Research Documentation: Quantitative Structure-Activity Relationship (QSAR) Modeling for Coronavirus Main Protease Inhibitors

Authors: Aditya Mukhopadhyay **Date:** July 02, 2024 **Project Version:** 1.0

Executive Summary

This research project outlines the development and optimization of Quantitative Structure-Activity Relationship (QSAR) models aimed at predicting the inhibitory activity (pIC50) of chemical compounds against the SARS coronavirus main protease (CHEMBL3927). The methodology encompasses a complete cheminformatics pipeline: data acquisition from the ChEMBL database, robust data preprocessing, diverse molecular feature engineering (Lipinski's rules, comprehensive RDKit descriptors, and Morgan fingerprints), exploratory data analysis, and advanced machine learning modeling.

A RandomForestRegressor was employed as the predictive algorithm, with extensive hyperparameter tuning carried out using GridSearchCV and RandomizedSearchCV. The performance of these optimized models was rigorously evaluated and compared against a baseline model. The findings indicate that both tuning strategies significantly improved model performance, with the **RandomizedSearchCV-tuned model achieving the best predictive capability (R^2 = 0.546, RMSE = 0.524 on the test set)**. This highlights the effectiveness of broader parameter space exploration. Further analysis into feature importances elucidated the most influential molecular properties, offering valuable insights for future medicinal chemistry efforts, despite the inherent challenge in interpreting individual fingerprint bits. The study concludes with a comparative assessment of fingerprint-based vs. descriptor-based models, emphasizing the trade-off between predictive power and chemical interpretability.

1. Introduction

1.1 Background and Significance of QSAR in Drug Discovery

Drug discovery is a notoriously lengthy, expensive, and high-risk endeavor. The process of identifying new therapeutic agents typically involves several stages, from target identification and lead discovery to preclinical and clinical development. A significant bottleneck in the early stages is the inefficient screening of vast chemical libraries for compounds with desired biological activities.

Quantitative Structure-Activity Relationship (QSAR) modeling emerges as a powerful computational solution to address this challenge. QSAR is a correlative approach that establishes a mathematical relationship between a compound's molecular structure (represented by numerical descriptors or fingerprints) and its biological activity. By building such models, researchers can:

- **Virtually Screen Libraries:** Predict the activity of millions of compounds without the need for physical synthesis and experimental testing.
- **Prioritize Synthesis:** Focus resources on synthesizing compounds with the highest predicted activity, thereby reducing time and cost.
- **Guide Lead Optimization:** Provide insights into which molecular features enhance or diminish activity, facilitating iterative design cycles.
- **Elucidate Mechanism of Action:** Offer clues about the structural requirements for binding to a target, informing the understanding of drug-target interactions.

In the context of emerging infectious diseases, such as those caused by coronaviruses, the rapid identification of potential antiviral agents is paramount. QSAR accelerates this process by enabling fast, in-silico hypothesis generation, making it an indispensable tool for pandemic preparedness and response.

1.2 Problem Statement

The COVID-19 pandemic underscored the urgent need for effective antiviral therapies. One promising target for coronavirus inhibitors is the main protease (Mpro), an enzyme critical for viral replication. While experimental screening generates valuable bioactivity data, this process is laborious. The objective is to develop a robust QSAR model that can accurately predict the inhibitory activity of compounds against SARS-CoV Mpro based solely on their chemical structures. This model would serve as a computational filter to identify promising candidates, thereby streamlining the drug discovery pipeline. A key challenge is optimizing the model's performance to ensure high predictive accuracy and reliability.

1.3 Research Objectives

This research project pursued the following specific objectives:

1. **Data Acquisition and Preprocessing:** To systematically retrieve bioactivity data for coronavirus-related targets from the ChEMBL database, focusing on SARS-CoV Mpro, and prepare it for QSAR modeling by handling missing values, classifying bioactivity, and standardizing activity units.
2. **Molecular Feature Engineering:** To transform the chemical structures of compounds into numerical representations by calculating diverse molecular descriptors (Lipinski's rules, comprehensive RDKit descriptors) and generating molecular fingerprints (Morgan fingerprints).
3. **Exploratory Data Analysis (EDA):** To perform a thorough analysis of the prepared dataset to understand the distribution of bioactivity, identify potential outliers, and assess the relationships between molecular properties and biological activity.
4. **Baseline QSAR Model Development:** To establish a foundational predictive model using RandomForestRegressor and evaluate its initial performance metrics (R-squared, RMSE) on a held-out test set.
5. **Hyperparameter Optimization:** To systematically improve the performance of the RandomForestRegressor by employing advanced hyperparameter tuning techniques: GridSearchCV (exhaustive search) and RandomizedSearchCV (stochastic search).
6. **Comparative Model Evaluation:** To comprehensively compare the predictive performance (R-squared, RMSE) of the baseline model against the models optimized via GridSearchCV and RandomizedSearchCV, utilizing various cross-validation strategies for robust generalization error estimation.
7. **Feature Importance Analysis:** To interpret the contributions of different molecular features to the predictive models, particularly comparing the insights derived from fingerprint bits versus chemically intuitive RDKit descriptors.

2. Methodology

The research adopted a structured, multi-stage computational methodology, executed within a Google Colab environment to leverage cloud-based computational resources.

2.1 Computational Environment

- **Platform:** Google Colaboratory (Colab)
- **Hardware:** Default Colab CPU/GPU runtimes (specific GPU type and RAM allocated dynamically)
- **Programming Language:** Python 3.7+
- **Key Libraries:**

- `pandas` (for data manipulation)
- `chembl_webresource_client` (for ChEMBL API interaction)
- `rdkit` (for cheminformatics functionalities: SMILES parsing, descriptor calculation, fingerprint generation)
- `numpy` (for numerical operations, especially pIC50 conversion)
- `seaborn` & `matplotlib` (for data visualization)
- `scikit-learn` (for machine learning models, hyperparameter tuning, and evaluation metrics)

2.2 Data Acquisition

1. **ChEMBL API Connection:**

```
!pip install chembl_webresource_client
import pandas as pd
from chembl_webresource_client.new_client import new_client
```

2. **Target Identification:** The ChEMBL API was queried to identify protein targets related to "coronavirus."

```
target = new_client.target
target_query = target.search('coronavirus')
targets = pd.DataFrame.from_dict(target_query)
```

From the `targets` DataFrame, the entry at index 6, **CHEMBL3927**, named "SARS coronavirus 3C-like proteinase" (also known as Main Protease or Mpro), was specifically selected as the target for this study due to its critical role in coronavirus replication and its established potential as a drug target.

```
selected_target1 = targets.target_chembl_id[6] # 'CHEMBL3927'
```

3. **Bioactivity Data Retrieval:** Activity data associated with `CHEMBL3927` was filtered to include only `IC50` (`standard_type="IC50"`) measurements, ensuring a consistent and interpretable activity metric.

```
activity = new_client.activity
res1 = activity.filter(target_chembl_id=selected_target1).filter(standard_type="IC50")
df = pd.DataFrame.from_dict(res1)
```

4. **Initial Data Storage:** The raw retrieved data was saved to a CSV file for persistent storage and ease of access.

```
df.to_csv('bioactivity_data.csv', index=False)
```

2.3 Data Preprocessing

1. **Handling Missing Activity Values:** Rows where the `standard_value` (IC50) was not available were removed, as these entries lack the crucial information for QSAR modeling.

```
df2 = df[df.standard_value.notna()]
```

2. **Bioactivity Classification:** Compounds were classified into three categories: 'active', 'intermediate', and 'inactive', based on commonly accepted thresholds for IC50 values (in nM):
   - Active: `IC50 <= 1000 nM`
   - Intermediate: `1000 nM < IC50 < 10000 nM`
   - Inactive: `IC50 >= 10000 nM` This classification is useful for exploratory data analysis and could be used for classification models in a broader study.
3. **pIC50 Transformation:** For regression modeling, raw IC50 values (which are often log-normally distributed and range widely) were converted to `pIC50`. This transformation makes the distribution more normal and linearizes the concentration scale, which is beneficial for many regression algorithms.
   - **Capping of High IC50 Values:** To prevent `-np.inf` values when taking the logarithm of very large IC50s (e.g., millions of nM), values exceeding `100,000,000 nM` (100 mM, corresponding to a pIC50 of 4) were capped at this maximum. This ensures numerical stability and prevents outliers from disproportionately influencing the model.

```
def calculate_pIC50(df_input):
    df = df_input.copy()
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df.dropna(subset=['standard_value'], inplace=True)
    norm_values = []
    for i in df['standard_value']:
        if i > 100000000: # Capping at 100mM
            i = 100000000
        norm_values.append(i)
    df['standard_value_norm'] = norm_values
    df['pIC50'] = -np.log10(df['standard_value_norm'] * (10**-9)) # Convert nM to M (10^-9)
    df.drop(columns=['standard_value_norm'], inplace=True)
    return df
df_pIC50 = calculate_pIC50(df_combined) # df_combined includes Lipinski descriptors already
df_final = df_pIC50 # This DataFrame now contains the pIC50 column
```

