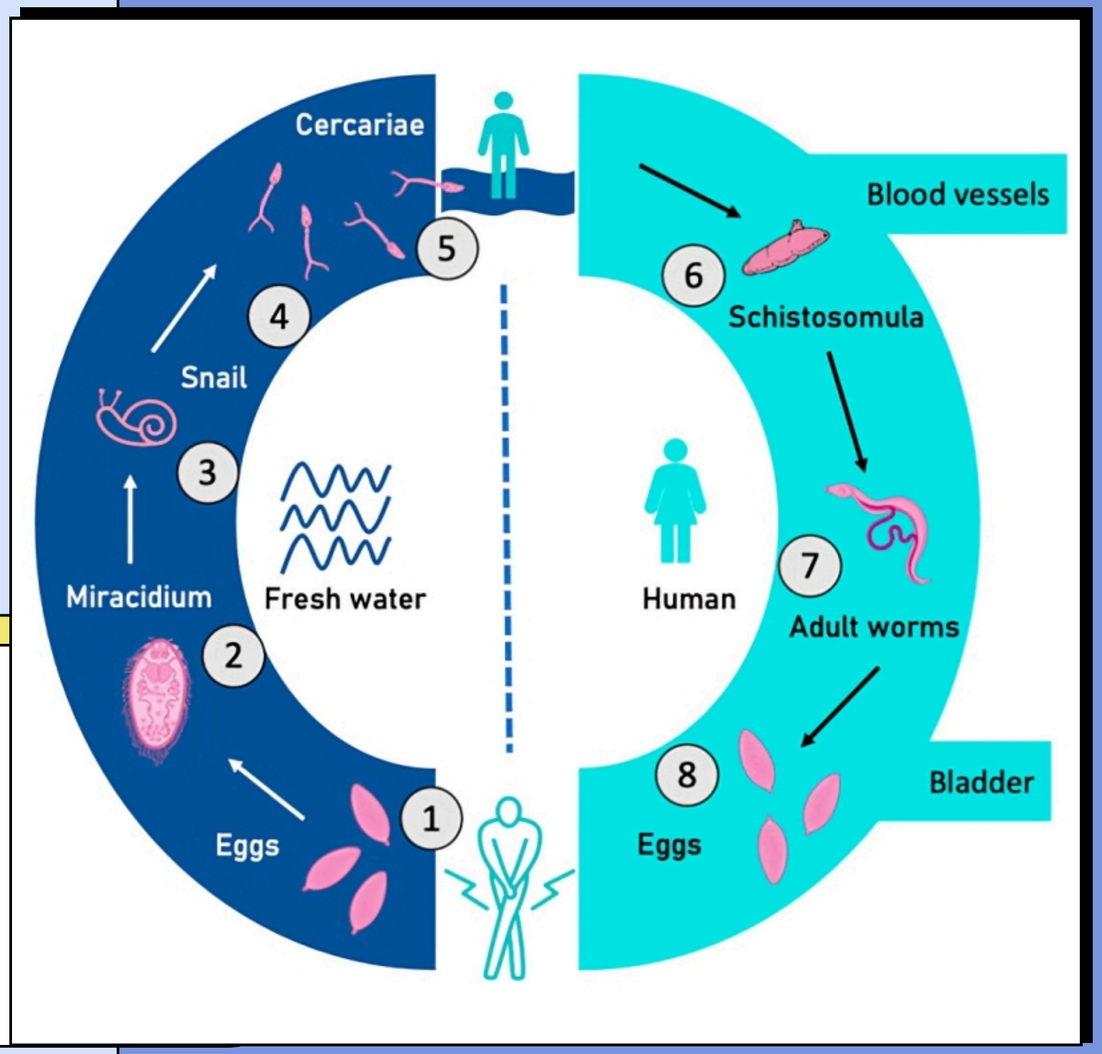
AI-Based Drug Activity Prediction for Schistosomiasis

