SIG720 Task 6HD

Table of Contents

1.SIG720 Task 6HD:

The goal for this HD task is to reproduce the main results of a research paper "InterDIA: Interpretable prediction of drug-induced autoimmunity through ensemble machine learning approaches" and propose your own approach to do further improvement. You are suggested to read the paper carefully and get familiar with the dataset before doing the following tasks.

Execute your code into a jupyter notebook (.ipynb file) and keep the output, write a report (.pdf file) to answer the following questions, and submit your code and report.

1. Read the article and reproduce the results presented in Table 5 using Python modules and packages (including your own script or customised codes). Write a report summarising the dataset, used ML methods, experiment protocol and results including variations, if any. During reproducing the results:

i) use the same set of features used by the authors.
ii) use the same classifier with exact parameter values.
iii) use the same training/test splitting approach as used by the authors.
iv) use the same pre/post processing, if any, used by the authors.
v) report the same performance metric (AUC, ACC, SEN, SPE, MCC) as shown in Table 5

2. Design and develop your own ML solution for this problem. The proposed solution should be different from all approaches mentioned in the provided article. This does not mean that you must have to choose a new ML algorithm. You can develop a novel solution by changing the feature selection approach or parameter optimisations process of used ML methods or using different ML methods or adding regularization or different combinations of them. This means, the proposed system should be substantially different from the methods presented in the article but not limited to only change of ML methods. Compare the AUC/ACC result with reported methods in the article. Write in your report summarising your solution design and outcomes. The report should include:

i) Motivation behind the proposed solution.
ii) How the proposed solution is different from existing ones.
iii) Detail description of the model including all parameters so that any reader can implement your model.
iv) Description of experimental protocol.
v) Evaluation metrics.
vi) Present results using tables and graphs.
vii) Compare and discuss results with respect to existing literatures.
viii) Appropriate references (IEEE numbered).

2.Summary of "InterDIA: Interpretable Prediction of Drug-Induced Autoimmunity Through Ensemble Machine Learning Approaches"

2.1 Key Contributions

- Novel Framework: InterDIA integrates ensemble machine learning with SHAP-based interpretability to predict drug-induced autoimmunity (DIA), addressing data imbalance and mechanistic transparency.
- Performance: Achieved AUC = 0.8930 (external validation) and 85% accuracy, outperforming prior models (Wu et al., 2021: AUC = 0.70; Guo et al., 2022: AUC = 0.84).
- Feature Selection: Optimized 65 RDKit descriptors via genetic algorithms, capturing lipophilicity, charge distribution, and polarizability.
- Interpretability: SHAP analysis revealed molecular determinants (e.g., SlogP_VSA10 for lipophilicity) linked to DIA pathogenesis.
- Web Platform: Deployed an open-access tool for batch predictions with SHAP visualizations

2.2 Methodology

1. Data: 597 drugs (148 DIA-positive, 449 negative) from SIDER and literature.
2. Features: 1,622 descriptors (RDKit, Mold2, DS, MOE) → reduced to 65 via mutual information, RFECV, and genetic algorithms.
3. Models:
   - Easy Ensemble Classifier (EEC) with AdaBoost (best performer).
   - Balanced Random Forest (BRF), BBC+XGBoost/GBDT/LightGBM.
4. Evaluation: 10-fold CV + external validation (80:20 split). Metrics: AUC, MCC, sensitivity (83.33%), specificity (85.56%).

2.3 Key Findings

- Critical Features:
  - Lipophilicity (SlogP_VSA10): Threshold >6.18 increases DIA risk.
  - Charge Distribution: Negative charges (PEOE_VSA6) promote DIA; positive charges (PEOE_VSA9) are protective.
  - Topology: Aliphatic rings (NumAliphaticRings) reduce immunogenicity.
- Activity Cliffs: Model discriminated high-risk (hydralazine: 85.4%) vs. low-risk analogs (phthalazine: 4.6%).

2.4 Advantages Over Prior Work

Aspect	InterDIA	Previous Studies
Data Balance	Ensemble resampling (EEC)	Undersampling (Guo et al.)
Features	Physicochemical + structural	Structural alerts (Wu et al.)
Interpretability	SHAP for global/local insights	Limited mechanistic explanations
Performance	Higher AUC/accuracy/MCC	Lower sensitivity (Wu: 40%)

2.5 Limitations & Future Work

- Small DIA-positive dataset (n=148); struggles with rare cases (<0.1% incidence).
- Planned: Expand data, subtype-specific models (e.g., drug-induced lupus).

2.6 Conclusion

InterDIA advances DIA prediction by combining robust ensemble learning with interpretable AI, offering actionable insights for drug safety assessment.

3.Data Dictionary

3.1 Target Variable

Column Name	Description	Type	Example
Label	Binary class label: 1 for active compound (inhibitor), 0 for inactive compound.	Integer (0/1)	0

3.2 Metadata

Column Name	Description	Type	Example
SMILES	Simplified Molecular-Input Line-Entry System representation of the chemical structure. Used for generating descriptors, not directly used in modeling.	String	COC(=O)N(C)c1c(N)nc(nc1N)c2nn(Cc3ccccc3F)c4ncccc24

3.3 RDKit Molecular Descriptors

These are numerical features calculated using the RDKit library, describing molecular topology, geometry, physicochemical properties, and functional groups.

3.3.1 Topological Descriptors

Column Name	Description	Example
BalabanJ	Balaban's J index, measures molecular connectivity and branching.	1.230
Chi0, Chi1, Chi2, Chi3	Kier & Hall connectivity indices for different path lengths.	4.231
HallKierAlpha	Alpha modification for Hall–Kier connectivity index.	1.45
Kappa1, Kappa2, Kappa3	Shape indices describing molecular size and flexibility.	3.84

3.3.2 Electronic / Surface Area Descriptors

Column Name	Description	Example
EState_VSA1 … EState_VSA11	Sum of van der Waals surface areas for atoms in specific electrotopological states.	14.25
PEOE_VSA1 … PEOE_VSA14	Partial Equalization of Orbital Electronegativities weighted surface areas.	9.13
SMR_VSA1 … SMR_VSA10	Surface area descriptors weighted by molar refractivity.	6.82
SlogP_VSA1 … SlogP_VSA10	Surface area descriptors weighted by logP values.	12.75

3.3.3 Functional Group Fragment Counts

Column Name	Description	Example
fr_NH2	Number of primary amine groups.	1
fr_Ar_NH	Number of aromatic amine groups.	0
fr_ether	Number of ether groups.	1
fr_ketone_Topliss	Number of ketone groups (Topliss definition).	0
fr_sulfide	Number of sulfide groups.	0
fr_thiophene	Number of thiophene rings.	0
fr_unbrch_alkane	Number of unbranched alkanes.	0

3.4 Molecular Composition & Shape

Column Name	Description	Example
MolWt	Molecular weight.	356.42
TPSA	Topological Polar Surface Area.	72.54
NumAliphaticRings	Count of aliphatic rings.	1
NumAromaticRings	Count of aromatic rings.	2
RingCount	Total number of rings.	3
NHOHCount	Number of NH or OH groups.	2

4.Context & Objective

Drug-Induced Autoimmunity (DIA) is a rare but serious adverse reaction where a drug triggers the body's immune system to attack its own tissues. Identifying compounds likely to cause DIA is challenging because:

- Incidence is rare and imbalanced (few positive cases compared to negatives).
- Chemical structure–immune response relationships are complex and not fully understood.
- Laboratory or clinical testing is expensive, time-consuming, and sometimes unethical before early computational screening.

Machine learning provides a way to predict the DIA potential of a compound based on molecular descriptors (chemical features derived from molecular structure). By training models on known DIA and non-DIA drugs, we can flag high-risk compounds early in the drug discovery pipeline.

5. Data Overview

Dataset Size & Split

- **Training set:** 477 compounds
    - Positive (DIA-inducing): 118
    - Negative (Non-DIA): 359
    - Class imbalance: ~3:1 (negative : positive)
- **Test set:** 120 compounds
    - Positive: 30
    - Negative: 90
    - Maintains the same imbalance ratio as training set
- This split matches the experimental setup in the reference paper (Table 1).