4. **Final Preprocessed Data Storage:** The DataFrame containing molecule IDs, canonical SMILES, bioactivity class, standard value, and pIC50 was saved.

```
df3.to_csv('bioactivity_preprocessed_data.csv', index=False)
# Also copied to Google Drive for persistent storage.
```

For specific analyses (e.g., initial Lipinski rule analysis and Mann-Whitney U tests), intermediate compounds were excluded to focus on a clearer active/inactive distinction, resulting in `df_2class`.

2.4 Feature Engineering

Molecular structures (canonical SMILES) were transformed into numerical features using RDKit, a cheminformatics toolkit.

1. **Lipinski's Rule of Five Descriptors:** These are fundamental descriptors related to drug-likeness and oral bioavailability.
   - Molecular Weight (MW)
   - LogP (octanol-water partition coefficient, a measure of lipophilicity)
   - NumHDonors (number of hydrogen bond donors)
   - NumHAcceptors (number of hydrogen bond acceptors)

```
from rdkit import Chem
from rdkit.Chem import Descriptors, Lipinski

def lipinski(smiles, verbose=False):
    # ... (function implementation as in code) ...
    return descriptors

df_lipinski = lipinski(df.canonical_smiles)
df_combined = pd.concat([df, df_lipinski], axis=1) # Combine with original data for pIC50 calculation
```

2. **Comprehensive RDKit Descriptors:** A wider array of 200+ physicochemical and structural descriptors from RDKit were calculated to capture more nuanced molecular properties.

```
def calculate_all_rdkit_descriptors(smiles_list):
    # ... (function implementation as in code) ...
    return pd.DataFrame(all_descriptors, columns=descriptor_names)

df_descriptors = calculate_all_rdkit_descriptors(df_preprocessed['canonical_smiles'])
# Clean up: Drop columns with NaN values or no variance (constant values across all molecules).
df_descriptors = df_descriptors.dropna(axis=1, how='any')
df_descriptors = df_descriptors.loc[:, (df_descriptors != df_descriptors.iloc[0]).any()]
```

3. **Morgan Fingerprints:** These are a type of circular fingerprint, similar to Extended Connectivity Fingerprints (ECFPs), encoding structural features as a bit vector. They are highly effective for QSAR tasks due to their ability to represent diverse substructures.
   - `radius=2`: Captures features up to 2 bonds away from each atom (equivalent to ECFP4).
   - `n_bits=2048`: Specifies the length of the bit vector, a common choice balancing dimensionality and information density.

```
from rdkit.Chem import AllChem

def generate_morgan_fingerprints(smiles_list, radius=2, n_bits=2048):
    # ... (function implementation as in code) ...
    return df_fp

df_fingerprints = generate_morgan_fingerprints(df_preprocessed['canonical_smiles'])
```

Note on Data Alignment: It was crucial to ensure that after various preprocessing steps (like dropping rows with missing IC50 or invalid SMILES), the feature DataFrames (fingerprints, descriptors) were perfectly aligned with the `pIC50` target vector. This was managed by consistently resetting indices and using `dropna()` after concatenation where necessary.

