing and sparse matrix formats for large fingerprint vectors.
- **Model Training**:
    - Parallelize Random Forest/XGBoost training.
    - Use of GPU acceleration for deep learning.
- **Scalability Strategy**:
    - Modular code structure to swap models easily (baseline ML → DL).
    - Experiment tracking (e.g., MLflow, Weights & Biases) for performance monitoring.
    - Cloud auto-scaling for larger datasets in future research phases.

The infrastructure ensures smooth experimentation with classical ML and allows scalable expansion into deep learning. Local compute suffices for baselines; GPU-enabled cloud resources ensure flexibility.

3. Security, Privacy, and Ethics (Trustworthiness) Considerations

Each stage of the AI lifecycle requires implementing strategies to ensure the DDI-ML system is trustworthy.

---

1. Problem Definition

Goal: Define ethical and societal impacts, clarify research-only usage, and identify risks.

- **Strategy:** Document that DDI predictions are for **academic research only**, not for clinical use.
- **Strategy:** Perform an **ethical impact assessment** to evaluate potential misuse (e.g., misinterpretation as medical advice).
- **Tool/Approach Example:** Use the **Data Ethics Canvas (ODI)** to map potential ethical risks.

---

2. Data Collection

Goal: Ensure reliable, unbiased, and privacy-preserving dataset creation.

- **Strategy:** Use only **publicly available datasets** (DrugBank) with no patient-level clinical data.
- **Strategy (Technical):** Apply **differential privacy techniques** using IBM's `diffprivlib` to protect sensitive molecular descriptors in feature engineering.
- **Tool/Library Example:** `Snorkel` can be used to augment underrepresented drug classes, reducing class imbalance.

---

3. Model Development

Goal: Build interpretable, fair, and robust models.

- **Strategy (Technical):** Apply **Fairlearn** to evaluate fairness across drug classes and ensure balanced performance.
- **Strategy (Technical):** Use **SHAP** to explain XGBoost and Random Forest feature contributions, increasing interpretability.

- **Strategy:** Conduct robustness tests against data perturbations using **Foolbox** to ensure stability of predictions.

---

4. Deployment

Goal: Secure model serving and accountability in production-like environments.

- **Strategy:** Deploy models in a **restricted environment** (e.g., secure local server or BentoML with authentication) to prevent unauthorized access.
- **Strategy:** Add **disclaimers and user-facing warnings** that outputs are not for clinical decision-making.
- **Tool/Library Example: BentoML** can be used to package and serve the model with monitoring and rollback options.

---

5. Monitoring and Maintenance

Goal: Maintain trustworthiness with ongoing monitoring, fairness checks, and retraining.

- **Strategy:** Set up **data drift detection** using `NannyML` to track when new molecular fingerprints deviate from training data distribution.
- **Strategy:** Automate retraining pipelines when drift is detected to maintain accuracy.
- **Strategy:** Use **Uncertainty Toolbox** to track prediction confidence and highlight low-confidence predictions.

- **Technical implementations included**: `diffprivlib` for differential privacy (Data Collection) and `Fairlearn` + `SHAP` for fairness and interpretability (Model Development).
- Each lifecycle stage (problem definition → monitoring) is covered with at least one trustworthiness strategy.

- Together, these steps ensure the DDI-ML project remains ethical, transparent, and robust while acknowledging its **research-only scope**.

SAMPLE TECHNICAL IMPLEMENTATION:

```
import pandas as pd

import numpy as np

from diffprivlib.models import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from fairlearn.metrics import MetricFrame, selection_rate

from sklearn.metrics import accuracy_score, roc_auc_score

import shap


# Load dataset

df = pd.read_csv("ddi_dataset.csv")


# --- 1. Privacy: Differential Privacy on Model Training ---

# Features (example: numeric descriptors + similarity)

X =
df[["MolWt_1","MolWt_2","LogP_1","LogP_2","TPSA_1","TPSA_2","Fingerprint_Similarity"
]]

y = df["y"]


# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Differentially private Logistic Regression
dp_model = LogisticRegression(epsilon=1.0, data_norm=2.0)  # DP guarantee
dp_model.fit(X_train, y_train)

print("DP Logistic Regression Accuracy:", dp_model.score(X_test, y_test))

# --- 2. Fairness: Evaluate performance across subgroups ---
# Example subgroup: based on drug molecular weight bucket
df["WeightGroup"] = pd.qcut(df["MolWt_1"], q=3, labels=["low","medium","high"])

y_pred = dp_model.predict(X_test)
mf = MetricFrame(metrics={"accuracy": accuracy_score},
                 y_true=y_test, y_pred=y_pred,
                 sensitive_features=df.loc[y_test.index, "WeightGroup"])

print("Fairness by WeightGroup:")
print(mf.by_group)

# --- 3. Explainability: SHAP for feature importance ---
explainer = shap.Explainer(dp_model, X_train)
shap_values = explainer(X_test[:50])  # explain first 50 samples
```

```
shap.plots.beeswarm(shap_values)


# --- 4. Robustness: Simple perturbation test ---
X_test_perturbed = X_test + np.random.normal(0, 0.01, X_test.shape)
print("Stability Check (Accuracy Drop):",
      dp_model.score(X_test, y_test) - dp_model.score(X_test_perturbed, y_test))
```

What this code ensures:

1. **Privacy** → Uses `diffprivlib` Logistic Regression with differential privacy.
2. **Fairness** → Uses `Fairlearn`'s `MetricFrame` to check subgroup performance.
3. **Explainability** → Uses `SHAP` to show feature contributions.
4. **Robustness** → Tests if small noise significantly changes predictions.