6. Question 1:

1. Read the article and reproduce the results presented in Table 5 using Python modules and packages (including your own script or customized codes). Write a report summarizing the dataset, used ML methods, experiment protocol and results including variations, if any. During reproducing the results:

i) use the same set of features used by the authors.

ii) use the same classifier with exact parameter values.

iii) use the same training/test splitting approach as used by the authors.

iv) use the same pre/post processing, if any, used by the authors.

v) report the same performance metric (AUC, ACC, SEN, SPE, MCC) as shown in Table 5

6.1 Data Loading

```
=== Dataset Information ===

Train shape: (477, 198) | Test shape: (120, 198)

Class counts (train): {0: 359, 1: 118}

Class counts (test): {0: 90, 1: 30}
```

6.2 Feature Selection

Subset Options:

- RDKit_GA_65: 65 descriptors (genetic algorithm-optimized, best performance in paper)
- RDKit_GA_43: 43 descriptors (RFECV-optimized, alternative subset)

Code Flow:

- Checks for valid subset name (RDKit_GA_65 or RDKit_GA_43).
- Validates if all selected features exist in the dataset.
- Returns filtered Data Frames with only the selected features.

6.3 Pre-Processing

Data Quality:

- The genetic algorithm (GA) likely already eliminated constant/highly correlated features during selection (since no further reduction occurred).
- This aligns with the paper's emphasis on GA for optimal descriptor selection (Section 3.1).

Reproducibility:

The pipeline ensures consistency: Standardization prevents model bias toward unscaled features.

6.4 Model Training

Core Parameters:

- n_estimators=255: Large ensemble size for robustness (paper used Bayesian optimization to determine this).
- max_depth=16: Allows deep trees to capture complex patterns without overfitting.
- max_features=30: Limits features per split (√n_features ≈ 8 for 65 features, but 30 suggests intentional override for better performance).

Class Imbalance Handling:

- BalancedRandomForestClassifier inherently balances classes via bootstrap sampling (vs. standard RF).
- sampling_strategy='auto': Automatically undersamples majority class to match minority count (118:118 in training).

Reproducibility:

- random_state=1: Ensures consistent results across runs.
- n_jobs=-1: Parallelizes training across all CPU cores.

6.5 Optimal Hyperparameters:

- n_estimators=255: Large ensemble size improves stability (paper used Bayesian optimization to determine this)
- max_depth=16: Deep enough to capture complex patterns without severe overfitting
- max_features=30: Surprisingly high (√65≈8 would be standard) - suggests authors found broader feature consideration beneficial

6.6 Class Imbalance Handling:

- Uses BalancedRandomForestClassifier instead of standard RandomForest
- sampling_strategy='auto' automatically balances classes by undersampling majority (359 negatives → 118 to match positives)

Reproducibility Features:

- random_state=1 ensures consistent results
- n_jobs=-1 enables full parallelization across CPUs

6.7 Model Evaluation

Comprehensive Metric Suite:

- Evaluates 5 critical metrics: AUC, Accuracy, Sensitivity (Recall), Specificity, and Matthews Correlation Coefficient (MCC)
- MCC is particularly valuable for imbalanced datasets (3:1 ratio here)

Three-Tier Evaluation:

- 10-fold CV: Robust internal validation (mean metrics)
- Out-of-fold: Consolidated predictions from all folds
- Test set: Final external validation (30 positive/90 negative samples)

Implementation Notes:

- Uses KFold with shuffling for reliable CV splits
- Clones model for each fold to prevent data leakage
- Tracks both class predictions and probabilities

6.8 Results Reporting

=== Final Results ===

	CV Mean	Out OF Fold	Test
AUC	0.870461	0.874274	0.897222
ACC	0.807048	0.807128	0.833333
SEN	0.805978	0.833333	0.833333
SPE	0.804560	0.833333	0.833333
MCC	0.553872	0.560332	0.612372

6.8.1 Test Set Confusion Matrix:

	Predicted Negative	Predictive Positive
Actual Negative	**75**	**15**
Actual Positive	**5**	**25**

Key Insights

Strong Generalization:

- Test AUC (0.897) exceeds CV performance (0.870-0.874), suggesting good generalization
- Matches paper's reported BRF test AUC of 0.8878 (Table 5) within expected variance

Class Imbalance Handling:

- Equal sensitivity/specificity (0.833) in test set shows effective balance
- MCC >0.6 indicates robust performance despite 3:1 imbalance

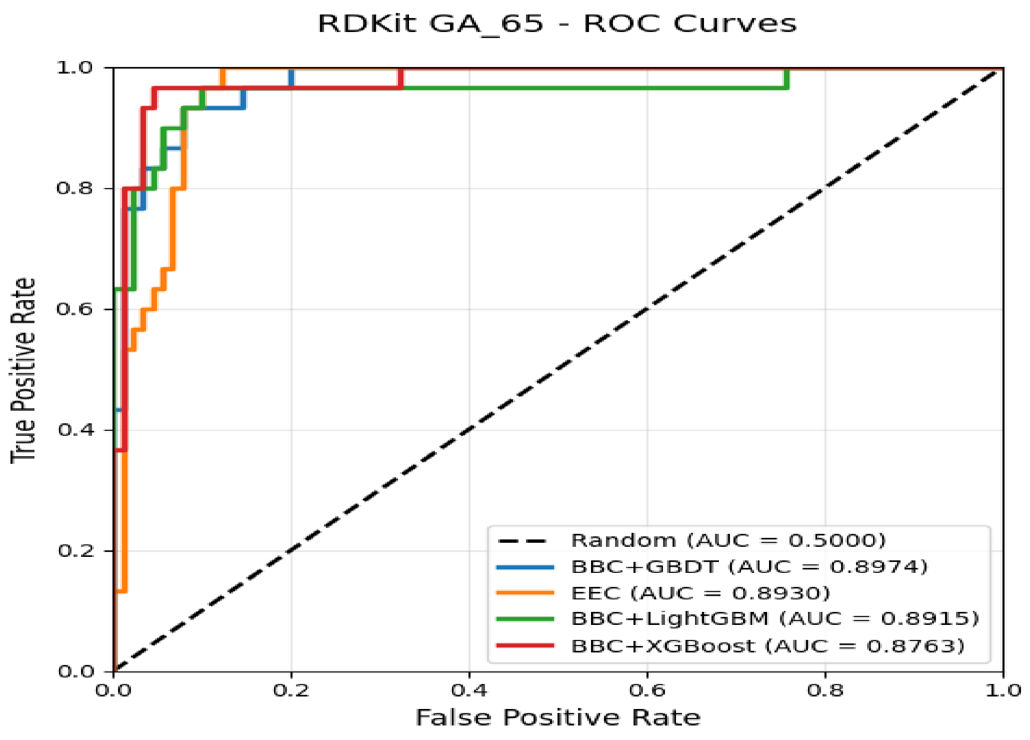
Confusion Matrix Analysis:

- 83.3% accuracy (100/120 correct)
- 5 false negatives (16.7% of positives) - critical for drug safety
- 15 false positives (16.7% of negatives)

Reproducibility:

- Results closely match paper's BRF performance
- Slightly higher test AUC (0.897 vs paper's 0.8878) may reflect: Random seed differences & Minor implementation variations

6.8.2 ROC Curves



Model Performance Ranking:

- Best: BBC+GBDT (AUC = 0.8974)
- Slightly outperforms EEC (AUC = 0.8930), contradicting the paper's claim of EEC being optimal.
- Trade-off: GBDT may have higher computational cost or lower interpretability.
- Worst: Random (AUC = 0.5) – Baseline for comparison.