By

TEAM

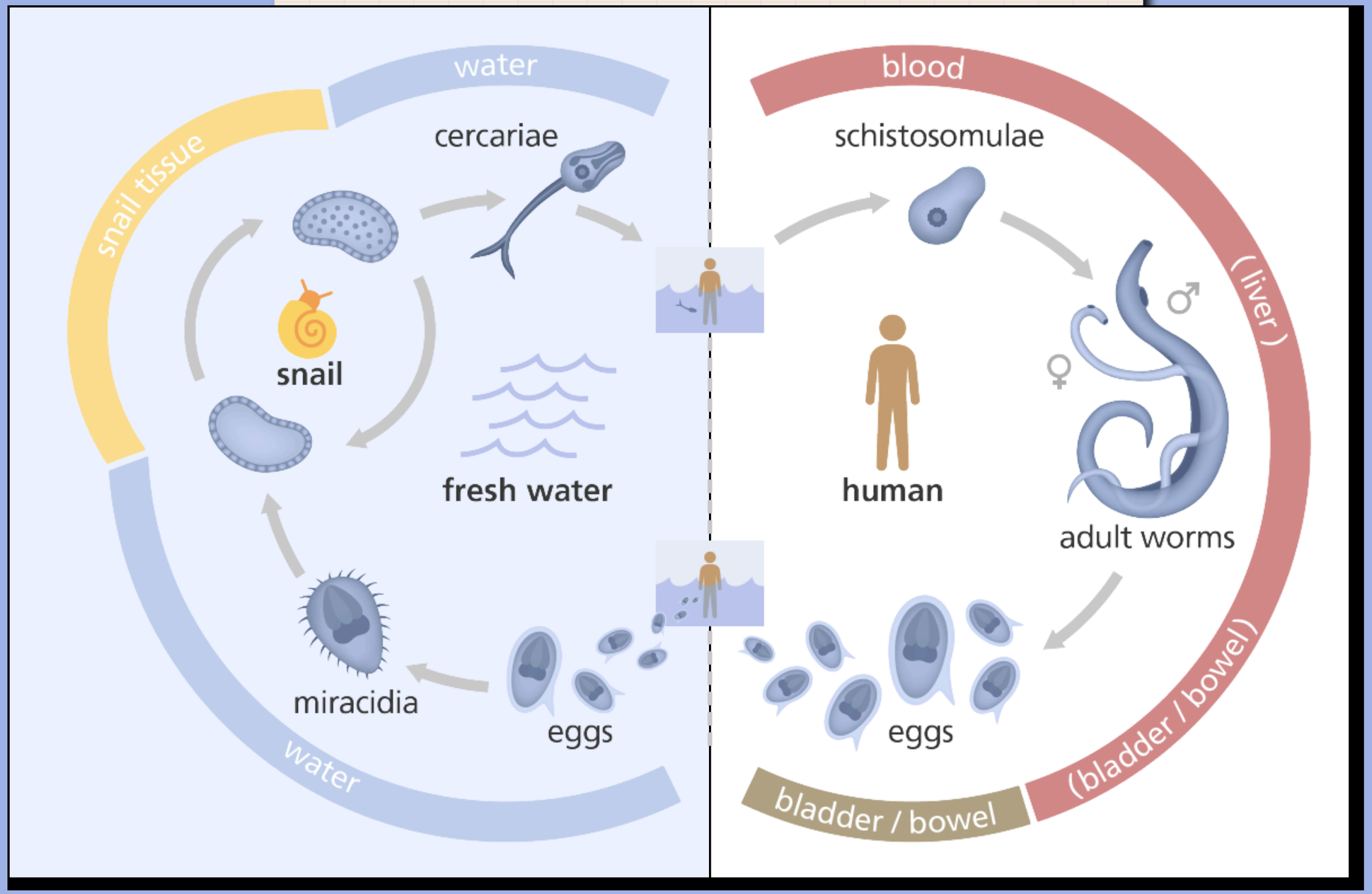
TRIOSPEX



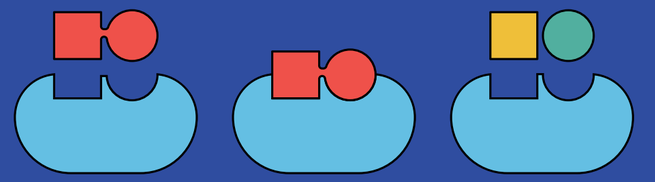
Schistosomiasis (bilharzia)

- According to **WHO**, it is a parasitic acute and chronic neglected tropical disease caused by blood flukes (trematode worms) of the genus *Schistosoma* which currently affects approximately **251 million** people in **55+** endemic countries.
- Over **700–750 million** individuals living in endemic areas are at risk.
- **Female Genital Schistosomiasis (FGS)** affects an estimated **56 million women and girls**, primarily in **sub-Saharan Africa**—and greatly increases **HIV** risk.
- Drugs Used for Schistosomiasis Treatment :
  - **Praziquantel** – First-line drug, kills all major *Schistosoma* species by paralyzing worms.
  - **Oxamniquine** – Alternative for S. *mansoni*, disrupts parasite DNA.
  - **Metrifonate** – Used for S. *haematobium*, paralyzes worms (less common now).

Life Cycle of *Schistosoma*



Predicting Compounds Active Against Schistosoma Parasites



What is TGR ?

- **Thioredoxin Glutathione Reductase**, a vital **antioxidant enzyme** found in ***Schistosoma*** species – (S. *mansoni*, S. *haematobium*, S. *japonicum*).
- Maintains the parasite's **redox balance** and protects it from **oxidative stress** caused by the host immune system.

Why Target TGR ?

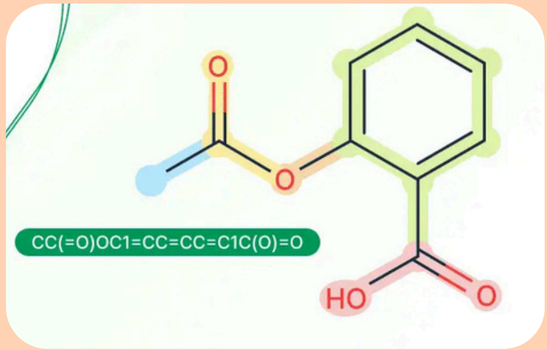
- TGR is essential for **parasite survival** and **function**.
- Structurally distinct from human TGR, allowing **selective targeting**.
- Conserved across species, making it a **broad-spectrum drug target**.

How We'll Predict Active Compounds ?

DATA

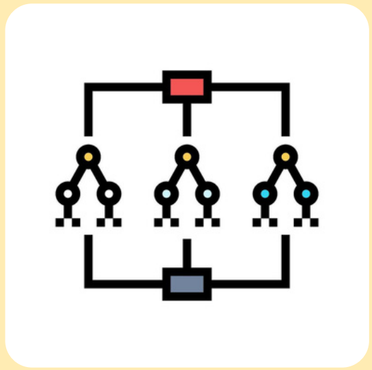
Subset of 21,000+ molecules from **PubChem** (AID 485364) tested for Schistosoma.

FEATURES



Generate **Morgan fingerprints** from compound **SMILES** (text-based molecular formats).

MODEL



Apply **Random Forest** — a supervised ensemble ML algorithm.