2.5 Exploratory Data Analysis (EDA)

1. **IC50 vs. pIC50 Scatter Plot:** Visualized the relationship between raw IC50 and transformed pIC50 values, color-coded by `bioactivity_class`. A logarithmic scale was applied to the x-axis (IC50) to better illustrate the transformation.

```
# ... (scatterplot code) ...
plt.xscale('log')
```

2. **Bioactivity Class Frequency Count Plot:** Illustrated the distribution of 'active' vs. 'inactive' compounds (after removing 'intermediate' for this specific plot) to assess class balance.

```
sns.countplot(x='bioactivity_class', data=df_2class, edgecolor='black')
```

3. **MW vs. LogP Scatter Plot:** Examined the distribution of compounds based on two key Lipinski's descriptors, showing how active/inactive compounds populate this chemical space. The size of points was mapped to pIC50 to add another dimension of information.

```
sns.scatterplot(x='MW', y='LogP', data=df_2class, hue='bioactivity_class', size='pIC50', edgecolor='black', alpha=0.7)
```

4. **Box Plots for Descriptors vs. Bioactivity:** Box plots were generated for `pIC50`, `MW`, `LogP`, `NumHDonors`, and `NumHAcceptors` against `bioactivity_class` to visually inspect differences in distributions between active and inactive groups.

```
sns.boxplot(x = 'bioactivity_class', y = 'pIC50', data = df_2class)
```

5. **Mann-Whitney U Tests:** Non-parametric statistical tests (Mann-Whitney U) were performed to quantify the statistical significance of differences observed in descriptor distributions between the 'active' and 'inactive' compound groups.
    - **Null Hypothesis (H0):** The distributions of the descriptor values for 'active' and 'inactive' compounds are the same.
    - **Alternative Hypothesis (H1):** The distributions are different.
    - **Alpha (α) level:** 0.05.

```
from scipy.stats import mannwhitneyu
def mannwhitney(descriptor, verbose=False):
    # ... (function implementation as in code) ...
    return results
```

2.6 QSAR Model Building and Hyperparameter Optimization

1. **Feature (X) and Target (y) Selection:**
    - The **Morgan Fingerprints (`df_fingerprints`)** were chosen as the primary feature set (`X`) for their demonstrated predictive power in cheminformatics.
    - The **`pIC50`** column from `df_final` was used as the target variable (`y`).
    - For a more **interpretable model**, a separate analysis later used the **RDKit Descriptors (`df_descriptors_aligned`)** as `X` .
2. **Data Alignment:** Crucially, `X` and `y` were concatenated and any rows with `NaN` values (which could arise from issues in SMILES parsing or fingerprint generation) were dropped to ensure that `X` and `y` had matching rows, preventing dimension mismatch errors.
3. **Train-Test Split:** The dataset was partitioned into an 80% training set and a 20% test set. `random_state=42` was used to ensure reproducibility of the split.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. **Baseline RandomForestRegressor:** A RandomForestRegressor with default hyperparameters (`n_estimators=100` , `random_state=42`) was trained as a benchmark. `n_jobs=-1` was used to utilize all available CPU cores.

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)
model.fit(X_train, y_train)
```

5. **Hyperparameter Tuning with GridSearchCV:**
    - **Purpose:** To systematically search a predefined parameter grid for the optimal combination of hyperparameters.
    - **`param_grid` Definition:**

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None], # None means unlimited depth
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

    - **Grid Search Execution:** `GridSearchCV` performs cross-validation for each combination in the grid.

```
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_grid_model = grid_search.best_estimator_
```

6. **Hyperparameter Tuning with RandomizedSearchCV:**
    - **Purpose:** To efficiently explore a wider hyperparameter space by randomly sampling a fixed number of combinations. This is particularly useful when the grid for GridSearchCV becomes prohibitively large.
    - **`random_param_grid` Definition:**

```
from scipy.stats import randint
random_param_grid = {
    'n_estimators': randint(100, 800), # Sample integers between 100 and 800
    'max_depth': [5, 10, 15, 20, 25, 30, None],
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10)
}
```

    - **Randomized Search Execution:** `n_iter=10` was chosen to sample 10 random combinations, balancing exploration with computational cost.

```
from sklearn.model_selection import RandomizedSearchCV
rf = RandomForestRegressor(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distributions=random_param_grid, n_iter=10, cv=5, random_state=42
random_search.fit(X_train, y_train)
best_random_model = random_search.best_estimator_
```

2.7 Model Evaluation and Interpretation

1. **Performance Metrics:** R-squared (R^2) and Root Mean Squared Error (RMSE) were the primary metrics used.
    - R^2: Represents the proportion of the variance in the dependent variable that is predictable from the independent variables. A higher R^2 (closer to 1) indicates a better fit.
    - RMSE: Measures the average magnitude of the errors. A lower RMSE (closer to 0) indicates a better fit.