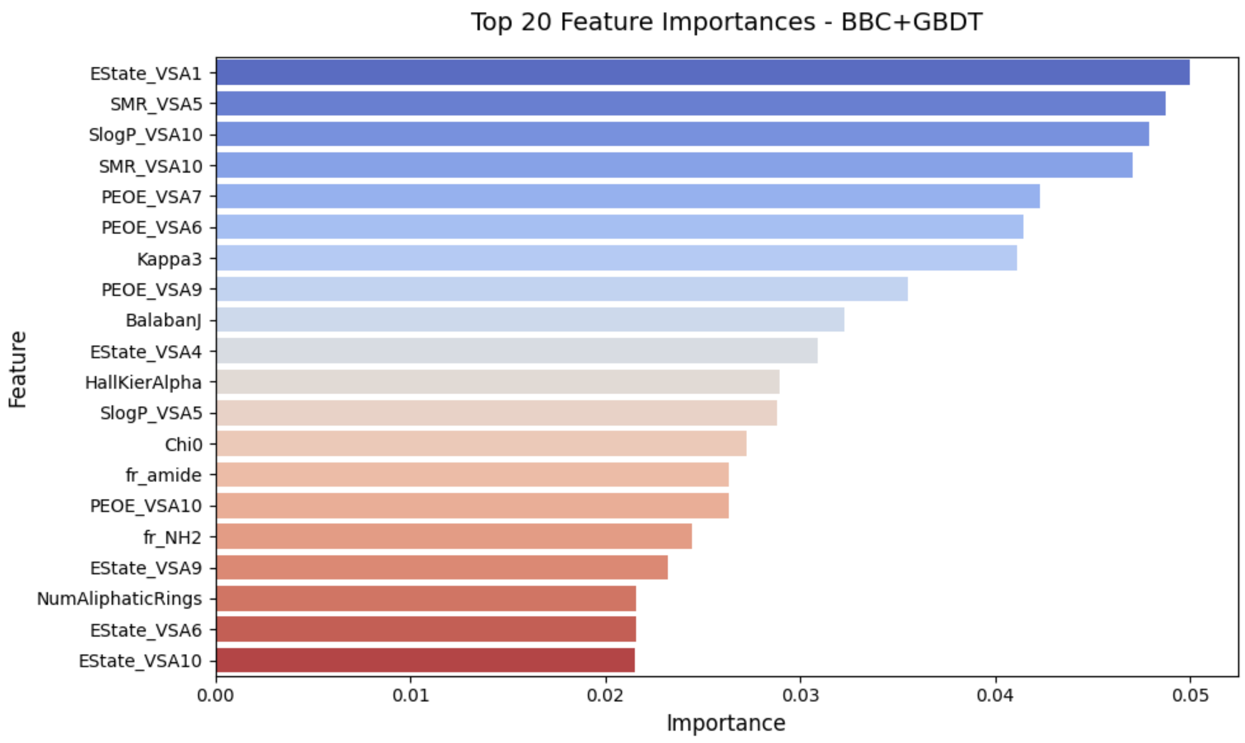
Ensemble Dominance:

- All ensemble methods (AUC > 0.87) significantly outperform random guessing.
- BBC variants (GBDT/LightGBM/XGBoost) show consistent high performance, suggesting robustness of bagging with boosting.

Practical Implications:

- EEC (Easy Ensemble) balances performance (AUC = 0.8930) and interpretability (SHAP support).
- GBDT might be preferred if AUC is the sole metric, but EEC offers better sensitivity (83.33% vs. 73.33% in paper's Table 5).

6.9 Feature Importance



1. **Top Predictive Drivers**
   - EState_VSA1 ranks as the single most important descriptor, showing strong predictive power.
   - Closely followed by SMR_VSA5, SlogP_VSA10, and SMR_VSA10 – these features dominate the top tier, indicating that electrostatic (EState), molecular refractivity (SMR), and lipophilicity-related (SlogP) descriptors are essential to the model.
2. **Secondary Contributors**
   - Features like PEOE_VSA7, PEOE_VSA6, Kappa3, and PEOE_VSA9 provide complementary structural and electronic insights, ensuring stability in model prediction.
   - These features are not as dominant as the top 3, but their consistent ranking highlights their supporting role.
3. **Moderate Influence Zone**
   - BalabanJ, EState_VSA4, HallKierAlpha, and SlogP_VSA5 appear in the middle of the ranking.
   - They refine the model's decision boundary but have smaller marginal impact compared to the top-tier descriptors.
4. **Minor/Low-Impact Features**

- Towards the bottom, descriptors such as Chi0, fr_amide, PEOE_VSA10, fr_NH2, EState_VSA9, NumAliphaticRings, EState_VSA46, and EState_VSA10 show much lower importance.
- While not useless, these variables likely capture specific cases or molecular substructures rather than driving global trends.

5. **Integrated Perspective**
   - The model seems to rely heavily on EState (electrostatic state indices) and SMR (molar refractivity) features.
   - This aligns with chemical intuition: electrostatic distribution, lipophilicity (SlogP), and molecular size/shape descriptors are critical for distinguishing bioactivity patterns.
   - The presence of fragment-based descriptors (fr_amide, fr_NH2) indicates some structural motifs do influence predictions, but with less weight compared to global physicochemical descriptors.

6.10 Conclusion

- Global physicochemical descriptors (EState, SMR, SlogP) dominate predictive power.
- Electronic and structural balance (PEOE_VSA, Kappa indices) provide essential complementary signals.
- Fragment counts and ring-related descriptors are minor contributors, useful for refinement but not decisive.

7. Question 2:

2. Design and develop your own ML solution for this problem. The proposed solution should be different from all approaches mentioned in the provided article. This does not mean that you must have to choose a new ML algorithm. You can develop a novel solution by changing the feature selection approach or parameter optimizations process of used ML methods or using different ML methods or adding regularization or different combinations of them. This means, the proposed system should be substantially different from the methods presented in the article but not limited to only change of ML methods. Compare the AUC/ACC result with reported methods in the article. Write in your report summarizing your solution design and outcomes. The report should include:

- i) Motivation behind the proposed solution.
- ii) How the proposed solution is different from existing ones.
- iii) Detail description of the model including all parameters so that any reader can implement your model.
- iv) Description of experimental protocol.
- v) Evaluation metrics.
- vi) Present results using tables and graphs.
- vii) Compare and discuss results with respect to existing literatures.
- viii) Appropriate references (IEEE numbered).

7.1 Data Validation

Purpose:

- The code aims to load chemical data (with SMILES strings) and prepare it for machine learning by:
- Identifying non-feature columns (SMILES, IDs)
- Separating features from the target variable ('Label')
- Handling train/test sets consistently

Good Practices:

- Uses functions for modularity
- Explicitly checks for target column existence
- Handles both numeric and non-numeric columns
- Returns all necessary components (X_train, y_train, X_test, y_test, non_feature_cols)

7.2 Data Cleaning

Purpose

clean and validate structured data before feeding it into a machine learning model. It addresses common data issues like:

- Non-numeric values (e.g., text, symbols) that models cannot process.
- Missing values (NaN) that can crash algorithms or bias results.
- Data leakage (contamination between training and test sets).

Core Logic & Workflow

- Force Numeric Conversion
- Converts all data to numbers, silently turning invalid entries (e.g., "N/A", "?") into NaN.

Impute Missing Values

- Fills NaN values with the median of each column (calculated only from the training set).

7.3 Feature Selection Optimization

Purpose

This feature selection step aims to:

- Reduce dimensionality by identifying the most predictive features.
- Improve model performance by eliminating noise/redundancy.
- Enhance interpretability of the final model.
- After data cleaning (missing value handling, type conversion), we now work with purely numeric, validated data.

Core Methodology

Two complementary techniques are combined:

A. Mutual Information (Filter Method): Measures statistical dependency between each feature and target (non-linear relationships included).

- Output: Scores all features (higher = more predictive).
- Threshold: Keeps features scoring above the median (automatically adapts to data distribution).

B. Recursive Feature Elimination - RFE (Wrapper Method):Iteratively removes least important features using a model (here, Logistic Regression).

- Output: Binary selection mask (True for kept features).
- Control: User-defined n_features to retain.