Imports for Molecular Fingerprinting and Classification

- **RDKit** converts SMILES to molecular fingerprints
- **Random Forest** For predicting compound activity
- **Scikit-learn** handles training and evaluation
- **Pandas** & **NumPy** manage data processing and transformation.
- **Seaborn** & **Matplotlib** visualize prediction results and t-SNE plots.
- Performance metrics include **accuracy**, **precision**, **recall**, **F1**, and **ROC-AUC**.

```
!pip install rdkit-pypi  # Install RDKit for chemical informatics tasks

import warnings
warnings.filterwarnings('ignore')  # Suppress warnings for cleaner output

from sklearn.ensemble import RandomForestClassifier  # ML model
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score  # Evaluation metrics
import pandas as pd  # Data handling
from rdkit import Chem  # Chemical parsing
from rdkit.Chem import AllChem  # Fingerprint generation
from sklearn.model_selection import train_test_split  # Train-test splitting
from sklearn.metrics import classification_report,
confusion_matrix  # Evaluation reporting
import seaborn as sns  # Visualization
import matplotlib.pyplot as plt  # Plotting
import numpy as np  # Numerical computing
from sklearn.manifold import TSNE  # Dimensionality reduction for visualization
import joblib  # Save/load models

pd.set_option("display.max_columns",None)  # Show all DataFrame columns
```

Loading and Preparing Dataset

- Load dataset from Google Drive.
- Keep only **SMILES** and **activity labels**.
- Split into **Active** and **Inactive** classes.
- **Balance** data by sampling Inactives.
- Combine into one **balanced dataset**.

```python
# Load dataset
df = pd.read_csv("https://drive.google.com/uc?export=download&id=170QCKOiuQV2-cuoV7Knn75-bFekRpyWg")

#Drop unrelated columns
columns_to_drop = [col for col in df.columns if col not in
['PUBCHEM_EXT_DATASOURCE_SMILES', 'PUBCHEM_ACTIVITY_OUTCOME']]
df = df.drop(columns=columns_to_drop) # Keep only necessary columns

#Separate and balance classes
active_df = df[df['PUBCHEM_ACTIVITY_OUTCOME'] == 'Active']
inactive_df = df[df['PUBCHEM_ACTIVITY_OUTCOME'] == 'Inactive']
inactive_sample = inactive_df.sample(n=10784, random_state=42)
balanced_df = pd.concat([active_df, inactive_sample])

#View balanced data
display(balanced_df)
```

TABLE : *balanced_df*

	PUBCHEM_EXT_DATASOURCE_SMILES	PUBCHEM_ACTIVITY_OUTCOME
11	C1CCN(CC1)C2(CCN(CC2)C3CC(=O)N(C3=O)CC4=CC=C(C...	Active
55	COC1=C(C=C(C=C1)CCNC(=O)/C=C/C2=CC=CO2)OC	Active
60	COC1=CC=C(C=C1)N2C(=O)CC(C2=O)N3CCN(CC3)C4=NC5...	Active
112	CC1CN(CCN1C2=CC=CC(=C2)C)C(=O)C3=NOC4=C3COC5=C...	Active
114	CCC(=O)NC1=NN2C(C=C(NC2=N1)C3=CC=CC=C3)C4=CC=C...	Active
...	...	...
223157	CC1=CC(=CC(=C1)NC(=S)/N=C(\N)/NC2=NC3=CC=CC=C3...	Inactive
75573	COC1=C(C=C(C=C1)Cl)NC(=O)N(CC=C)CC=C	Inactive
11265	COC1=CC=C(C=C1)C(=O)N2CCN(CC2)CC(=O)N3CCOCC3	Inactive
19285	CCOC(=O)C1=C(N(N=C1C)C2=CC=C(C=C2)NC(=O)C3=CC=...	Inactive
6362	CCOC(=O)C1=CN=C2C(=C(C=CC2=C1NCCN3CCOCC3)C)C	Inactive

21568 rows × 2 columns

Fingerprint Generation & Label Encoding

- Convert **SMILES** strings to molecular fingerprints (2048-bit).
- Apply **fingerprinting** to all compounds in the balanced dataset

```python
# Convert SMILES to fingerprints
def smiles_to_fp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return list(AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=2048))
    else:
        return [0]*2048

#Apply fingerprinting
balanced_df['fingerprint'] = balanced_df['PUBCHEM_EXT_DATASOURCE_SMILES'].apply(smiles_to_fp)
X = np.array(balanced_df['fingerprint'].tolist())

#Encode activity
y = balanced_df['PUBCHEM_ACTIVITY_OUTCOME'].apply(lambda x: 1 if x == 'Active' else 0)
```

- Store fingerprints as **input features** for model training
- Encode compound activity as binary labels : **Active = 1**, **Inactive = 0**

Model Training & Evaluation

```python
#Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42, stratify=y)

#Train random forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1] if hasattr(model,
 "predict_proba") else [0]*len(y_test)

#Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=0)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

#Display results
print(f"Classification Report:\n")
print(classification_report(y_test, y_pred))
results = {"Accuracy": accuracy, "Precision": precision, "Recall": recall,
    "F1 Score": f1, "ROC-AUC Score": roc_auc}
results_df = pd.DataFrame([results])
display(results_df)
```

- Split data into training and test sets **(80/20 split).**
- Train **Random Forest** on fingerprint features.
- **Predict** activity on test data
- Compute **evaluation metrics** : Accuracy, Precision, Recall, F1, ROC-AUC
- Display **classification report** and **performance summary.**

OUTPUT : *results_df*

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.86	0.84	2157
1	0.86	0.81	0.83	2157
accuracy			0.84	4314
macro avg	0.84	0.84	0.84	4314
weighted avg	0.84	0.84	0.84	4314

	Accuracy	Precision	Recall	F1 Score	ROC-AUC Score
0	0.839128	0.857352	0.81363	0.834919	0.907226