```
from sklearn.metrics import r2_score, mean_squared_error
# For baseline model: r2, rmse = r2_score(y_test, y_pred), np.sqrt(mean_squared_error(y_test, y_pred))
# For GridSearchCV model: r2_grid, rmse_grid = r2_score(y_test, y_pred_grid), np.sqrt(mean_squared_error(y_test, y_pred_grid))
# For RandomizedSearchCV model: r2_random, rmse_random = r2_score(y_test, y_pred_random), np.sqrt(mean_squared_error(y_test, y
```

2. **Cross-Validation for Robustness:**
    - **K-Fold Cross-Validation (k=5):** Applied to the entire dataset (X, y) to provide a more reliable estimate of generalization performance, mitigating the bias from a single train-test split. The mean R^2 and standard deviation across folds were reported.

```
from sklearn.model_selection import KFold, cross_val_score
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
scores_original_kfold = cross_val_score(model, X, y, cv=kfold, scoring='r2', n_jobs=-1)
# Similar for best_grid_model and best_random_model
```

    - **Leave-One-Out Cross-Validation (LOOCV):** For datasets up to 500 samples, LOOCV (where k=N, number of samples) provides an almost unbiased estimate of generalization error. However, it can be computationally intensive.

```
from sklearn.model_selection import LeaveOneOut
loocv = LeaveOneOut()
# scores_original_loocv = cross_val_score(model, X, y, cv=loocv, scoring='r2', n_jobs=-1)
# (Similarly for tuned models)
```

    - **Leave-P-Out Cross-Validation (LPOCV):** For datasets larger than 100 samples, LPOCV (where p samples are left out) was explicitly skipped due to extreme computational demands, as its combinations grow factorially. **Note on LOOCV/LPOCV NaN Results:** In some runs, `cross_val_score` with `LeaveOneOut` or `LeavePOut` might produce `NaN` if individual folds result in R-squared values that are too low (e.g., negative, which R-squared can be) or if the folds are too small to yield meaningful statistics. For the dataset size of 245 samples, LOOCV is technically feasible but may encounter numerical instability for individual 1-sample test sets, leading to `nan` results if not all folds produce positive R2. K-Fold CV, with larger fold sizes, is generally more stable.

3. **Feature Importance Analysis:** The `feature_importances_` attribute of the trained RandomForestRegressor models was used to identify which features (fingerprint bits or RDKit descriptors) contributed most to predictions.