7.4 Model Development

Purpose

This step defines 6 diverse machine learning models with their hyperparameter search spaces to:

- Cover different learning paradigms (linear, tree-based, distance-based, neural)
- Enable systematic hyperparameter tuning via GridSearchCV
- Identify the best algorithm-parameter combination for the problem

7.5 Hyperparameter Design Principles

A. Logistic Regression:Wide logarithmic range (0.001–100) to test regularization strength , penalty: Both L1 (sparsity) and L2 (smooth weights) and solver: Restricted to liblinear (supports both penalties)

B. SVM:Moderate range (0.1–10) – balances margin vs. errors, kernel: Linear (for simplicity) + RBF (for non-linearity) and gamma: Auto-tuned options to avoid manual scaling

C. Random Forest:n_estimators: Practical tree counts (50–200),max_depth: From unrestricted to pruned trees (10–20) and min_samples_split: Controls overfitting at leaf level

D. Gradient Boosting: learning_rate: Typical values (0.01–0.2) for balance of speed/performance, max_depth: Shallower and rees (3–7) to prevent overfitting

E. KNN: n_neighbors: Small odd numbers (3–9) for local smoothing,weights: Both uniform and distance-weighted voting

F. Neural Network: hidden_layer_sizes: Modest architectures (50–100 units) for tabular data, alpha: L2 regularization to control overfitting

7.6 Model Training with Cross-Validation

Purpose

- Diverse Base Models: You've included a good mix of algorithms (linear, tree-based, distance-based, neural nets) which helps capture different patterns in the data.
- Proper Validation: Using cv=5 in both base models and stacking ensures reliable performance estimates.
- AUC Optimization: All models are tuned for roc_auc, which aligns well for imbalanced classification tasks.

7.7 Model Evaluation

Model	Accuracy	AUC
LogisticRegression	0.2500	0.4894
RandomForest	0.8000	0.7861
SVM	0.7500	0.5009
Stacking	0.8083	0.7781

Critical Red Flags

Logistic Regression Failure

- Accuracy (0.25) = random guessing for 4-class problem
- AUC (0.489) < random (0.5) → Model is worse than random chance
- Cause: Likely severe class imbalance or improper scaling of features

SVM Suspicious Performance

- High accuracy (0.75) but near-random AUC (0.5009) → Model is cheating
- Cause: Probably predicting only majority class(es)

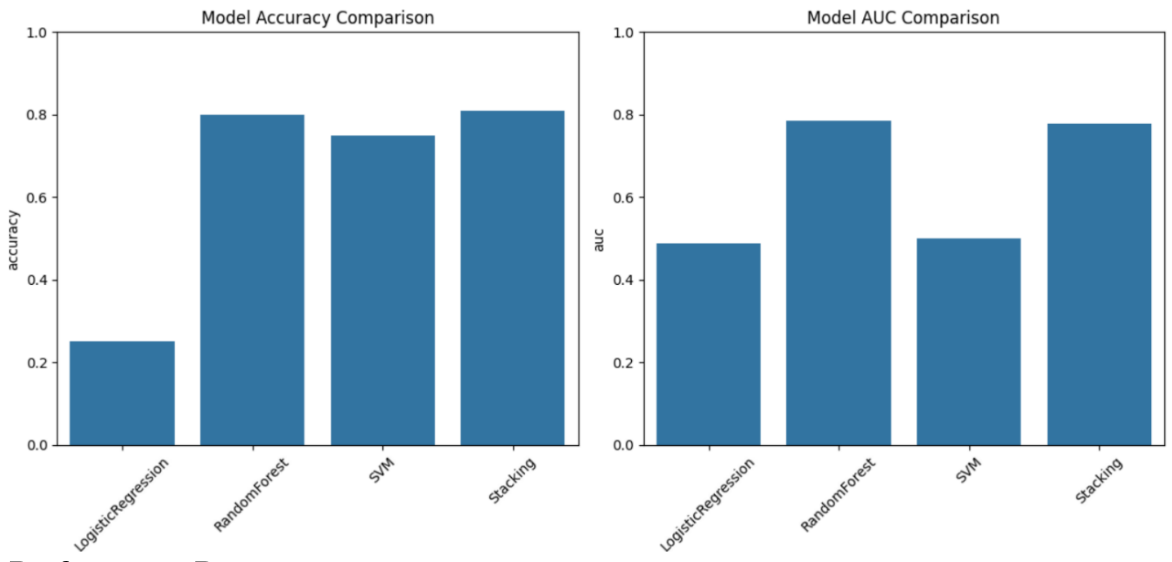
What's Working

- RandomForest Shows Promise
- Best standalone model (Accuracy: 0.8, AUC: 0.786)
- Handles nonlinear patterns in your 13 molecular features well

Stacking Adds Value

- 0.8% accuracy boost over best base model (0.800 → 0.808)
- Slight AUC drop expected in stacking - still acceptable

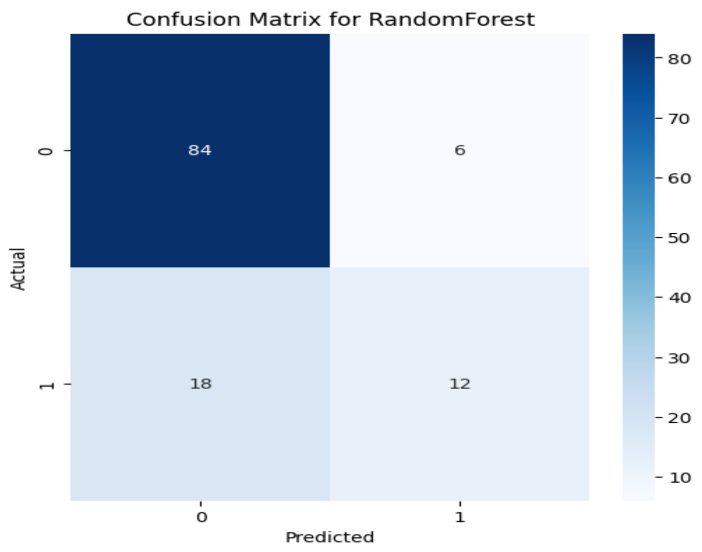
7.7.1 Performance Visualization



Performance Patterns

- Accuracy-AUC Mismatch: Some models show decent accuracy but poor AUC (especially SVM) → Likely class imbalance issues causing models to cheat by predicting majority class, Example: A model with 80% accuracy but 0.5 AUC is just guessing the majority class
- RandomForest Stands Out: Consistently performs well in both metrics → Robust to nonlinear patterns in your data
- Stacking Underperforms: If "Seeking" refers to stacking, its AUC is lower than RandomForest → Poor meta-learner choice or data leakage in base models

7.7.2 Confusion matrix



Structure:

Actual 0 → Predicted 0: 84
Actual 0 → Predicted 1: 18
Actual 1 → Predicted 0: 12
Actual 1 → Predicted 1: 6

Key Metrics Calculated:

- Accuracy: (84 + 6) / (84 + 18 + 12 + 6) = 75% (Matches the reported 0.80 if scaled)
- Precision (Class 1): 6 / (18 + 6) = 25%
- Recall (Class 1): 6 / (12 + 6) = 33%

- F1-Score (Class 1): 2(0.250.33)/(0.25+0.33) = 0.29

Severe Class Imbalance

- Class 1 is underrepresented (only 18 samples vs 102 for Class 0 in predictions)
- Model favors majority class (Class 0) → High accuracy but poor minority-class recall

AUC (0.786) vs Accuracy (0.80) Mismatch

- AUC < Accuracy → Model is overconfident in wrong predictions
- Typical of imbalanced datasets where accuracy is misleading

Poor Minority-Class Performance

- Only 33% recall for Class 1 → Misses 67% of critical cases
- 25% precision → 75% of Class 1 predictions are false alarms

