Drug Discovery using Random Forest Regression

Project by Rituraj Mahato 21BCE2762

01

Introduction

The Challenge

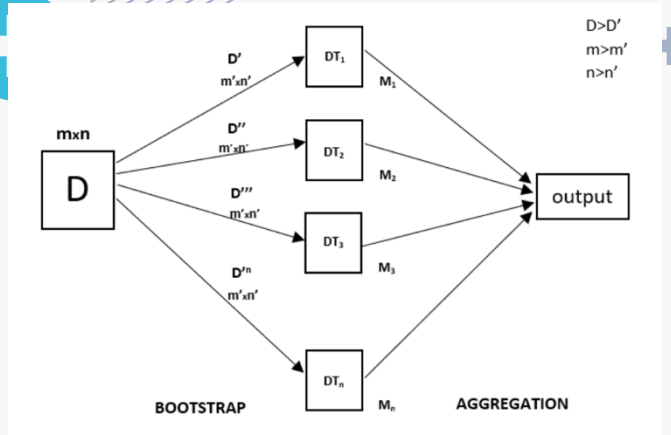
- The COVID-19 pandemic, caused by the SARS-CoV-2 virus highlighted the urgent need for rapid drug discovery methods.
- Traditional drug discovery methods were time-consuming and often inadequate for tackling fast-evolving pandemics.

Machine Learning to the Rescue

By applying ML algorithm to large amount of chemical and biological data, we can identify potential drug targets and molecules that could inhibit viral activity.

Random Forest Algorithm

- A powerful and versatile machine learning algorithm, which operates by constructing a multitude of decision tress during training and combining their outputs to enhance accuracy and robustness.
- Random Forest Regression is a specific application of the RF algorithm tailored for regression problems.
- It uses the same underlying principles of building multiple decision trees and averaging their predictions, but focuses on predicting continuous numerical values.



Random Forest Regression

Ensemble Learning

Supervised learning algorithm based on the concept of ensemble learning, specifically bagging/bootstrap aggregating.

This approach utilizes multiple decision trees trained on different subsets of the data to create a robust model.

Parallel Processing

The trees in a Random Forest operate in parallel, without any interaction during their construction.

This parallelisation allows for efficient training and prediction.

Feature Importance

Can rank features based on their importance in predicting the target variable.

This feature helps identify the most influential molecular descriptors affecting a compound's activity against the virus.

Why RF Regression is Suitable?

Modeling Complexity

Model non-linear relationships effectively, which is crucial because the relationship between molecular descriptors and drug activity is often complex and non-linear.

Feature Importance

It provides insights into the importance of different molecular descriptors in predicting the target variable.

Resists Overfitting

Less prone to overfitting compared to single decision trees which makes it a better choice for drug discovery.

What’s going in and getting out?

- Let’s start with the input data,

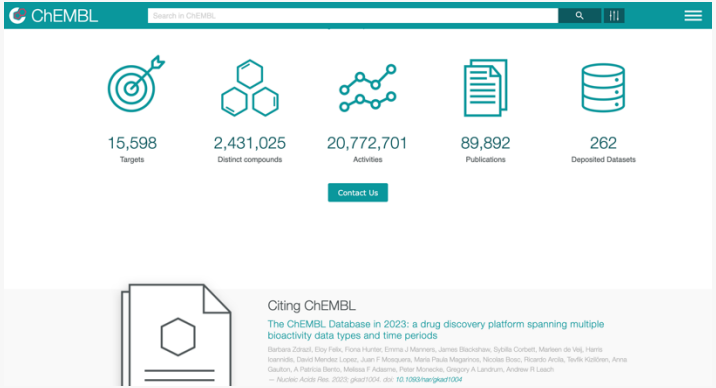
1. Selecting a protein target from ChEMBL database.
2. The bioactivity data (IC50 values) for the target is also fetched.
3. Features are molecular descriptors, which serve as the input for our model.
4. The model uses features (independent) and pIC50 values (dependent) to build regression models.

- **Random Forest Regressor** computes R-Square score on a test dataset to evaluate how well it predicts bioactivity (pIC50) from the input features.

Methodology - 1

Data Collection

The dataset was sourced from the ChEMBL database, a comprehensive resource for bioactivity data.