```
feature_importances = best_random_model.feature_importances_
feature_names = X.columns
importances_series = pd.Series(feature_importances, index=feature_names)
sorted_importances = importances_series.sort_values(ascending=False)
# Similar process for the RDKit descriptor-based model
```

   Visualization (bar plots) was used to highlight the top N most important features.

---

3. Results

3.1 Data Characteristics and Exploratory Data Analysis (EDA) Findings

- **Dataset Size:** The initial ChEMBL query for `CHEMBL3927` yielded a total of **247** raw entries. After removing entries with missing `standard_value` and invalid SMILES, the final dataset comprised **245 unique compounds** for QSAR modeling.
- **pIC50 Distribution:** The pIC50 values ranged from approximately **2.7 to 7.3**, with a mean of **4.89**. This distribution is typical for biological activity data and is more suitable for regression compared to the original, highly skewed IC50 values.
- **Bioactivity Class Balance (Active/Inactive):**
    - The filtered `df_2class` (excluding 'intermediate' compounds) contained **198 compounds**.
    - The distribution between 'active' and 'inactive' compounds was:
        - Active: [Number of active compounds, e.g., 90]
        - Inactive: [Number of inactive compounds, e.g., 108]
    - This indicates a reasonably balanced dataset for classification tasks, though the primary focus here is regression.
- **Lipinski's Descriptors Analysis:**
    - **pIC50:** The Mann-Whitney U test for `pIC50` between 'active' and 'inactive' groups yielded a p-value of **4.76e-20**, which is significantly less than 0.05. This strong statistical difference confirms that the chosen pIC50 threshold effectively distinguishes between active and inactive compounds.
    - **Molecular Weight (MW):** Mann-Whitney U test p-value: **0.009** (< 0.05). Interpretation: There is a statistically significant difference in MW distribution between active and inactive compounds.

- **LogP:** Mann-Whitney U test p-value: **0.063** (> 0.05). Interpretation: No statistically significant difference in LogP distribution was found between active and inactive compounds. This suggests that lipophilicity, in isolation, might not be a primary differentiating factor for activity in this specific dataset based on this threshold.
- **NumHDonors:** Mann-Whitney U test p-value: **0.001** (< 0.05). Interpretation: A statistically significant difference in the number of hydrogen bond donors exists between the groups.
- **NumHAcceptors:** Mann-Whitney U test p-value: **0.012** (< 0.05). Interpretation: A statistically significant difference in the number of hydrogen bond acceptors exists between the groups.

3.2 Model Performance on Test Set

The following table summarizes the R-squared (R^2) and Root Mean Squared Error (RMSE) of the different RandomForestRegressor models on the unseen 20% test set:

Model	R-squared (R^2)	RMSE
Original Fingerprint Model	0.517	0.541
GridSearchCV Fingerprint Model	0.533	0.532
RandomizedSearchCV Fingerprint Model	**0.546**	**0.524**
Descriptor-based Model	0.471	0.566

Interpretation: The RandomizedSearchCV-tuned model demonstrated the best performance on the test set, achieving the highest R-squared and lowest RMSE. This indicates that it explains the most variance in pIC50 and has the lowest average prediction error among the tested models.

3.3 Hyperparameter Tuning Outcomes

- **GridSearchCV Best Parameters:**

```
{'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}
```

  - Best cross-validation R-squared score: **0.6065**

- **RandomizedSearchCV Best Parameters:**

```
{'n_estimators': 800, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_depth': None}
```

  - Best cross-validation R-squared score: **0.6016**

Interpretation: Both methods found parameters that yielded good cross-validation scores. Notably, RandomizedSearchCV selected a much larger `n_estimators` value (800) and allowed `max_depth` to be `None` (unlimited), which aligns with its broader search capabilities.

3.4 Cross-Validation for Robustness (Full Dataset)

To assess the models' generalization performance more robustly, K-Fold Cross-Validation was performed on the *entire* dataset. LOOCV and LPOCV were considered but noted to be computationally expensive (LOOCV produced NaN in this dataset's context, likely due to numerical instability with very small test sets, while LPOCV was explicitly skipped for efficiency).

Model	K-Fold (R^2) Mean	K-Fold (R^2) Std	LOOCV (R^2)	LPOCV (R^2)
Original Fingerprint Model	0.588	0.096	nan	N/A
GridSearchCV Fingerprint Model	0.597	0.076	nan	N/A
RandomizedSearchCV Fingerprint Model	**0.602**	**0.080**	nan	N/A

Interpretation: The K-Fold CV results generally align with the test set evaluation, showing that tuning improves the average R-squared across different data splits. RandomizedSearchCV still maintains the highest average R-squared. The standard deviation indicates the variability of the R-squared across folds, providing a measure of model stability.

3.5 Feature Importance Analysis

Top 20 Most Important Features (Morgan Fingerprint Bits - from RandomizedSearchCV Model):

```
morgan_1729    0.103123
morgan_1086    0.094664
morgan_675     0.051307
morgan_1197    0.043211
morgan_41      0.039579
morgan_866     0.037984
morgan_1871    0.036055
morgan_1603    0.020981
morgan_1088    0.020539
morgan_862     0.020167
morgan_1720    0.019389
morgan_1535    0.019083
morgan_116     0.018281
morgan_1573    0.017744
morgan_142     0.015062
morgan_1602    0.012581
morgan_1964    0.012467
morgan_980     0.010728
morgan_1087    0.008389
morgan_110     0.008294
```

Interpretation: Specific fingerprint bits (`morgan_1729` , `morgan_1086` , etc.) show high importance. These bits represent the presence of particular substructures within the molecules. While identifying the exact substructure corresponding to each bit requires further mapping (e.g., using RDKit's `MolToMorganFingerprint` with `bitInfo` or `GetMorganFingerprintAsNum`) and is not directly outputted here, their high importance suggests that the presence or absence of these specific structural patterns is highly predictive of activity.

Top 20 Most Important Features (RDKit Descriptors - from Descriptor-based Model):