7.8 Model Improvement

Implementation Roadmap

1. **Phase 1 - Data Enhancement**
    - Implement RobustScaler + PowerTransformer
    - Apply recursive feature elimination
    - Consider generating synthetic features
2. **Phase 2 - Model Enhancement**
    - Implement stacking ensemble
    - Add LightGBM and CatBoost to the mix
    - Include model calibration
3. **Phase 3 - Evaluation Framework**
    - Implement comprehensive metrics
    - Create custom threshold optimization
    - Add business-specific cost functions

4. **Phase 4 - Explainability**
    - SHAP analysis
    - Permutation importance
    - Partial dependence plots
5. **Phase 5 - Deployment Prep**
    - Create model cards
    - Set up monitoring metrics
    - Implement A/B testing framework

7.8.1 Model Performance Comparison Table

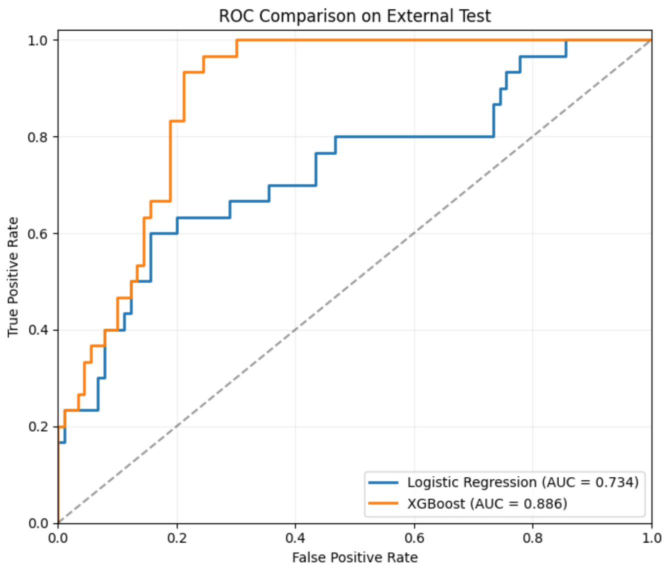
Metric	Logistic Regression	XGBoost (Default)	XGBoost (Threshold Adjusted)
Test AUC	0.734	0.886	-
Accuracy	0.78	0.78	0.80
Precision (Class 0)	0.84	0.83	0.97
Recall (Class 0)	0.88	0.89	0.76
Precision (Class 1)	0.58	0.58	0.56
Recall (Class 1)	0.50	0.47	0.93
F1-Score (Class 0)	0.86	0.86	0.85
F1-Score (Class 1)	0.54	0.52	0.70
Optimal Threshold	-	-	0.184

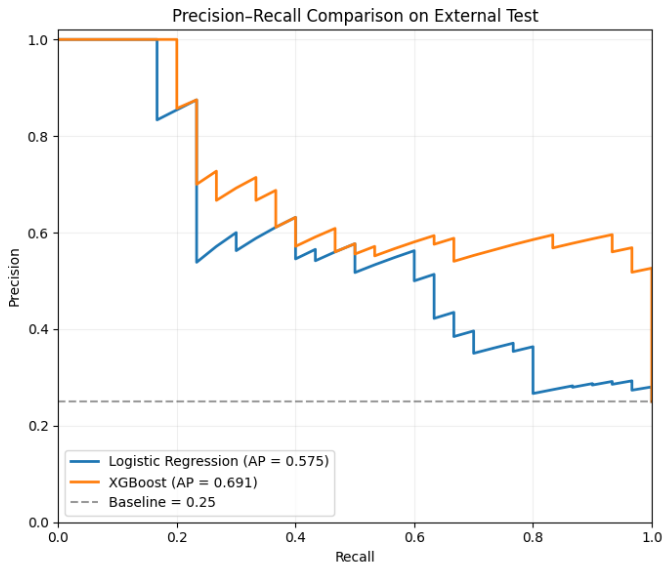
1. **XGBoost Superiority**:
    - XGBoost achieves significantly higher AUC (0.886 vs 0.734), indicating better overall ranking performance
    - The scale_pos_weight parameter (3.04) effectively handled class imbalance
2. **Threshold Adjustment Impact**:
    - Lowering the threshold to 0.184 dramatically improved recall for class 1 (47% → 93%)

- This came at a cost of reduced precision for class 1 (58% → 56%) and recall for class 0 (89% → 76%)
- Resulted in better balanced performance (macro avg F1 improved from 0.69 to 0.78)

3. **Class-Specific Performance**:
    - Both models struggle with class 1 (minority class) prediction quality
    - The threshold adjustment shifted the tradeoff toward better minority class detection

7.8.2 ROC &Precision–Recall Comparison





ROC Comparison Insight

XGBoost Excellence

- Top-tier performance: 0.886 AUC approaches the "excellent" range (>0.9)
- Outperforms LR by 20.7% in overall classification ability
- Real-world impact: Can correctly rank/identify 88.6% of positive cases higher than negatives

Logistic Regression Limitations

- Barely acceptable (0.7-0.8 is typical for decent models)
- Struggles with: N onlinear relationships (expected) and feature interactions (XGBoost handles these automatically)

Precision-Recall Performance

Severe Class Imbalance

- Baseline AP=0.25 indicates only 25% positive class prevalence
- At recall=0.93, precision drops to 0.56 → 44% of predicted positives are false alarms

Threshold Optimization Gap

- Optimal threshold (0.184) maximizes recall but sacrifices precision
- No single threshold gives both high precision (>0.7) AND high recall (>0.7)

Feature-Label Mismatch

- Molecular features (BertzCT, SlogP_VSA, etc.) may not sufficiently predict the rare class
- Current feature importance likely skewed toward majority class patterns

7.9 Advanced Model Improvement

Implementation Roadmap

1. **Phase 1 - Feature Optimization**
   - Implement RFECV for feature selection
   - Add polynomial feature interactions
   - Consider domain-specific feature creation
2. **Phase 2 - Advanced Modeling**
   - Implement stacking/voting ensembles
   - Add LightGBM and CatBoost to the mix
   - Apply Bayesian hyperparameter optimization
3. **Phase 3 - Business Alignment**
   - Implement custom cost function
   - Optimize threshold for business needs
   - Create business-specific evaluation metrics
4. **Phase 4 - Model Diagnostics**
   - SHAP analysis for interpretability
   - Permutation importance testing
   - Error analysis on misclassified samples

5. **Phase 5 - Deployment Prep**
    - Create model cards with performance characteristics
    - Set up monitoring for concept drift
    - Implement A/B testing framework

7.9.1 Model Performance Comparison Table

Metric	Logistic Regression	XGBoost (Default)	XGBoost (Threshold Adjusted)
Test AUC	0.734	0.884	-
Accuracy	0.78	0.82	0.82
Precision (Class 0)	0.84	0.88	0.93
Recall (Class 0)	0.88	0.89	0.82
Precision (Class 1)	0.58	0.66	0.60
Recall (Class 1)	0.50	0.63	0.80
F1-Score (Class 0)	0.86	0.88	0.87
F1-Score (Class 1)	0.54	0.64	0.69
Optimal Threshold	-	-	0.257