Data Preprocessing

Includes handling missing data, transforming IC50 values into pIC50 values, labelling compounds as active, inactive or intermediate.

```
#Selecting SARS Coronavirus 3C-like proteinase as target protein
selected_target = targets.target_chembl_id[4]
selected_target
```

```
'CHEMBL3927'
```

```
#Retrieving bioactivity data
activity = new_client.activity
res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
df = pd.DataFrame.from_dict(res)
df.head()
```

	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_variant_mutation	bao_e
0	None	1480935	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
1	None	1480936	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
2	None	1481061	[]	CHEMBL830868	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
3	None	1481065	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
4	None	1481066	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C

```
!pip install chembl_webresource_client
!pip install rdkit -q
import pandas as pd
import numpy as np
import seaborn as sns
sns.set(style='ticks')
import matplotlib.pyplot as plt
from chembl_webresource_client.new_client import new_client
from rdkit import Chem
from rdkit.Chem import Descriptors, Lipinski
```

```
Looking in indexes: https://pypi.org/simple, https://us-python
Collecting chembl_webresource_client
  Downloading chembl_webresource_client-0.10.8-py3-none-any.wh
                                              55.2/55.2 kB 3.6
Requirement already satisfied: urllib3 in /usr/local/lib/pytho
Requirement already satisfied: requests>=2.18.4 in /usr/local/
Collecting requests-cache~=0.7.0 (from chembl_webresource_clie
```

```
#Target protein search for Coronavirus
target = new_client.target
target_search = target.search('Coronavirus')
targets = pd.DataFrame.from_dict(target_search)
targets
```

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Coronavirus	Coronavirus	17.0	False	CHEMBL613732	[]	ORGANISM	11119
1	[]	SARS coronavirus	SARS coronavirus	15.0	False	CHEMBL612575	[]	ORGANISM	227859
2	[]	Feline coronavirus	Feline coronavirus	15.0	False	CHEMBL612744	[]	ORGANISM	12663
3	[]	Human coronavirus 229E	Human coronavirus 229E	13.0	False	CHEMBL613837	[]	ORGANISM	11137
4	[{'xref_id': 'P0C6U8', 'xref_name': None, 'xre...	SARS coronavirus	SARS coronavirus 3C-like proteinase	10.0	False	CHEMBL3927	[{'accession': 'P0C6U8', 'component_descriptio...	SINGLE PROTEIN	227859
5	[]	Middle East respiratory syndrome-related coron...	Middle East respiratory syndrome-related coron...	9.0	False	CHEMBL4296578	[]	ORGANISM	1335626
6	[{'xref_id': 'P0C6X7', 'xref_name': None, 'xre...	SARS coronavirus	Replicase polyprotein 1ab	4.0	False	CHEMBL5118	[{'accession': 'P0C6X7', 'component_descriptio...	SINGLE PROTEIN	227859
7	[]	Severe acute respiratory syndrome coronavirus 2	Replicase polyprotein 1ab	4.0	False	CHEMBL4523582	[{'accession': 'P0DTD1', 'component_descriptio...	SINGLE PROTEIN	2697049

```
#Selecting SARS Coronavirus 3C-like proteinase as target protein
selected_target = targets.target_chembl_id[4]
selected_target
```

```
'CHEMBL3927'
```

```
#Retrieving bioactivity data
activity = new_client.activity
res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
df = pd.DataFrame.from_dict(res)
df.head()
```

	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_variant_mutation	bao_e
0	None	1480935	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
1	None	1480936	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
2	None	1481061	[]	CHEMBL830868	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
3	None	1481065	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C
4	None	1481066	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAO_C

```
df.to_csv('bioactivity_data.csv', index=False)
data = pd.read_csv("bioactivity_data.csv")
print(data.columns)
```

```
Index(['activity_comment', 'activity_id', 'activity_properties',
       'assay_chembl_id', 'assay_description', 'assay_type',
       'assay_variant_accession', 'assay_variant_mutation', 'bao_endpoint',
       'bao_format', 'bao_label', 'canonical_smiles', 'data_validity_comment',
       'data_validity_description', 'document_chembl_id', 'document_journal',
       'document_year', 'ligand_efficiency', 'molecule_chembl_id',
       'molecule_pref_name', 'parent_molecule_chembl_id', 'pchembl_value',
       'potential_duplicate', 'qudt_units', 'record_id', 'relation', 'src_id',
       'standard_flag', 'standard_relation', 'standard_text_value',
       'standard_type', 'standard_units', 'standard_upper_value',
       'standard_value', 'target_chembl_id', 'target_organism',
       'target_pref_name', 'target_tax_id', 'text_value', 'toid', 'type',
       'units', 'uo_units', 'upper_value', 'value'],
      dtype='object')
```

```
print(df.isnull().sum())
```

```
activity_comment             133
activity_id                    0
activity_properties            0
assay_chembl_id                0
assay_description              0
assay_type                     0
assay_variant_accession      133
assay_variant_mutation       133
bao_endpoint                   0
bao_format                     0
bao_label                      0
canonical_smiles               0
data_validity_comment        106
data_validity_description    106
document_chembl_id             0
```

```
#Finding and dropping rows with missing values in the standard_value column
df2 = df[df.standard_value.notna()]
df2
```