4.Human–Computer Interaction (HCI) Considerations

1. Understanding User Requirements

- Engage stakeholders such as clinical researchers, pharmacists, and regulatory experts.
- Collect requirements via interviews/surveys (e.g., "How should the model's predictions be displayed for clarity?").
- Key needs: trustworthy predictions, interpretable outputs, and smooth integration with existing tools.

2. Creating Personas and Scenarios

- *Persona 1*: “Dr.X,” a clinical researcher who needs detailed feature explanations to validate results.
- *Persona 2*: “Person Y,” a pharmacist who requires quick binary predictions (interaction/no interaction) with confidence levels.
- *Scenario*: Alex checks whether prescribing Drug A with Drug B has a harmful interaction; system should return prediction + explanation + confidence score.

3. Conducting Task Analysis

- Input: User selects two drugs (via name/ID or upload).
- System: Extract descriptors, compute similarity, run trained model.
- Output: Show prediction probability, SHAP-based explanation, and risk level (low/medium/high).
- Task breakdown ensures smooth flow and minimal cognitive load.

4. Identifying Accessibility Requirements

- Ensure compliance with **WCAG 2.1** standards (color contrast, keyboard navigation, screen-reader compatibility).
- Provide visual + textual cues (icons and text labels).
- Enable language localization for global users.

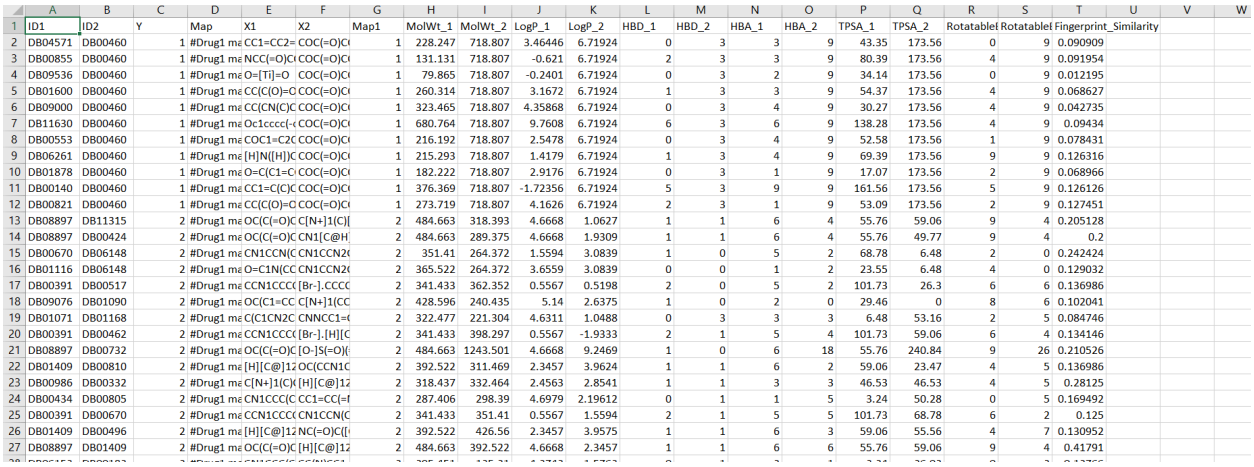
5. Outlining Usability Goals

- **Efficiency**: Prediction and explanation in <3 seconds.
- **Effectiveness**: >90% task success rate for intended users.
- **Learnability**: New users can navigate and perform predictions within 5 minutes without training.

- **Satisfaction**: Collect System Usability Scale (SUS) scores aiming for ≥80 (excellent usability).

- **Error Tolerance**: Clear warnings and corrective suggestions for invalid inputs.