```
FpDensityMorgan3       0.088133
AvgIpc                 0.074304
SMR_VSA5               0.038775
fr_pyridine            0.035761
Chi4v                  0.034197
Chi3v                  0.032673
fr_sulfide             0.030792
FpDensityMorgan1       0.029787
SMR_VSA6               0.029527
VSA_EState6            0.019258
MaxAbsPartialCharge    0.018511
MinPartialCharge       0.017989
Chi2v                  0.016498
Chi0                   0.015871
EState_VSA10           0.015124
TPSA                   0.013683
EState_VSA8            0.013387
SMR_VSA7               0.012771
VSA_EState2            0.012595
MolLogP                0.012055
```

Interpretation: This list provides more chemically intuitive insights:

- **`FpDensityMorgan3` and `FpDensityMorgan1` :** These descriptors relate to the density of atom neighborhoods within the molecule. Their high importance suggests that the overall 'compactness' or 'branching' of the molecule, and how its atoms are distributed, is crucial for activity.
- **`AvgIpc` (Average Information Content):** Measures the average complexity or diversity of the atom-bond environments. High importance implies that compounds with specific patterns of structural complexity are more active.
- **`fr_pyridine` :** Indicates the number of pyridine ring fragments. Its high importance points to the potential role of pyridine groups in binding to the main protease.
- **`fr_sulfide` :** Similarly, suggests that compounds containing sulfide (-S-) linkages are favored.
- **`Chi` descriptors (e.g., Chi4v, Chi3v, Chi2v, Chi0):** These are topological indices related to molecular connectivity and branching patterns. Their significance highlights the importance of the molecule's overall shape and branching for activity.
- **`MolLogP` (Lipophilicity):** Despite not being a primary differentiator in the Mann-Whitney test, its presence here suggests that it still contributes to the overall predictive power when considered alongside other descriptors.
- **`TPSA` (Topological Polar Surface Area):** Relates to a molecule's drug transport properties and interactions.

---

4. Discussion and Conclusion

4.1 The Value of Hyperparameter Tuning

The results unequivocally demonstrate the critical role of hyperparameter tuning in QSAR model development. Both GridSearchCV and RandomizedSearchCV led to improved R-squared values and reduced RMSE on the test set when compared to the baseline RandomForestRegressor. This indicates that the default hyperparameters were not optimal for this specific dataset and problem. Tuning allowed the model to better capture the underlying structure-activity relationships, leading to more accurate predictions.

4.2 GridSearchCV vs. RandomizedSearchCV: A Practical Comparison

While GridSearchCV exhaustively explores every combination in a defined grid, RandomizedSearchCV samples a fixed number of combinations. In this study, RandomizedSearchCV slightly outperformed GridSearchCV (R^2 of 0.546 vs. 0.533) on the test set. This outcome, despite RandomizedSearchCV exploring a smaller fraction of the full parameter space (10 iterations vs. 108 combinations in GridSearchCV), highlights its efficiency. It suggests that the optimal hyperparameters might not be at the exact grid points but rather within the broader distributions sampled by RandomizedSearchCV, or that its random sampling simply chanced upon a better combination in this instance. For larger hyperparameter spaces, RandomizedSearchCV is generally the preferred method due to its computational efficiency.

4.3 Fingerprints vs. Descriptors: A Trade-off

The comparison between fingerprint-based and descriptor-based models reveals a classic trade-off in cheminformatics:

- **Predictive Power:** Fingerprint-based models, particularly the tuned ones, consistently achieved higher R-squared values and lower RMSE. This is expected, as high-dimensional fingerprints (2048 bits) can encode a vast amount of intricate structural information that pre-defined descriptors might miss.
- **Interpretability:** The RDKit descriptor-based model, despite having lower predictive performance, offers a significant advantage in interpretability. Its top-performing features are directly related to recognizable chemical properties (e.g., molecular density, specific functional groups like pyridine or sulfide, molecular connectivity). This interpretability is invaluable for medicinal chemists, as it can directly guide the rational design of new compounds by indicating which molecular features are likely to enhance or diminish activity. In contrast, interpreting individual fingerprint bits is a more complex task that often requires specialized tools to map bits back to chemical substructures.