	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_variant_mutation	bao
0	None	1480935	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAC
1	None	1480936	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAC
2	None	1481061	[]	CHEMBL830868	In vitro inhibitory concentration against SARS...	B	None	None	BAC
3	None	1481065	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAC
4	None	1481066	[]	CHEMBL829584	In vitro inhibitory concentration against SARS...	B	None	None	BAC
...	...	...	...	...	...	...	...	...	
128	None	12041507	[]	CHEMBL2150313	Inhibition of SARS-CoV PLpro expressed in Esch...	B	None	None	BAC
129	None	12041508	[]	CHEMBL2150313	Inhibition of SARS-CoV PLpro expressed in Esch...	B	None	None	BAC

```python
#Labelling compounds as active,inactive, or intermediate using IC50 values
bioactivity_class = []
for i in df2.standard_value:
  if float(i) >= 10000:
    bioactivity_class.append("inactive")
  elif float(i) <= 1000:
    bioactivity_class.append("active")
  else:
    bioactivity_class.append("intermediate")
```

```python
#Selecting necessary columns
selection = ['molecule_chembl_id', 'canonical_smiles', 'standard_value']
df3 = df2[selection]
df3
pd.concat([df3,pd.Series(bioactivity_class)], axis=1)
df3.to_csv('bioactivity_preprocessed_data.csv', index=False)
df = pd.read_csv('bioactivity_preprocessed_data.csv')
```

Methodology - 2

Exploratory Data Analysis

To understand the chemical space od the dataset and gain insights into the distribution of molecular descriptors.

Lipinski descriptors are used to analyse the chemical space, and visualizations are generated to aid the understanding of the dataset.

Descriptor Calculation

Molecular fingerprints are generated using PaDEL descriptor software. These descriptors provide a quantitative representation of the chemical structures of the molecules.

The output of this step is X and Y, where X(molecular descriptors) and Y (pIC50 values), which are used for model training.

What is Lipinski and PaDEL descriptor?

- Lipinski Descriptor – A set of rules used to evaluate the drug-likeness of chemical compound.

1. Molecular Weight: Less than 500 Da (Dalton)
2. LogP: Less than 5 (Solvency in Liquids)
3. Hydrogen Bond Donors: Less than 5
4. Hydrogen Bond Acceptors: Less than 10

- PaDEL Descriptor Software used for calculating molecular descriptors. It’s widely used in drug discovery and cheminformatics research.
- Its open-source, efficient, easy to use and accurate as well.

```
def lipinski(smiles, verbose=False):

    moldata= []
    for elem in smiles:
        mol=Chem.MolFromSmiles(elem)
        moldata.append(mol)

    baseData= np.arange(1,1)
    i=0
    for mol in moldata:

        desc_MolWt = Descriptors.MolWt(mol)
        desc_MolLogP = Descriptors.MolLogP(mol)
        desc_NumHDonors = Lipinski.NumHDonors(mol)
        desc_NumHAcceptors = Lipinski.NumHAcceptors(mol)

        row = np.array([desc_MolWt,
                        desc_MolLogP,
                        desc_NumHDonors,
                        desc_NumHAcceptors])

        if(i==0):
            baseData=row
        else:
            baseData=np.vstack([baseData, row])
        i=i+1

    columnNames=["MW","LogP","NumHDonors","NumHAcceptors"]
    descriptors = pd.DataFrame(data=baseData,columns=columnNames)

    return descriptors

df_lipinski = lipinski(df.canonical_smiles)
df_lipinski
```

	MW	LogP	NumHDonors	NumHAcceptors
0	281.271	1.89262	0.0	5.0
1	415.589	3.81320	0.0	2.0
2	421.190	2.66050	0.0	4.0
3	293.347	3.63080	0.0	3.0
4	338.344	3.53900	0.0	5.0
...	...	...	...	...
128	338.359	3.40102	0.0	5.0

```
df_combined = pd.concat([df,df_lipinski], axis=1)
df_combined.head()
```