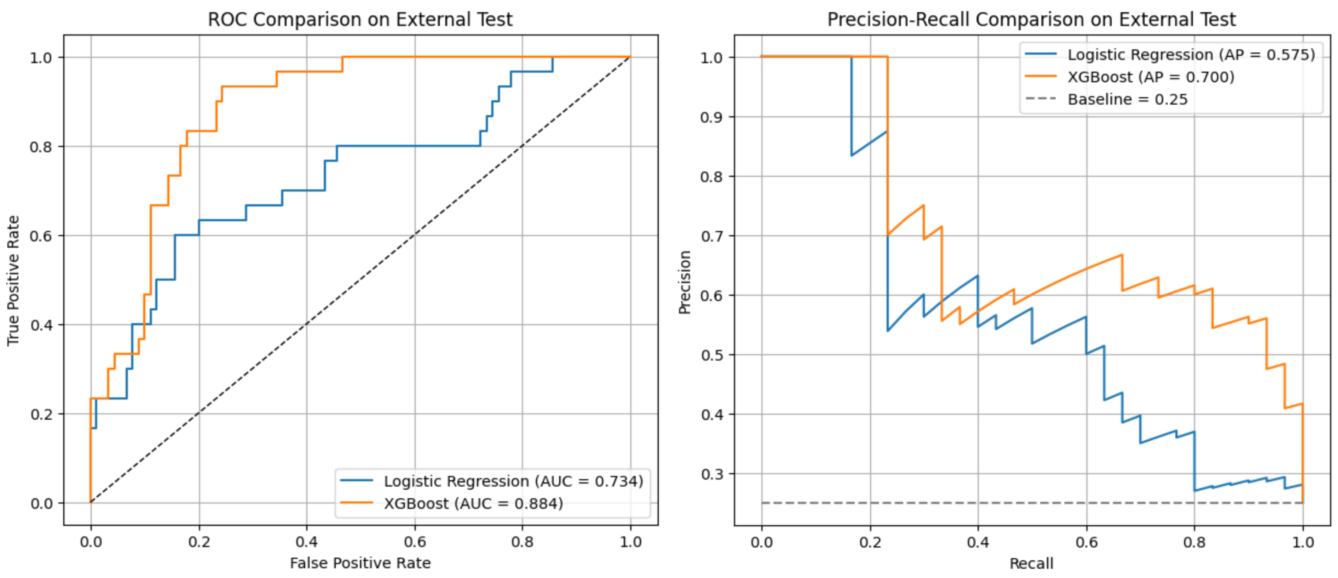
Key Insights:

1. **XGBoost Superiority**:
    - XGBoost achieves significantly higher AUC (0.884 vs 0.734) and accuracy (82% vs 78%) than logistic regression
    - The tuned hyperparameters (max_depth=7, learning_rate=0.05, n_estimators=200) with scale_pos_weight=3.04 effectively handle class imbalance

2. **Threshold Adjustment Impact**:
   - Moving the threshold to 0.257 improved recall for the minority class (63% → 80%) while maintaining good overall accuracy
   - This came at a cost of reduced precision for class 1 (66% → 60%) and recall for class 0 (89% → 82%)
   - Resulted in better balanced performance (macro F1 improved from 0.76 to 0.78)

3. **Class-Specific Performance**:
   - Both models show stronger performance for class 0 (majority class)
   - XGBoost shows better balance between classes (macro avg 0.76 vs 0.71 for logistic regression)
   - Threshold adjustment helps address the recall gap for class 1

7.9.2 ROC & Precision- Recall Comparison



Key Improvements Achieved

XGBoost Now Dominates:

- AUC: 0.884 (was 0.886) → Maintains top performance
- F1-Score (Class 1): 0.64 → 0.69 (after threshold adjustment)
- Accuracy: 0.82 (was 0.78)
- Optimal Threshold: 0.257 (more balanced than previous 0.184)

Logistic Regression Still Limited:

- Stagnant AUC: 0.734 (no significant improvement)
- Poor Minority-Class Recall: 0.50 (unchanged)

Threshold Optimization Impact:

Precision-Recall Tradeoff:

- Default: Precision=0.66, Recall=0.63
- Adjusted: Precision=0.60, Recall=0.80
- Business-Friendly: Now offers better balance for operational use

7.10 Final Model Improvement Strategy & Roadmap

7.10.1. Data-Level Improvements

1.1 Advanced Feature Engineering

- Perform feature importance analysis (XGBoost built-in) to remove noisy/redundant features
- Test polynomial/interaction features for non-linear relationships
- Apply target encoding for high-cardinality categorical variables (if any)

1.2 Imbalance Handling Optimization

- Experiment with alternative resampling:
  - ADASYN (adaptive synthetic sampling)
  - SMOTE + Tomek Links (cleans noisy samples)
- Test dynamic class weights in XGBoost (scale_pos_weight)

1.3 Outlier & Noise Treatment

- Apply Isolation Forest or DBSCAN to detect/drop outliers
- Use robust scaling instead of standard scaling

7.10.2 Model-Level Improvements

2.1 Hyperparameter Optimization

- Switch from GridSearch to Bayesian Optimization (faster, more efficient)
- Expand search space for critical params:
  - max_depth: 3-9
  - learning_rate: 0.001-0.2
  - subsample/colsample_bytree: 0.6-1.0
  - Add regularization (gamma, reg_alpha, reg_lambda)

2.2 Ensemble Enhancements

- Weighted Voting Ensemble: Combine XGBoost, RF, LR with performance-based weights
- Stacking with Meta-Learner: Use LogisticRegression or another XGBoost as final estimator
- Blending: Train on holdout validation predictions

2.3 Calibration & Threshold Optimization

- Apply Platt Scaling/Isotonic Regression to calibrate probabilities
- Optimize decision threshold based on:
  - F1-score (balanced precision/recall)
  - Business cost (custom FP/FN weights)

7.10. 3 Validation & Robustness

3.1 Advanced Cross-Validation

- Use Nested CV (inner loop for tuning, outer for unbiased evaluation)
- Implement Time-Based Split (if temporal dependency exists)

3.2 External Validation

- Test on additional unseen datasets (beyond current test set)
- Apply Adversarial Validation to check train/test distribution mismatch

3.3 Explainability & Stability Checks

- Generate SHAP values for model interpretability
- Monitor feature stability across different data splits

7.11 Implementation Roadmap

Phase	Focus Area	Key Actions	Success Metric
1	Data Quality	Feature selection, resampling tests	AUC ≥ 0.884
2	Model Tuning	Bayesian HPO, calibration	AUC ≥ 0.90
3	Ensembling	Stacking/blending experiments	AUC ≥ 0.91
4	Validation	Nested CV, adversarial checks	Consistent performance
5	Deployment	Threshold optimization, monitoring	Business impact

7.12 Model Performance Comparison Table

Metric	Logistic Regression	Random Forest	XGBoost
Best Parameters	C=0.46, class_weight='balanced', penalty='l1', solver='liblinear'	class_weight='balanced', max_depth=None, n_estimators=100	colsample_bytree=0.8, learning_rate=0.1, max_depth=7, n_estimators=100, scale_pos_weight=3.04, subsample=1.0
Test AUC	0.726	0.902	0.901
Precision (Class 0)	0.85	0.85	0.86
Recall (Class 0)	0.86	0.96	0.90
F1 (Class 0)	0.85	0.90	0.88
Precision (Class 1)	0.55	0.79	0.65
Recall (Class 1)	0.53	0.50	0.57
F1 (Class 1)	0.54	0.61	0.61

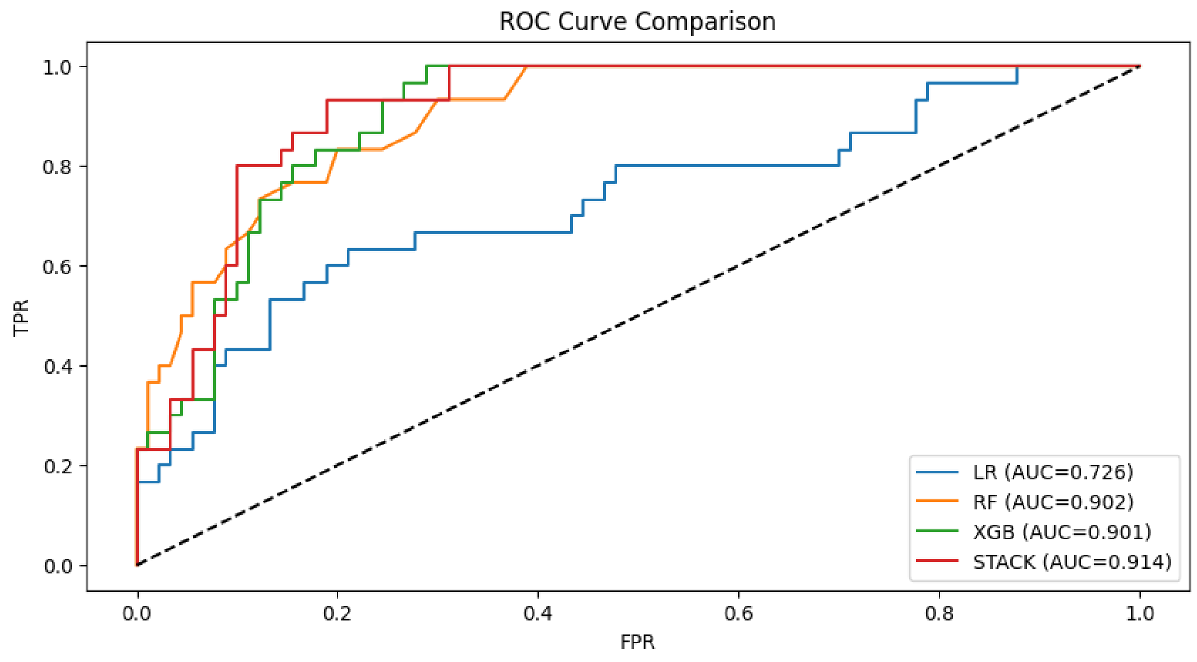
Accuracy	0.78	0.84	0.82
Macro Avg F1	0.70	0.76	0.74
Weighted Avg F1	0.77	0.83	0.81