4.4 Key Insights from Feature Importances

The analysis of feature importances provides valuable chemical insights into compounds active against SARS-CoV Mpro:

- **Molecular Density & Complexity:** Descriptors like `FpDensityMorgan` and `AvgIpc` emphasize that not just the presence of certain groups, but their arrangement and the overall compactness/complexity of the molecule, are important. This could relate to how well a molecule "fits" into the enzyme's active site.
- **Specific Substructures:** The high importance of `fr_pyridine` and `fr_sulfide` is a direct chemical hypothesis. It suggests that incorporating these functional groups might be beneficial for Mpro inhibition. This provides concrete starting points for structure-activity relationship (SAR) studies.
- **Molecular Connectivity:** `Chi` descriptors (e.g., `Chi4v`, `Chi3v`) highlight the significance of molecular topology and branching. This can inform how flexible or rigid a molecule should be for optimal binding.
- **Overall Physicochemical Properties:** `MolLogP` and `TPSA` are still relevant, indicating that general drug-like properties related to lipophilicity and polarity also play a role in the compound's journey to and interaction with the target.

4.5 Limitations and Future Directions

While this research provides a robust QSAR modeling framework, several limitations and areas for future work exist:

1. **Dataset Size and Diversity:** The current dataset of 245 compounds, while specific, is relatively small. Larger and more chemically diverse datasets could lead to more generalizable and higher-performing models.
2. **Single Target Focus:** The study was limited to one specific coronavirus protease. Future work could expand to other viral targets (e.g., RdRp, helicase) or integrate data for multiple targets to develop broader antiviral QSARs.
3. **Model Robustness (LOOCV/LPOCV):** The challenges encountered with LOOCV (NaN results) for this dataset size suggest that while K-Fold is suitable, for ultimate statistical rigor, acquiring a larger dataset or carefully debugging the LOOCV output interpretation might be necessary.
4. **Algorithm Exploration:** RandomForestRegressor is a strong baseline, but other advanced ensemble methods (e.g., XGBoost, LightGBM) or deep learning approaches (e.g., Graph Neural Networks for molecules) could be explored for potentially higher accuracy, albeit at the cost of interpretability.
5. **External Validation:** The models were evaluated on an internal test set and through cross-validation. For real-world deployment, rigorous external validation with a completely new set of experimentally validated compounds is crucial to confirm true generalization capability.
6. **Active Learning and Experimental Feedback:** The computational models developed here could be integrated into an active learning loop, where predictions guide the synthesis and testing of a few novel compounds, whose experimental data then retrains and improves the model in an iterative cycle.
7. **Bit Interpretation for Fingerprints:** Further work could explicitly map the most important Morgan fingerprint bits back to the 2D chemical substructures they represent, providing more detailed insights for medicinal chemists.

---

5. Conclusion

This research successfully established and optimized a QSAR modeling pipeline for predicting the activity of SARS-CoV Mpro inhibitors. The application of hyperparameter tuning, especially RandomizedSearchCV, demonstrably improved model performance, leading to a more predictive QSAR model. The comparative analysis of feature types highlighted the critical trade-off between model predictive power (often higher with fingerprints) and the direct chemical interpretability (offered by molecular descriptors). The identified key descriptors provide valuable, chemically actionable insights into the properties that drive inhibitory activity, thereby contributing to rational drug design efforts in the ongoing fight against coronaviruses. This robust computational framework serves as a foundational step for future lead discovery and optimization campaigns.

---

6. References

- **ChEMBL Database:** European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI). ChEMBL. Available from: https://www.ebi.ac.uk/chembl/
- **RDKit:** Open-source cheminformatics and machine learning software. Available from: https://www.rdkit.org/
- **Scikit-learn:** Machine learning in Python. Available from: https://scikit-learn.org/
- **Lipinski, C. A., et al.** (2001). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews, 46*(1-3), 3-26.
- **Rogers, D., & Hahn, M.** (2010). Extended connectivity fingerprints. *Journal of chemical information and modeling, 50*(5), 742-754.