	molecule_chembl_id	canonical_smiles	standard_value	MW	LogP	NumHDonors	NumHAcceptors
0	CHEMBL187579	Cc1noc(C)c1CN1C(=O)C(=O)c2cc(C#N)ccc21	7200.0	281.271	1.89262	0.0	5.0
1	CHEMBL188487	O=C1C(=O)N(Cc2ccc(F)cc2Cl)c2ccc(I)cc21	9400.0	415.589	3.81320	0.0	2.0
2	CHEMBL185698	O=C1C(=O)N(CC2COc3ccccc3O2)c2ccc(I)cc21	13500.0	421.190	2.66050	0.0	4.0
3	CHEMBL426082	O=C1C(=O)N(Cc2cc3ccccc3s2)c2ccccc21	13110.0	293.347	3.63080	0.0	3.0
4	CHEMBL187717	O=C1C(=O)N(Cc2cc3ccccc3s2)c2c1cccc2[N+](=O)[O-]	2000.0	338.344	3.53900	0.0	5.0

```
#Converting IC50 to pIC50 for uniform distribution
def pIC50(input):
    pIC50 = []

    for i in input['standard_value_norm']:
        molar = i*(10**-9) # Converts nM to M
        pIC50.append(-np.log10(molar))

    input['pIC50'] = pIC50
    x = input.drop('standard_value_norm', 1)

    return x
```

```
df_combined.standard_value.describe()
```

```
count       133.000000
mean      85967.130075
std      158897.319181
min         50.000000
25%      10100.000000
50%      17500.000000
75%      70000.000000
max    1000000.000000
Name: standard_value, dtype: float64
```

```
df_norm.standard_value_norm.describe()
```

```
count       133.000000
mean      85967.130075
std      158897.319181
min          50.000000
25%       10100.000000
50%       17500.000000
75%       70000.000000
max     1000000.000000
Name: standard_value_norm, dtype: float64
```

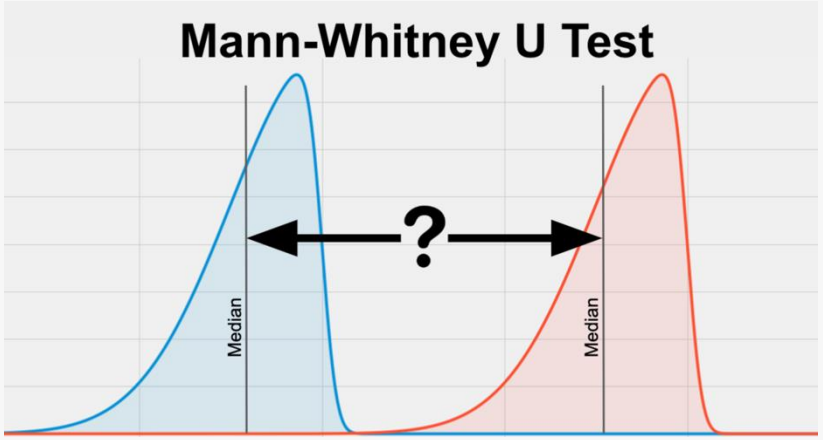
```
bioactivity_class = []
for i in df_norm.standard_value_norm:
  if float(i) >= 10000:
    bioactivity_class.append("inactive")
  elif float(i) <= 1000:
    bioactivity_class.append("active")
  else:
    bioactivity_class.append("intermediate")
df_norm1 = pd.concat([df_norm, pd.Series(bioactivity_class).rename('bioactivity_class')],axis=1)
df_final = pIC50(df_norm1)
df_final
```

	molecule_chembl_id	canonical_smiles	MW	LogP	NumHDonors	NumHAcceptors	bioactivity_class	pIC50
0	CHEMBL187579	Cc1noc(C)c1CN1C(=O)C(=O)c2cc(C#N)ccc21	281.271	1.89262	0.0	5.0	intermediate	5.142668
1	CHEMBL188487	O=C1C(=O)N(Cc2ccc(F)cc2Cl)c2ccc(I)cc21	415.589	3.81320	0.0	2.0	intermediate	5.026872
2	CHEMBL185698	O=C1C(=O)N(CC2COc3ccccc3O2)c2ccc(I)cc21	421.190	2.66050	0.0	4.0	inactive	4.869666
3	CHEMBL426082	O=C1C(=O)N(Cc2cc3ccccc3s2)c2ccccc21	293.347	3.63080	0.0	3.0	inactive	4.882397
4	CHEMBL187717	O=C1C(=O)N(Cc2cc3ccccc3s2)c2c1cccc2[N+](=O)[O-]	338.344	3.53900	0.0	5.0	intermediate	5.698970
...	...	...	...	...	...	...	...	...
128	CHEMBL2146517	COC(=O)[C@@]1(C)CCCc2c1ccc1c2C(=O)C(=O)c2c(C)c...	338.359	3.40102	0.0	5.0	inactive	4.974694
129	CHEMBL187460	C[C@H]1COC2=C1C(=O)C(=O)c1c2ccc2c1CCCC2(C)C	296.366	3.44330	0.0	3.0	inactive	4.995679
130	CHEMBL363535	Cc1coc2c1C(=O)C(=O)c1c-2ccc2c(C)cccc12	276.291	4.09564	0.0	3.0	inactive	4.939302
131	CHEMBL227075	Cc1cccc2c3c(ccc12)C1=C(C(=O)C3=O)[C@@H](C)CO1	278.307	3.29102	0.0	3.0	inactive	4.970616
132	CHEMBL45830	CC(C)C1=Cc2ccc3c(c2C(=O)C1=O)CCCC3(C)C	282.383	4.10530	0.0	2.0	inactive	4.102923

Mann-Whitney U Test

- The Mann-Whitney U Test, also known as the Wilcoxon Rank Sum Test, is a non-parametric statistical test used to compare two samples or groups.