Retraining & Full Prediction

- Retrain **Random Forest** on the entire balanced dataset.
- **Predict** activity for all compounds in the dataset
- Store **predictions** in a new column
- Filter and display compounds predicted as **Active.**

```
#Reinitialize model and retrain on full data
randomforest = RandomForestClassifier(random_state=42)
randomforest.fit(X, y)

#Predict full dataset
all_predictions = randomforest.predict(X)
balanced_df['predicted_activity'] = all_predictions
predicted_actives = balanced_df[balanced_df['predicted_activity'] == 1]
print("Compounds predicted to be active:")
display(predicted_actives)
```

TABLE : *predicted_actives*

Compounds predicted to be active:

	PUBCHEM_EXT_DATASOURCE_SMILES	PUBCHEM_ACTIVITY_OUTCOME	fingerprint	predicted_activity
11	C1CCN(CC1)C2(CCN(CC2)C3CC(=O)N(C3=O)CC4=CC=C(C...	Active	[0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
55	COC1=C(C=C(C=C1)CCNC(=O)/C=C/C2=CC=CO2)OC	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
60	COC1=CC=C(C=C1)N2C(=O)CC(C2=O)N3CCN(CC3)C4=NC5...	Active	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
112	CC1CN(CCN1C2=CC=CC(=C2)C)C(=O)C3=NOC4=C3COC5=C...	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
114	CCC(=O)NC1=NN2C(C=C(NC2=N1)C3=CC=CC=C3)C4=CC=C...	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
...	...	...	...	...
359752	C1=CC=C2C(=C1)C(=CC=[N+]2[O-])SC3=C(C=C(C=C3)[...	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
359756	COC1=CC=CC=C1NC(CC(=O)C2=CC=CC=C2)C(=O)OCC(=O)...	Active	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
359792	C=CCN1C(=C(C2=NC3=CC=CC=C3N=C21)S(=O)(=O)C4=CC...	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
359819	CC1=C(C2=CC=CC=C2N1)C(C3=CC=NC=C3)N4CCN(CC4)C5...	Active	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1
359826	CCN1[C@H](CN=C1NC2=CC=CC=C2)CC3=CC=CC=C3	Active	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1

10784 rows × 4 columns

Testing on a Randomly Selected Row

- Pick a random row from the dataset.
- Prepare its fingerprint for the model.
- Predict and print activity status.
- Show the row's details.

```python
# Select a random row, predict its activity, and display details
random_row = balanced_df.sample(n=1)
X_random = np.array(random_row['fingerprint'].iloc[0]).reshape(1, -1)

prediction = randomforest.predict(X_random)
print(f"Prediction for the random row: {'Active' if prediction[0] == 1
else 'Inactive'}")
print("\nDetails of the randomly selected row:")
display(random_row)
```

OUTPUT : *random_row*

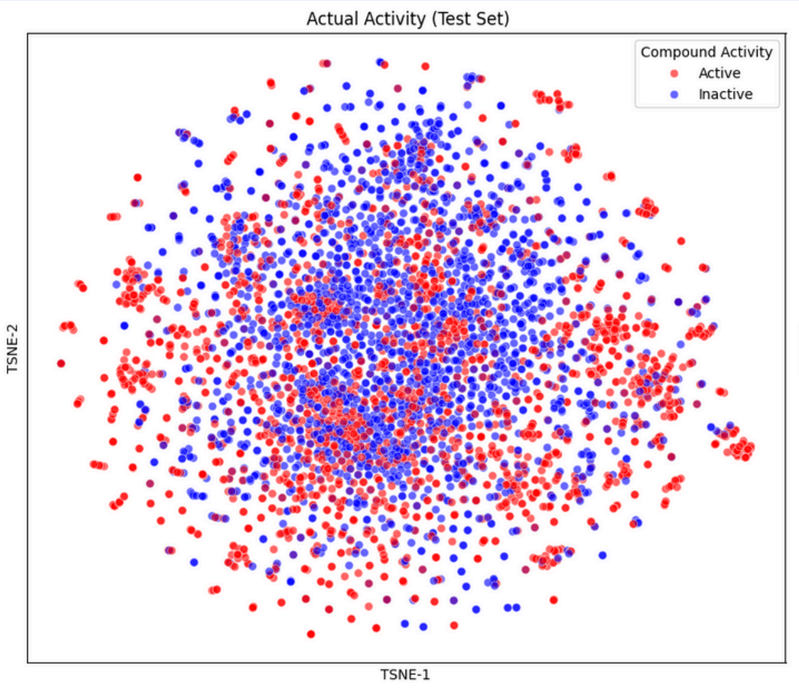
```
Prediction for the random row: Inactive

Details of the randomly selected row:
```

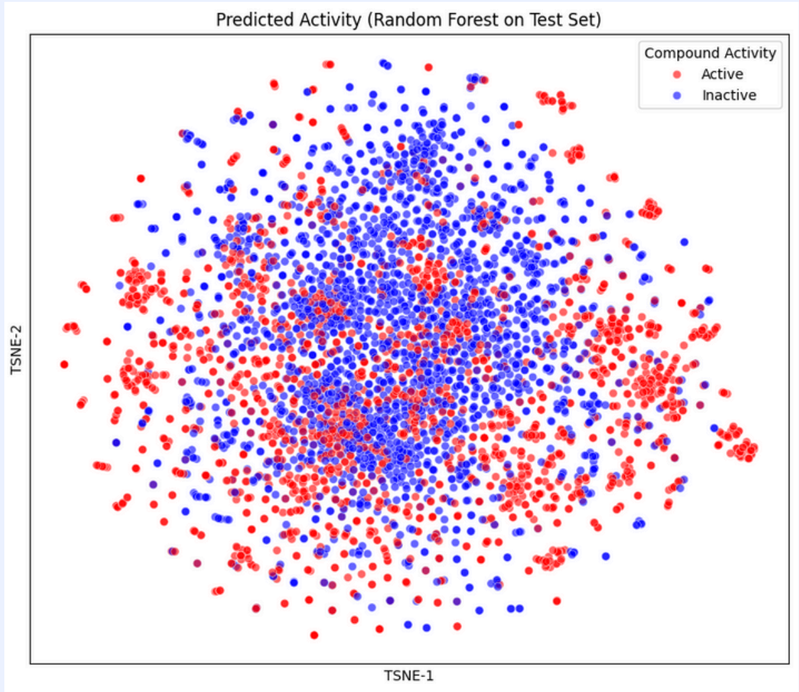
	PUBCHEM_EXT_DATASOURCE_SMILES	PUBCHEM_ACTIVITY_OUTCOME	fingerprint	predicted_activity
302732	CCCNC(=S)NNC(=O)C1=CC(=NC2=CC=CC=C21)C3CC3	Inactive	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0