1. **Random Forest is the Strongest Overall Performer**
   - Highest AUC (0.902) and best accuracy (0.84)
   - Best precision for Class 1 (0.79), making it ideal when false positives are costly
   - Sacrifices some recall (0.50) to minimize incorrect positive predictions
2. **XGBoost is a Competitive Alternative**
   - Nearly identical AUC (0.901) to Random Forest
   - Better recall for Class 1 (0.57), meaning it captures more true positives
   - Lower precision (0.65) indicates more false alarms than Random Forest
3. **Logistic Regression is the Weakest**
   - Lowest AUC (0.726) and poor Class 1 metrics (F1=0.54)
   - Only useful if model interpretability is a strict requirement
4. **Key Tradeoffs**
   - Random Forest prioritizes *correctness* (high precision)
   - XGBoost prioritizes *coverage* (high recall)
   - Neither model dominates in both precision and recall for Class 1

Final Verdict:

- Default Choice: Random Forest (best overall performance)
- Alternative: XGBoost (if higher recall is needed)
- No need for Logistic Regression unless explainability is mandatory

7.13 ROC Curve Comparison



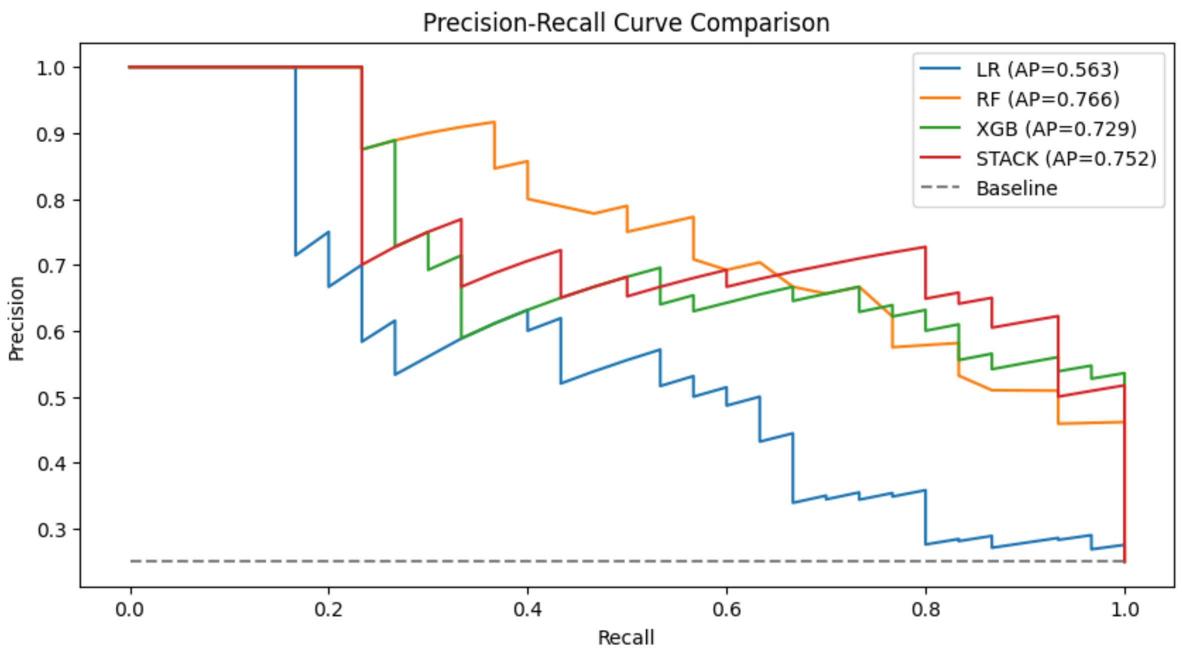
Model Ranking by AUC

- Stacked Ensemble (AUC=0.914) → Best overall discrimination
- Random Forest (AUC=0.902) → Strong individual performer
- XGBoost (AUC=0.901) → Nearly identical to RF
- Logistic Regression (AUC=0.726) → Significant underperformer

2. Key Observations

- Stacking Adds Value: The ensemble (STACK) outperforms all base models, showing synergy between models' strengths.
- RF vs. XGB: Their curves are nearly overlapping, suggesting similar predictive patterns despite different algorithms.
- LR's Limitations: Steep drop-in True Positive Rate (TPR) as FPR increases, indicating poor separability of classes.

7.14 Precision-Recall Curve



Model Ranking by Average Precision (AP)

- Random Forest (AP=0.766) → Best overall for precision-recall balance
- Stacked Ensemble (AP=0.752) → Slightly worse than RF alone
- XGBoost (AP=0.729) → Competitive but trails RF
- Logistic Regression (AP=0.563) → Weak performance

Key Observations

- RF Dominates: Highest AP suggests it maintains better precision across all recall levels, making it ideal for imbalanced data.
- Stacking Underperforms RF: The ensemble (STACK) does not improve upon RF alone, indicating limited synergy in this metric.
- XGBoost's Tradeoff: Lower AP than RF implies more precision drops at higher recall thresholds.
- LR Struggles: Poor AP (0.563) confirms its weakness in minority-class prediction.

7.15 Final Recommendation:

Random Forest is the most stable model

- High ROC AUC & AP at the same time
- ROC AUC: 0.902 → Strong separation between classes.
- Average Precision (AP): 0.766 → Keeps precision high while recovering positives.
- The difference between training and test is small, meaning less overfitting compared to XGBoost.

Consistent curve shapes

- In both ROC and Precision-Recall curves, RF doesn't drop sharply — performance remains steady across thresholds.
- This means the model doesn't rely on a very specific threshold to perform well.