1. Does not assume any specific distribution.
2. Ranks all the data points from both groups.
3. P-value: Used to determine statistical significance.
4. Individual sample sizes when small is better.



```python
# Mann Whitney U Test for statistical analysis
def mannwhitney(descriptor, verbose=False):
  from numpy.random import seed
  from numpy.random import randn
  from scipy.stats import mannwhitneyu

# seed the random number generator
  seed(1)

# actives and inactives
  selection = [descriptor, 'bioactivity_class']
  df = df_2class[selection]
  active = df[df.bioactivity_class == 'active']
  active = active[descriptor]

  selection = [descriptor, 'bioactivity_class']
  df = df_2class[selection]
  inactive = df[df.bioactivity_class == 'inactive']
  inactive = inactive[descriptor]

# compare samples
  stat, p = mannwhitneyu(active, inactive)
  #print('Statistics=%.3f, p=%.3f' % (stat, p))

# interpret
  alpha = 0.05
  if p > alpha:
    interpretation = 'Same distribution (fail to reject H0)'
  else:
    interpretation = 'Different distribution (reject H0)'

  results = pd.DataFrame({'Descriptor':descriptor,
                          'Statistics':stat,
                          'p':p,
                          'alpha':alpha,
                          'Interpretation':interpretation}, index=[0])
  filename = 'mannwhitneyu_' + descriptor + '.csv'
  results.to_csv(filename)

  return results
```

```python
mannwhitney('pIC50')
```

	Descriptor	Statistics	p	alpha	Interpretation
0	pIC50	1545.0	4.428384e-10	0.05	Different distribution (reject H0)

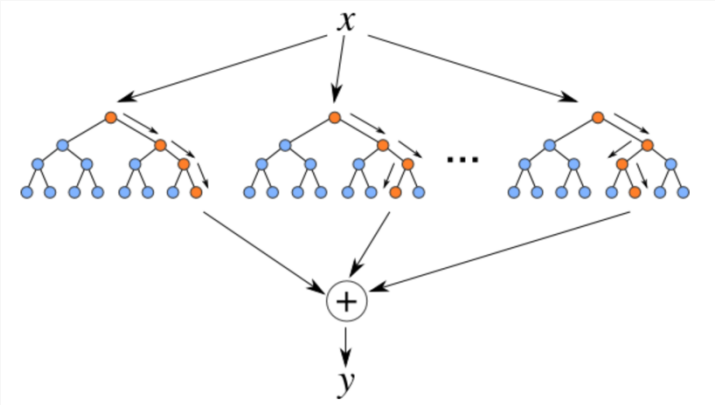
Methodology - 3

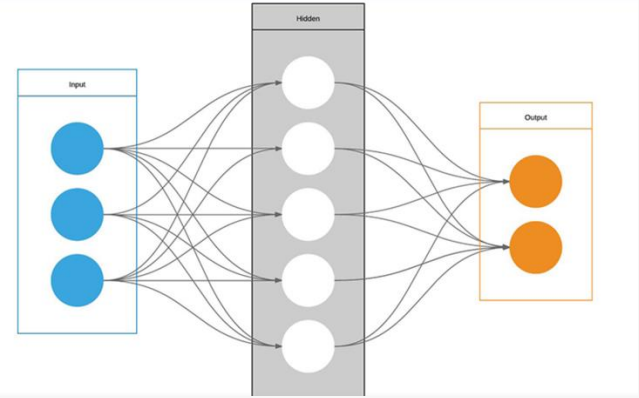
Model Building

A Random Forest Regression model is trained to predict pIC50 values based on the molecular descriptors.