t-SNE Visualization & Saving Model

- Reduce **X_test** from high dimensions to **2D** using t-SNE.
- Use **perplexity=30** and **n_iter=1000** for optimized mapping.
- Prepare DataFrame with reduced coordinates and **Actual / Predicted** labels.
- Plot **Actual Activity** vs **Predicted Activity** side-by-side.
- **Red** = Active and **Blue** = Inactive for clear class distinction.
- Save trained Random Forest model.

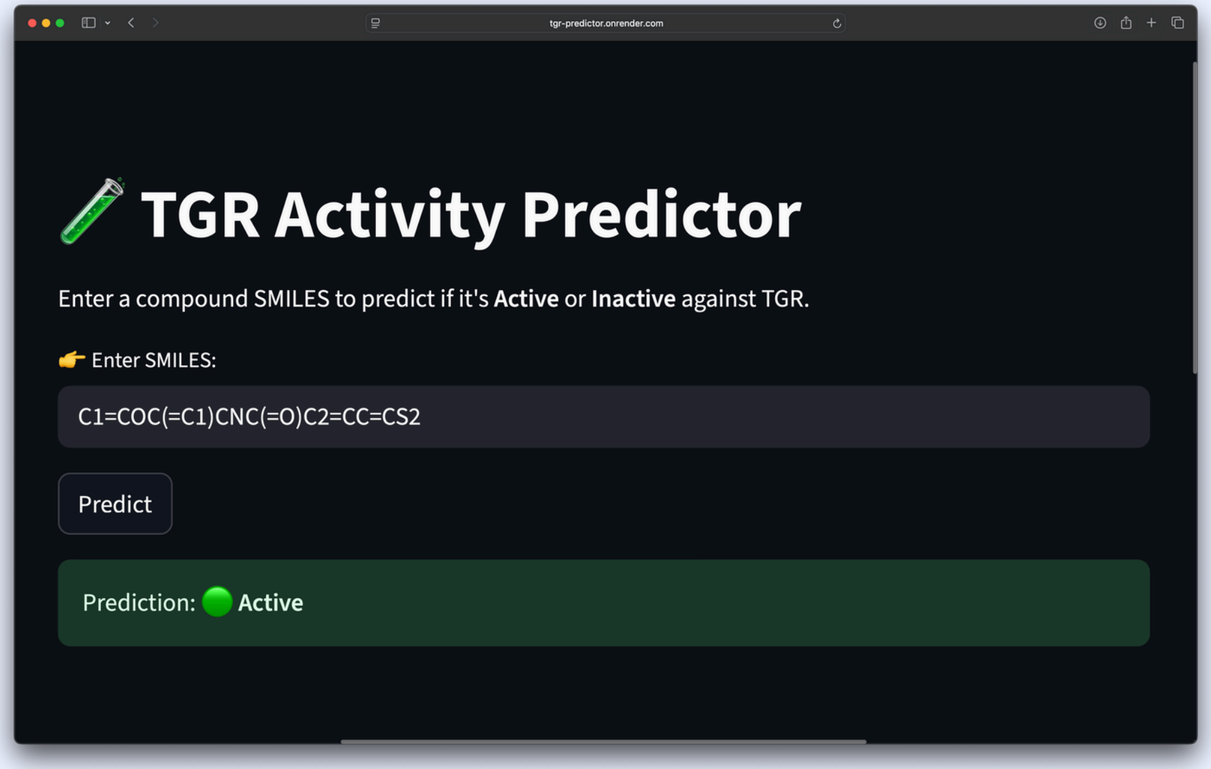
```python
#Initialize t-SNE
tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
#Reduce to 2D
X_test_reduced = tsne.fit_transform(X_test)

viz_df_test = pd.DataFrame(X_test_reduced, columns=["TSNE-1", "TSNE-2"])
viz_df_test['Actual Activity'] = y_test.reset_index(drop=True).replace({1: 'Active', 0: 'Inactive'})
viz_df_test['Predicted Activity'] = pd.Series(y_pred).replace({1: 'Active', 0: 'Inactive'})

fig, axes = plt.subplots(1, 2, figsize=(16, 7))
sns.scatterplot(data=viz_df_test, x='TSNE-1', y='TSNE-2', hue='Actual Activity',
 palette={'Active': 'red', 'Inactive': 'blue'}, alpha=0.6, ax=axes[0])
axes[0].set_title(" Actual Activity (Test Set)")
axes[0].legend(title="Compound Activity")
axes[0].grid(True)
axes[0].set_xticks([])
axes[0].set_yticks([])

sns.scatterplot(data=viz_df_test, x='TSNE-1', y='TSNE-2', hue='Predicted Activity',
 palette={'Active': 'red', 'Inactive': 'blue'}, alpha=0.6, ax=axes[1])
axes[1].set_title(f"Predicted Activity (Random Forest on Test Set)")
axes[1].legend(title="Compound Activity")
axes[1].grid(True)
axes[1].set_xticks([])
axes[1].set_yticks([])

plt.tight_layout()
plt.show()

joblib.dump(model, 'random_forest_tgr.pkl')
```