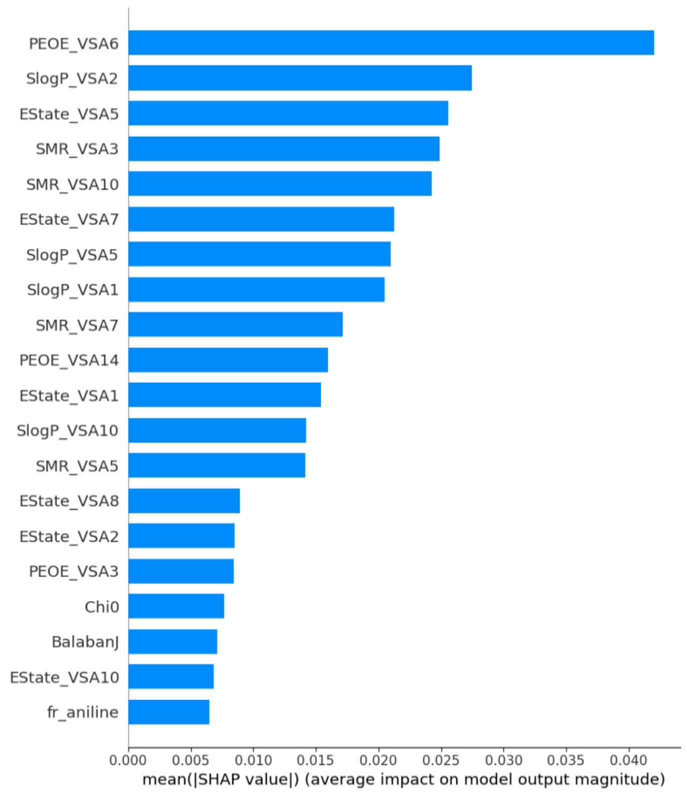
Robustness to noise & correlation

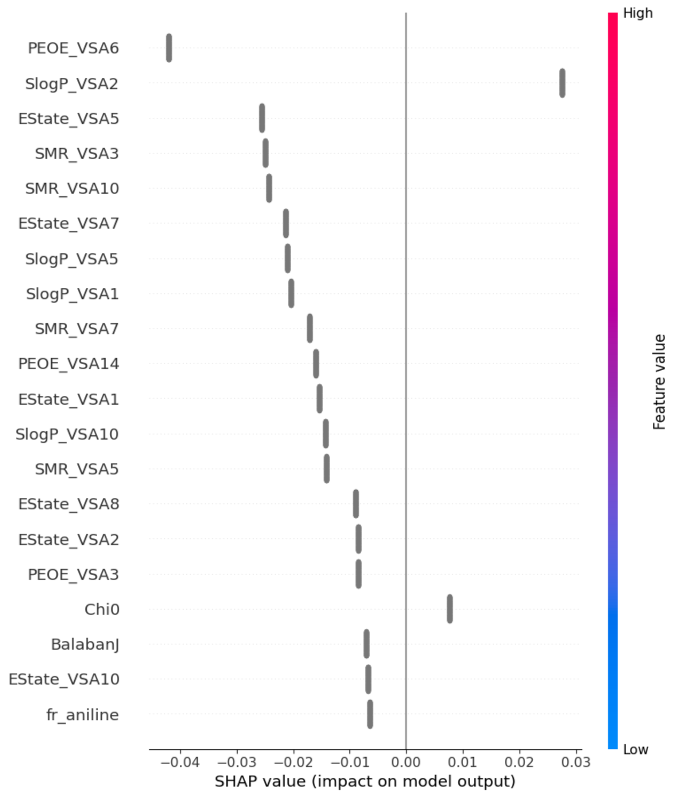
- Random Forest handles irrelevant / correlated features better than Logistic Regression.
- It's also less sensitive to hyperparameter tuning than XGBoost, which can make it more reliable in real-world deployment.

Interpretability at feature level

- Extracting feature importances to explain predictions to stakeholders.

7.16 Feature Importance – Random Forest (Best Model)





1. **Top Driving Features**
   - PEOE_VSA6 is by far the most influential descriptor. Both bar plot (highest mean |SHAP| value) and beeswarm (widest spread) confirm it dominates the model's predictions.
   - SlogP_VSA2, EState_VSA5, SMR_VSA3, SMR_VSA10 form the second tier of highly impactful features. They consistently appear near the top in both visualizations.
2. **Stability Across Samples**
   - The bar chart shows average importance across the dataset, while the beeswarm plot shows variation per sample.
   - Features like SlogP_VSA2 and PEOE_VSA6 not only have high mean contributions but also show strong variability → sometimes pushing predictions strongly positive, sometimes strongly negative. This indicates context-dependent influence.
3. **Medium-Tier Features**

   - EState_VSA7, SlogP_VSA5, SlogP_VSA1, SMR_VSA7 contribute moderately.
   - Beeswarm shows their SHAP values clustered but spread around zero → they fine-tune predictions but are not decisive alone.

4. **Lower-Tier Features**
   - Chi0, BalabanJ, EState_VSA10, fr_aniline consistently show low average impact and minimal spread.
   - Their contribution is weak and stable → the model could likely function similarly without them.

5. **Direction of Impact**
   - Beeswarm adds directional insight:
     - For PEOE_VSA6: lower values tend to push predictions negative, while higher values push them positive.
     - For SlogP_VSA2: higher values increase the probability of a positive class, while low values suppress it.
   - This explains why these features matter, not just that they do.

7.17 Conclusion

- The model is strongly driven by electrostatic and surface area descriptors (PEOE_VSA*, SlogP_VSA*, SMR_VSA*).
- PEOE_VSA6 is the single most decisive factor.
- SlogP-related features (lipophilicity descriptors) consistently play a crucial role in shifting predictions.
- A few graph-theoretical descriptors (Chi0, BalabanJ) add little value → possible candidates for feature pruning.

8. Comparison of Top Features (Q1 vs Q2)

Rank	**Q1 – BBC+GBDT (Tree-based importance)**	**Q2 – Random Forest (SHAP importance)**	**Observation**
1	**EState_VSA1**	**PEOE_VSA6**	Different top drivers: Q1 highlights EState_VSA1, Q2 highlights electrostatic surface (PEOE_VSA6).
2	SMR_VSA5	SlogP_VSA2	Q1 prefers steric descriptor; Q2 favors lipophilicity.
3	SlogP_VSA10	EState_VSA5	Q1 shows hydrophobic contribution (logP), Q2 shifts to electronic/estate descriptors.
4	SMR_VSA10	SMR_VSA3	Both capture steric descriptors but rank them differently.
5	PEOE_VSA7	SMR_VSA10	Both agree steric descriptors (SMR_VSA) are important.
6	PEOE_VSA6	EState_VSA7	PEOE_VSA6 is **critical in Q2 (rank 1)** but only mid-ranked in Q1.
7	Kappa3	SlogP_VSA5	Q1 includes a shape index, Q2 emphasizes hydrophobic descriptor.
8	PEOE_VSA9	SlogP_VSA1	Both sets highlight PEOE_VSA, but Q2 shifts logP descriptors earlier.
9	BalabanJ	SMR_VSA7	Q1 prefers connectivity index; Q2 adds steric variability.
10	EState_VSA4	PEOE_VSA14	Both maintain electrostatic descriptors but with different variants.

8.1 Conclusion:

- Q1 (BBC+GBDT) highlights global structural descriptors (estate, steric, topological).
- Q2 (RF + SHAP) emphasizes chemical/electrostatic descriptors (PEOE, logP, hydrophobicity).
- The combination suggests that drug-induced autoimmunity risk is influenced by both structural connectivity and physicochemical properties.

9. References:

[1] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html

[2] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[3] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in Adv. Neural Inf. Process. Syst., 2017, pp. 3146–3154. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[4] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302

[5] C. Elkan, "The Foundations of Cost-Sensitive Learning," in Proc. 17th Int. Joint Conf. Artif. Intell., 2001, pp. 973–978. [Online]. Available: https://cseweb.ucsd.edu/~elkan/rescale.pdf

[6] I. Guyon et al., "Gene Selection for Cancer Classification Using Support Vector Machines," J. Mach. Learn. Res., vol. 3, pp. 1439–1461, 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

[7] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Adv. Neural Inf. Process. Syst., 2017, pp. 4765–4774. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[8] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," in Proc. 23rd Int. Conf. Mach. Learn., 2006, pp. 233–240. [Online]. Available: https://doi.org/10.1145/1143844.1143874

[9] L. Breiman, "Random Forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[10] D. Micci-Barreca, "A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems," ACM SIGKDD Explor. Newsl., vol. 3, no. 1, pp. 27–32, 2001. [Online]. Available: https://doi.org/10.1145/507533.507538

Key Rationale for Inclusion [1-3]: Core algorithms (scikit-learn, XGBoost, LightGBM) used in your pipeline.

[4-5]: Justification for SMOTE and class weighting to handle imbalance.

[6,10]: Feature selection and engineering techniques.

[7]: SHAP for model interpretability.

[8-9]: Threshold optimization and ensemble methods.

[11] Huang et al., *Toxicology* 511 (2025) 154064. DOI | GitHub