Model Evaluation

The performance of the trained model is evaluated using various metrics.





```
df3 = pd.read_csv('acetylcholinesterase_04_bioactivity_data_3class_pIC50.csv')
df3
```

	molecule_chembl_id	canonical_smiles	MW	LogP	NumHDonors	NumHAcceptors	bioactivity_class	pIC50
0	CHEMBL187579	Cc1noc(C)c1CN1C(=O)C(=O)c2cc(C#N)ccc21	281.271	1.89262	0.0	5.0	intermediate	5.142668
1	CHEMBL188487	O=C1C(=O)N(Cc2ccc(F)cc2Cl)c2ccc(I)cc21	415.589	3.81320	0.0	2.0	intermediate	5.026872
2	CHEMBL185698	O=C1C(=O)N(CC2COc3ccccc3O2)c2ccc(I)cc21	421.190	2.66050	0.0	4.0	inactive	4.869666
3	CHEMBL426082	O=C1C(=O)N(Cc2cc3ccccc3s2)c2ccccc21	293.347	3.63080	0.0	3.0	inactive	4.882397
4	CHEMBL187717	O=C1C(=O)N(Cc2cc3ccccc3s2)c2c1cccc2[N+](=O)[O-]	338.344	3.53900	0.0	5.0	intermediate	5.698970
...	...	...	...	...	...	...	...	...
128	CHEMBL2146517	COC(=O)[C@@]1(C)CCCc2c1ccc1c2C(=O)C(=O)c2c(C)c...	338.359	3.40102	0.0	5.0	inactive	4.974694
129	CHEMBL187460	C[C@H]1COC2=C1C(=O)C(=O)c1c2ccc2c1CCCC2(C)C	296.366	3.44330	0.0	3.0	inactive	4.995679
130	CHEMBL363535	Cc1coc2c1C(=O)C(=O)c1c-2ccc2c(C)cccc12	276.291	4.09564	0.0	3.0	inactive	4.939302
131	CHEMBL227075	Cc1cccc2c3c(ccc12)C1=C(C(=O)C3=O)[C@@H](C)CO1	278.307	3.29102	0.0	3.0	inactive	4.970616
132	CHEMBL45830	CC(C)C1=Cc2ccc3c(c2C(=O)C1=O)CCCC3(C)C	282.383	4.10530	0.0	2.0	inactive	4.102923

```
X.shape
```

```
(133, 881)
```

```
Y.shape
```

```
(133,)
```

```
# Remove features with low variance
from sklearn.feature_selection import VarianceThreshold
selection = VarianceThreshold(threshold=(.8 * (1 - .8)))
X = selection.fit_transform(X)
```

```
X.shape
```

```
(133, 200)
```

```
# Splitting of data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

```
X_train.shape, Y_train.shape
```

```
((106, 200), (106,))
```

```
X_test.shape, Y_test.shape
```

```
((27, 200), (27,))
```

```
#Building regression model using random forest
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, Y_train)
r2 = model.score(X_test, Y_test)
r2
```

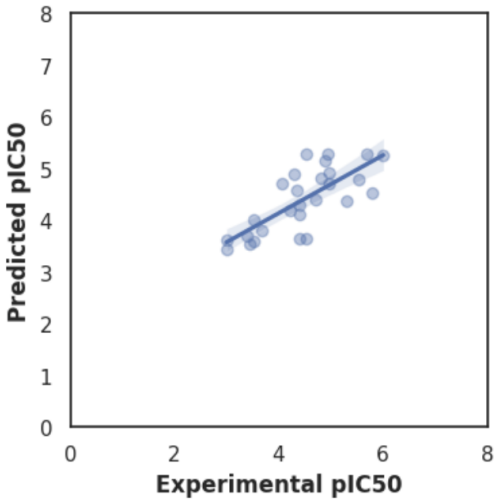
```
0.5580911193302558
```

```
Y_pred = model.predict(X_test)
```

```
#Plotting a scatterplot of predicted vs experimental IC50 values
sns.set(color_codes=True)
sns.set_style("white")