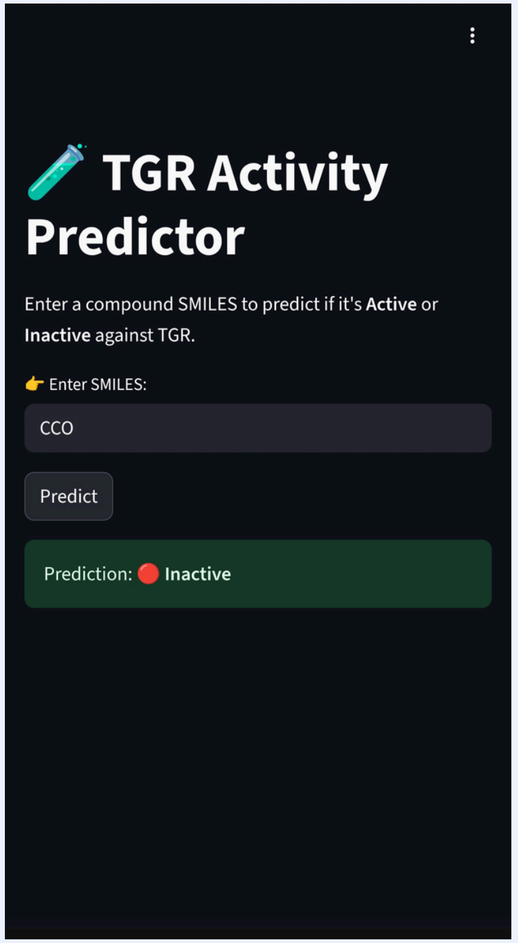




- **TSNE-1** and **TSNE-2** are components generated by **t-SNE**, a dimensionality reduction technique.
- These axes are not real chemical or biological properties.
- They represent compressed 2D coordinates derived from high-dimensional molecular fingerprints
    - **X-axis (TSNE-1)** : First reduced feature from fingerprint
    - **Y-axis (TSNE-2)**: Second reduced feature
- The position of a dot reflects chemical similarity — similar compounds appear closer and form clusters

ML Model Deployment





Website : - https://tgr-predictor.onrender.com

Model Evolution : From Linear Simplicity to Ensemble Power

- Started with **Logistic Regression** as baseline
  - Simple, linear decision boundary.
  - Quick and interpretable for initial testing.
- **Limitations :**
  - Couldn’t capture complex, non-linear patterns.
  - Moderate performance in classification report.
- Switched to **Random Forest**
  - Uses many decision trees working together to make better predictions.
  - Handles non-linearity and feature interactions well.
- Performance Gains :
  - **Higher** precision, recall, and F1-score.
  - t-SNE plots remained **visually similar**.
  - Clear **improvement** in predictions.

Report Comparison between Models

REPORT : *Logistics Regression*

Classification Report:

	precision	recall	f1-score	support
0	0.79	0.80	0.79	2157
1	0.80	0.79	0.79	2157
accuracy			0.79	4314
macro avg	0.79	0.79	0.79	4314
weighted avg	0.79	0.79	0.79	4314

	Accuracy	Precision	Recall	F1 Score	ROC-AUC Score
0	0.792768	0.797176	0.78535	0.791219	0.871381

REPORT : *Random Forest*

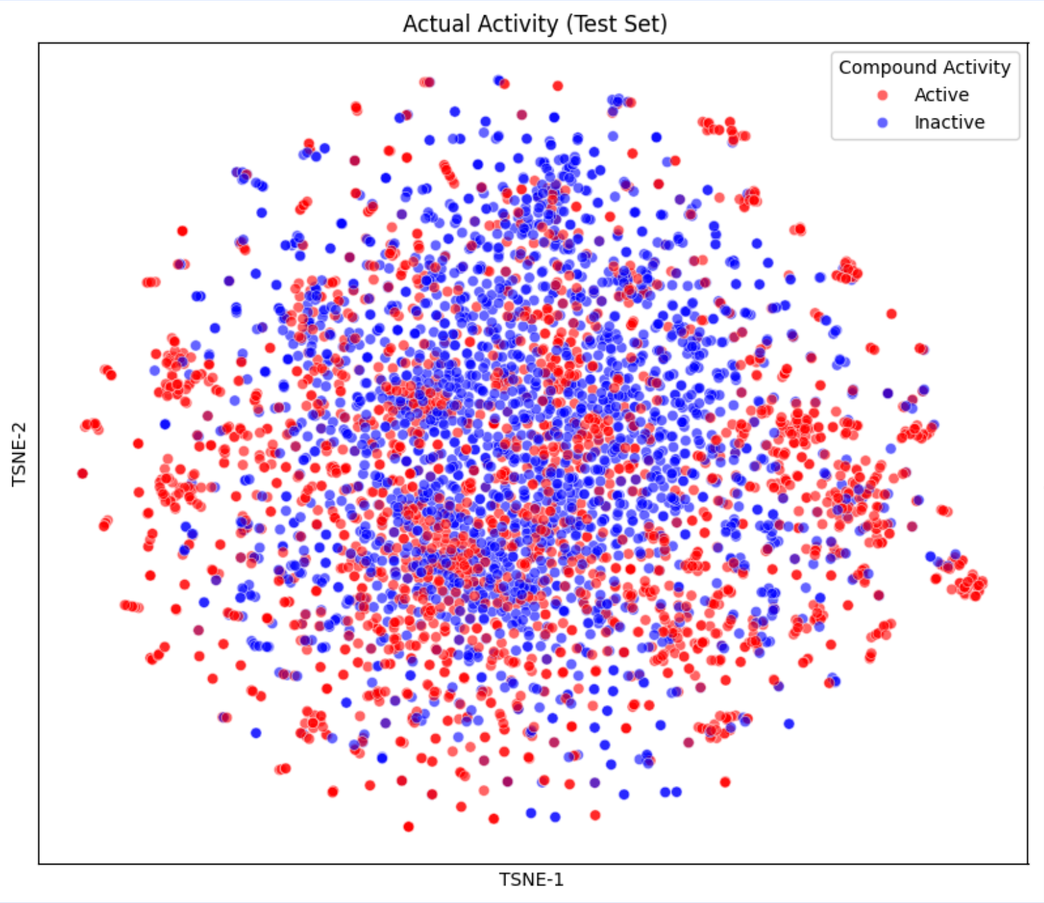
Classification Report:

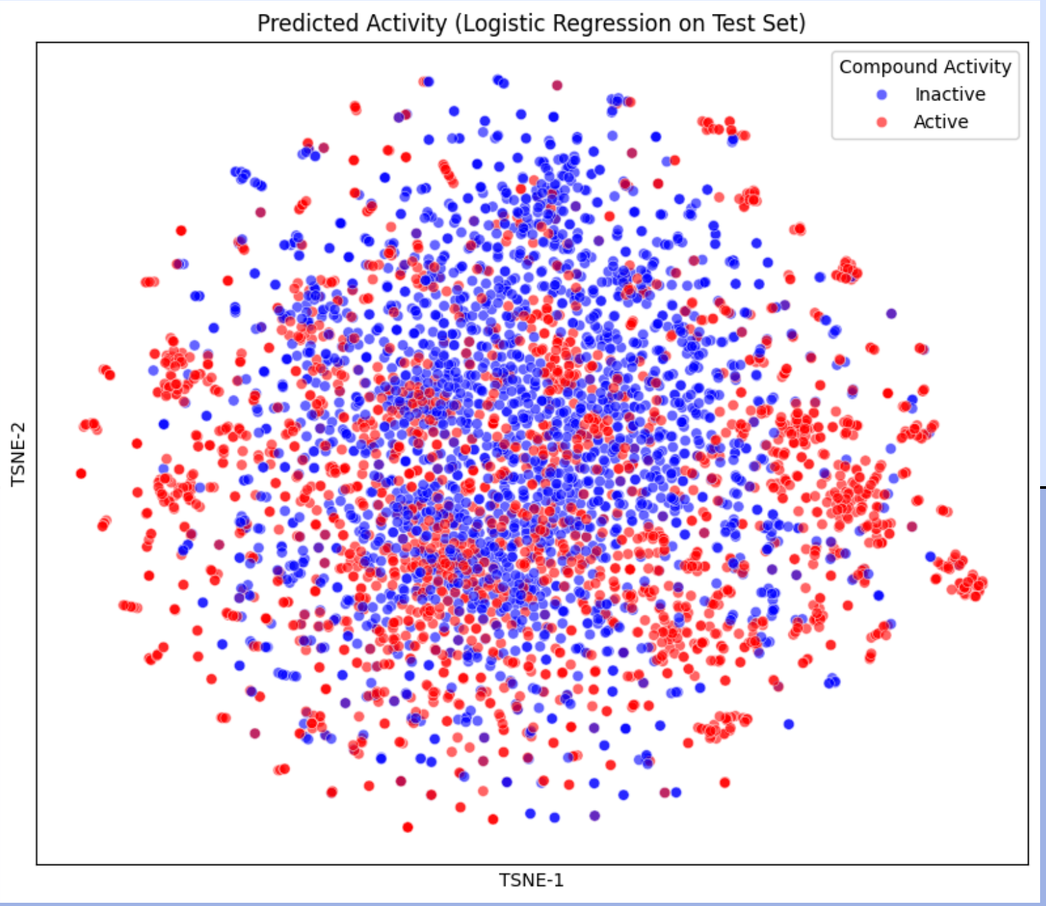
	precision	recall	f1-score	support
0	0.82	0.86	0.84	2157
1	0.86	0.81	0.83	2157
accuracy			0.84	4314
macro avg	0.84	0.84	0.84	4314
weighted avg	0.84	0.84	0.84	4314

	Accuracy	Precision	Recall	F1 Score	ROC-AUC Score
0	0.839128	0.857352	0.81363	0.834919	0.907226