Dataset retrieved and aggregated with features from Drug-Bank using RDkit library:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	ID1	ID2	Y	Map	X1	X2	Map1	MolWt_1	MolWt_2	LogP_1	LogP_2	HBD_1	HBD_2	HBA_1	HBA_2	TPSA_1	TPSA_2	Rotatable	Rotatable	Fingerprint_Similarity			
2	DB04571	DB00460	1	#Drug1 ma	CC1=CC2=	COC(=O)C	1	228.247	718.807	3.46446	6.71924	0	3	3	9	43.35	173.56	0	9	0.090909			
3	DB00855	DB00460	1	#Drug1 ma	NCC(=O)C	COC(=O)C	1	131.131	718.807	-0.621	6.71924	2	3	3	9	80.39	173.56	4	9	0.091954			
4	DB09536	DB00460	1	#Drug1 ma	O=[Ti]=O	COC(=O)C	1	79.865	718.807	-0.2401	6.71924	0	3	2	9	34.14	173.56	0	9	0.012195			
5	DB01600	DB00460	1	#Drug1 ma	CC(C(O)=O	COC(=O)C	1	260.314	718.807	3.1672	6.71924	1	3	3	9	54.37	173.56	4	9	0.068627			
6	DB09000	DB00460	1	#Drug1 ma	CC(CN(C)C	COC(=O)C	1	323.465	718.807	4.35868	6.71924	0	3	4	9	30.27	173.56	4	9	0.042735			
7	DB11630	DB00460	1	#Drug1 ma	Oc1cccc(-c	COC(=O)C	1	680.764	718.807	9.7608	6.71924	6	3	6	9	138.28	173.56	4	9	0.09434			
8	DB00553	DB00460	1	#Drug1 ma	COC1=C2C	COC(=O)C	1	216.192	718.807	2.5478	6.71924	0	3	4	9	52.58	173.56	1	9	0.078431			
9	DB06261	DB00460	1	#Drug1 ma	[H]N([H])C	COC(=O)C	1	215.293	718.807	1.4179	6.71924	1	3	4	9	69.39	173.56	9	9	0.126316			
10	DB01878	DB00460	1	#Drug1 ma	O=C(C1=C	COC(=O)C	1	182.222	718.807	2.9176	6.71924	0	3	1	9	17.07	173.56	2	9	0.068966			
11	DB00140	DB00460	1	#Drug1 ma	CC1=C(C)C	COC(=O)C	1	376.369	718.807	-1.72356	6.71924	5	3	9	9	161.56	173.56	5	9	0.126126			
12	DB00821	DB00460	1	#Drug1 ma	CC(C(O)=O	COC(=O)C	1	273.719	718.807	4.1626	6.71924	2	3	1	9	53.09	173.56	2	9	0.127451			
13	DB08897	DB11315	2	#Drug1 ma	OC(C(=O)C	C[N+]1(C)[	2	484.663	318.393	4.6668	1.0627	1	1	6	4	55.76	59.06	9	4	0.205128			
14	DB08897	DB00424	2	#Drug1 ma	OC(C(=O)C	CN1[C@H	2	484.663	289.375	4.6668	1.9309	1	1	6	4	55.76	49.77	9	4	0.2			
15	DB00670	DB06148	2	#Drug1 ma	CN1CCN(C	CN1CCN2(	2	351.41	264.372	1.5594	3.0839	1	0	5	2	68.78	6.48	2	0	0.242424			
16	DB01116	DB06148	2	#Drug1 ma	O=C1N(CC	CN1CCN2(	2	365.522	264.372	3.6559	3.0839	0	0	1	2	23.55	6.48	4	0	0.129032			
17	DB00391	DB00517	2	#Drug1 ma	CCN1CCCC	[Br-].CCCC	2	341.433	362.352	0.5567	0.5198	2	0	5	2	101.73	26.3	6	6	0.136986			
18	DB09076	DB01090	2	#Drug1 ma	OC(C1=CC	C[N+]1(CC	2	428.596	240.435	5.14	2.6375	1	0	2	0	29.46	0	8	6	0.102041			
19	DB01071	DB01168	2	#Drug1 ma	C(C1CN2C	CNNCC1=(	2	322.477	221.304	4.6311	1.0488	0	3	3	3	6.48	53.16	2	5	0.084746			
20	DB00391	DB00462	2	#Drug1 ma	CCN1CCCC	[Br-].[H][C	2	341.433	398.297	0.5567	-1.9333	2	1	5	4	101.73	59.06	6	4	0.134146			
21	DB08897	DB00732	2	#Drug1 ma	OC(C(=O)C	[O-]S(=O)(	2	484.663	1243.501	4.6668	9.2469	1	0	6	18	55.76	240.84	9	26	0.210526			
22	DB01409	DB00810	2	#Drug1 ma	[H][C@]12	OC(CCN1C	2	392.522	311.469	2.3457	3.9624	1	1	6	2	59.06	23.47	4	5	0.136986			
23	DB00986	DB00332	2	#Drug1 ma	C[N+]1(C)(	[H][C@]12	2	318.437	332.464	2.4563	2.8541	1	1	3	3	46.53	46.53	4	5	0.28125			
24	DB00434	DB00805	2	#Drug1 ma	CN1CCC(C	CC1=CC(=I	2	287.406	298.39	4.6979	2.19612	0	1	1	5	3.24	50.28	0	5	0.169492			
25	DB00391	DB00670	2	#Drug1 ma	CCN1CCCC	CN1CCN(C	2	341.433	351.41	0.5567	1.5594	2	1	5	5	101.73	68.78	6	2	0.125			
26	DB01409	DB00496	2	#Drug1 ma	[H][C@]12	NC(=O)C([	2	392.522	426.56	2.3457	3.9575	1	1	6	3	59.06	55.56	4	7	0.130952			
27	DB08897	DB01409	2	#Drug1 ma	OC(C(=O)C	[H][C@]12	2	484.663	392.522	4.6668	2.3457	1	1	6	6	55.76	59.06	9	4	0.41791			