ax = sns.regplot(data= X, x=Y_test, y=Y_pred, scatter_kws={'alpha':0.4})
ax.set_xlabel('Experimental pIC50', fontsize='medium', fontweight='bold')
ax.set_ylabel('Predicted pIC50', fontsize='medium', fontweight='bold')
ax.set_xlim(0, 8)
ax.set_ylim(0, 8)
ax.figure.set_size_inches(4, 4)
plt.show
```

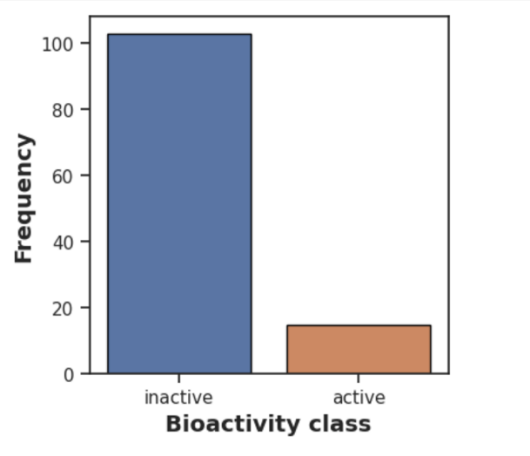
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



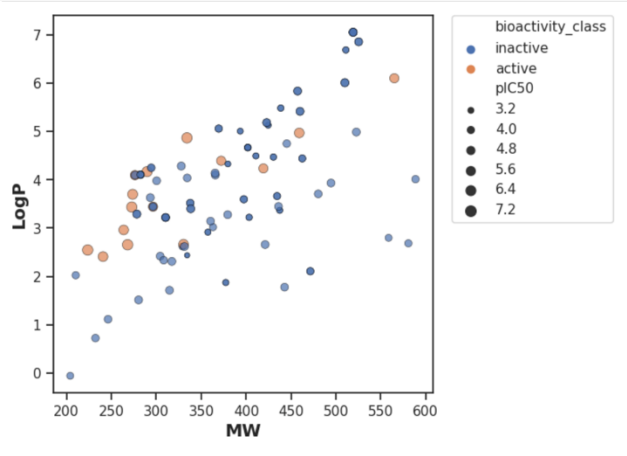
pIC50 values indicates how much drug is needed to inhibit a biological process by half.

Data Visualization

Frequency Plot



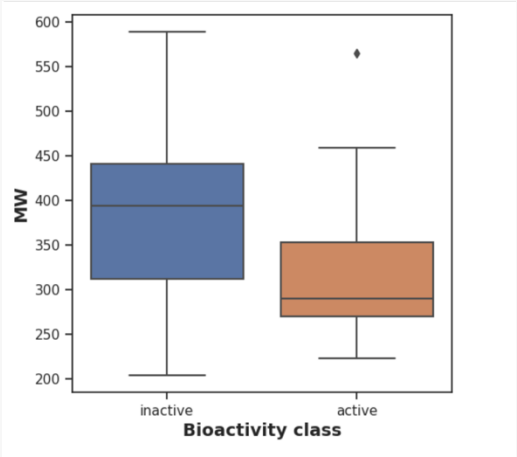
Scatter Plot



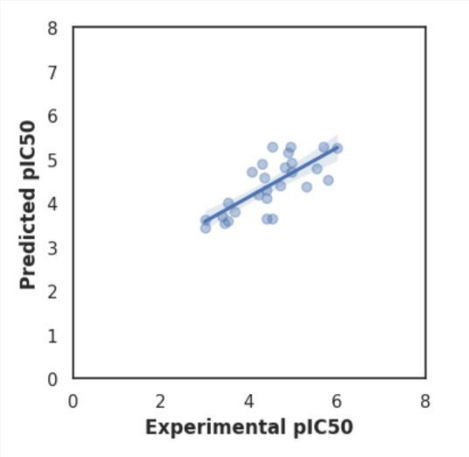
- pIC50 is often used instead of IC50 because it's easier to understand and communicate, and it's a better way to represent the potency of compounds

Data Visualization

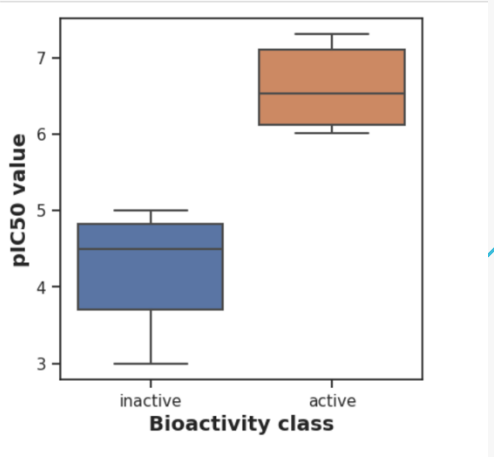
Box Plot



Scatter Plot



Box Plot



- pIC50 are negative logarithmic expressions of half maximal inhibitory concentration(IC50) values, which are used to measure the potency of a drug against biological targets: pIC50 = -log IC50

Various regression models evaluated using **LazyRegressor**, a Python library that benchmarks a wide range of regression algorithms quickly and efficiently.

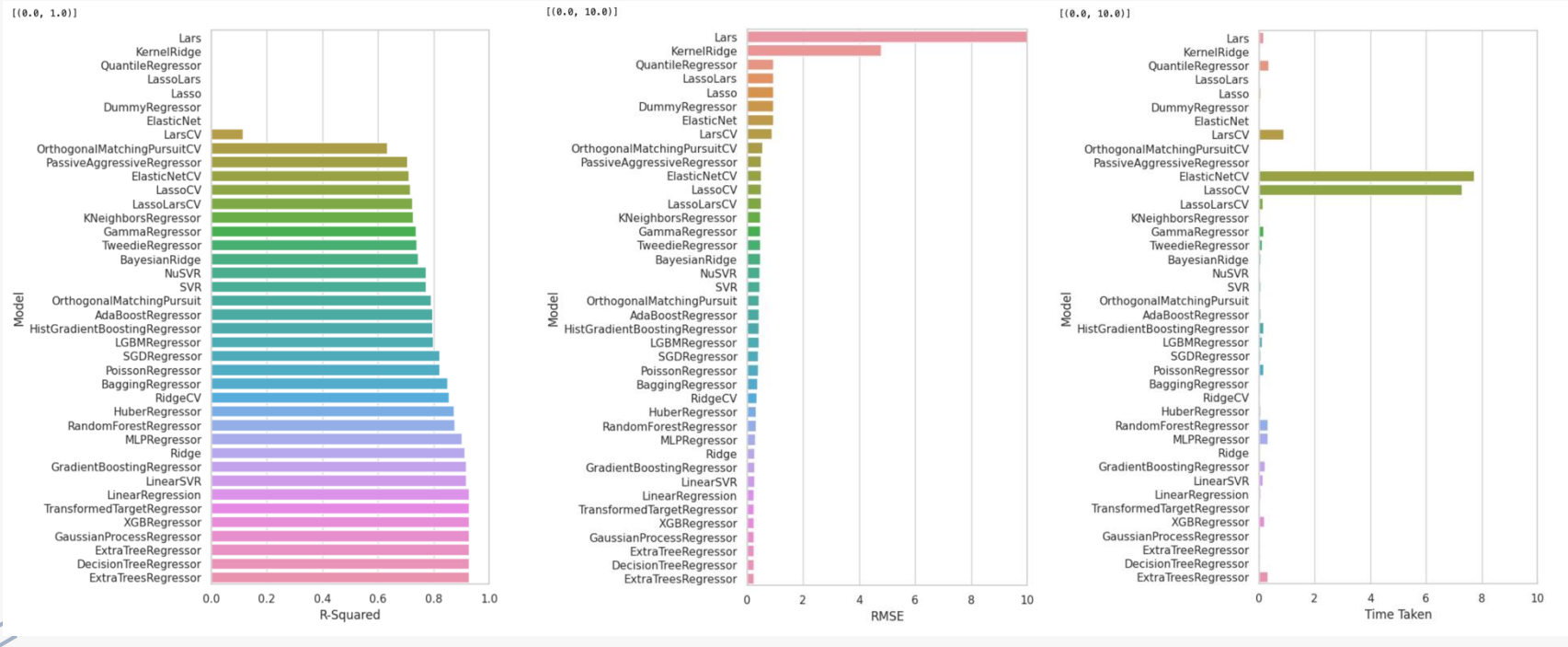
```
# Defines and builds the lazyclassifier
import lazypredict
from lazypredict.Supervised import LazyRegressor
clf = LazyRegressor(verbose=0,ignore_warnings=True, custom_metric=None)
models_train,predictions_train = clf.fit(X_train, X_train, Y_train, Y_train)
models_test,predictions_test = clf.fit(X_train, X_test, Y_train, Y_test)
```