Plots Comparison between Models

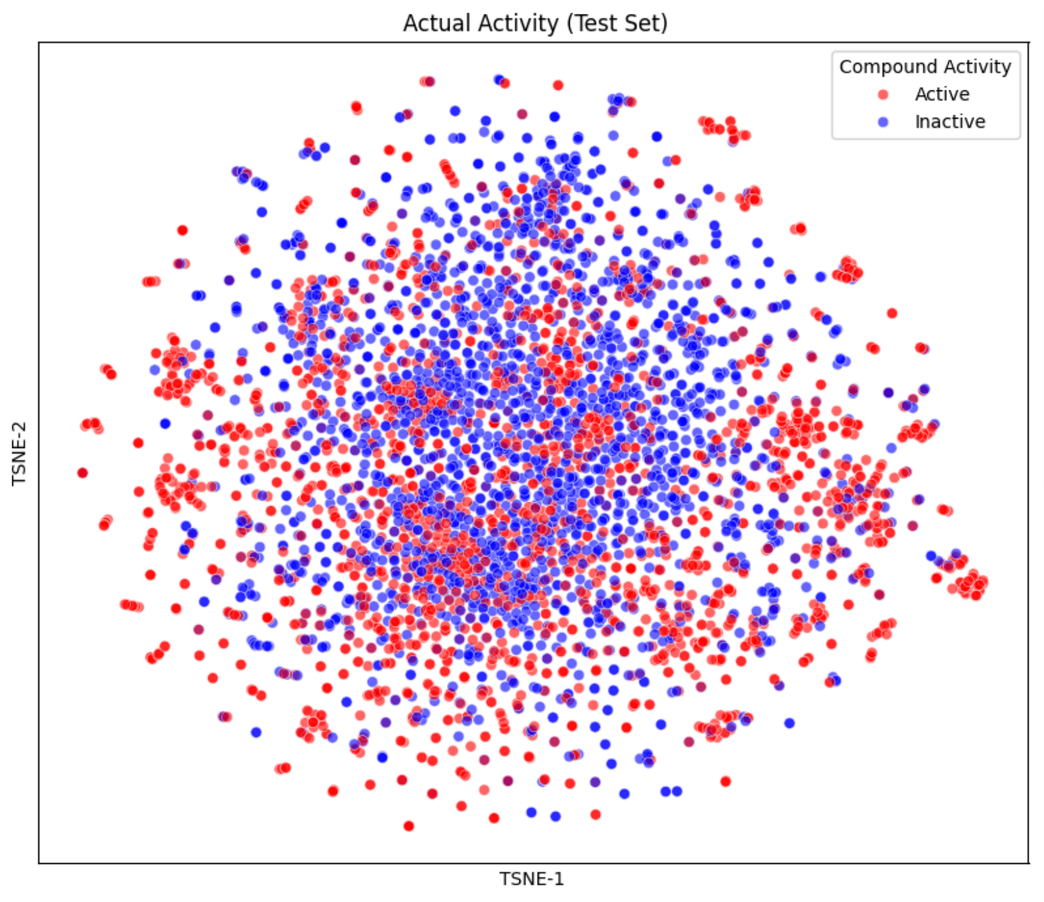
PLOTS : *Logistics Regression*

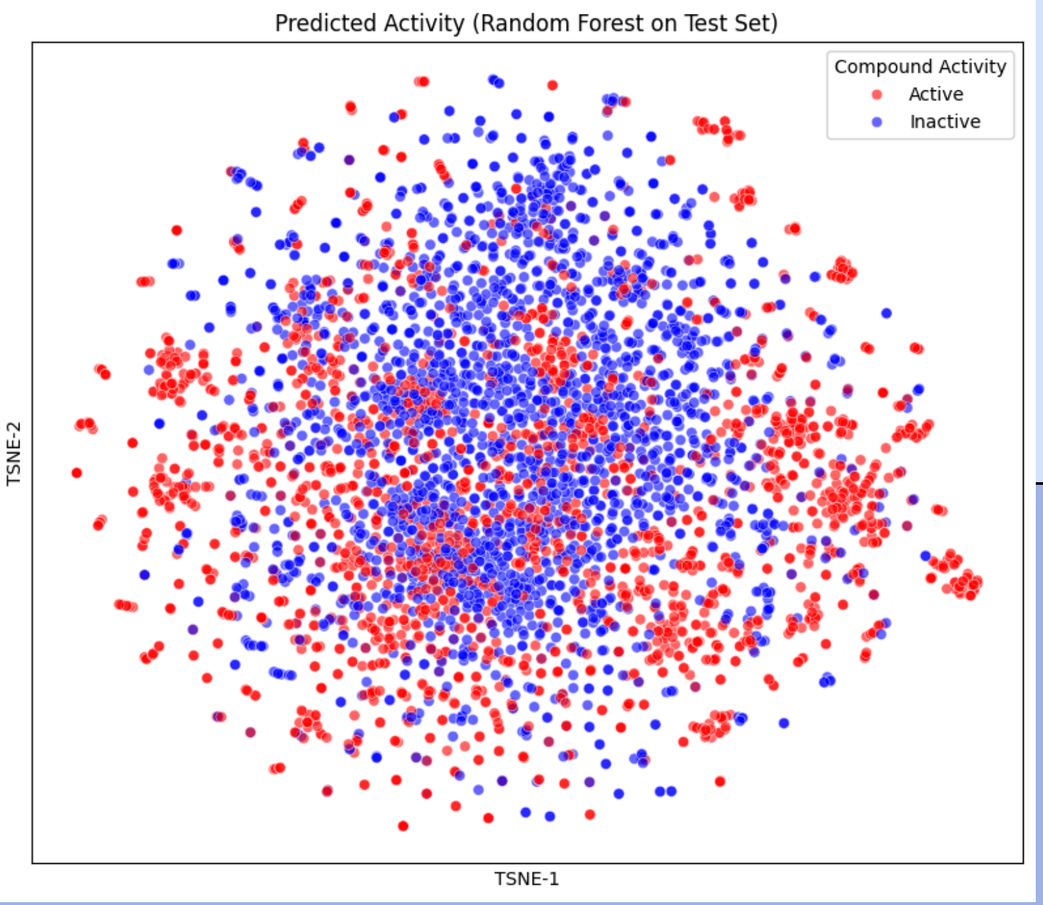




Plots Comparison between Models

PLOTS : *Random Forest*





Random Forest for Virtual Screening: Pros and Cons

Advantages

- Captures non-linear relationships.
- Robust to noise and outliers.
- Reduces overfitting.
- Gives feature importance.

Disadvantages

- More computationally expensive.
- Higher memory usage.
- Less interpretable.
- Can overfit without tuning.

REFERENCES

- https://www.who.int/health-topics/schistosomiasis#tab=tab_1
- https://www.yourgenome.org/theme/what-is-schistosomiasis/
- https://www.cdc.gov/dpdx/schistosomiasis/index.html
- https://pubchem.ncbi.nlm.nih.gov/bioassay/485364
- https://www.ncbi.nlm.nih.gov/books/NBK554434/
- https://www.mdpi.com/2414-6366/8/3/144
- https://pmc.ncbi.nlm.nih.gov/articles/PMC9521826/