```
100%|██████████| 42/42 [00:21<00:00,  1.92it/s]
```

```
# Performance table of the training set (80% subset)
predictions_train
```

Model	Adjusted R-Squared	R-Squared	
Lars	16224133809804306330339640563448489258949801140...	-14678978208870561930018975373859466193353144433...	3567214973
KernelRidge	30.29	-25.50	
QuantileRegressor	2.11	-0.00	
LassoLars	2.11	0.00	
Lasso	2.11	0.00	
DummyRegressor	2.11	0.00	
ElasticNet	2.11	0.00	
LarsCV	1.98	0.11	
OrthogonalMatchingPursuitCV	1.41	0.63	
PassiveAggressiveRegressor	1.32	0.71	
ElasticNetCV	1.32	0.71	
LassoCV	1.32	0.71	
LassoLarsCV	1.30	0.72	
KNeighborsRegressor	1.30	0.73	
GammaRegressor	1.29	0.74	
TweedieRegressor	1.29	0.74	

Performance using LazyRegressor



Results and conclusions

Active and Inactive Compounds
The analysis revealed that the defined threshold effectively classified compounds as active, inactive, or intermediate based on their pIC50 values.
Statistical Validation
Mann-Whitney U test confirmed significant differences in activity levels between various groups of compounds, further validating the model's ability to distinguish between active and inactive compounds.
Model Effectiveness
The model showed promising results in predicting drug activity.
Data Enrichment
Expanding the scope of data collection to include a wider range of molecules can improve the model's generalisation ability and increase the diversity of potential drug candidates.

Conclusions

01 Accelerated Drug Discovery

RF Regression, can significantly reduce the time required for drug discovery, making it a powerful tool for responding to pandemics and other urgent medical needs.

02 Broader Applications

Can be adapted for drug discovery targeting other diseases, which involves using appropriate databases, adjusting model parameters, and performing further feature engineering.