PREDICTIVE OF pIC50 VALUES OF BIOLOGICAL ACTIVITY COMPOUNDS AGAINST CERVICAL CANCER

Presented By,
Joy Blessing Selvaraj
195824132
V M.Sc. Bioinformatics

Internal Guide
Dr. K. Akila
Associate Professor
Head of the Department
Bishop Heber College,
Trichy.

External Guide
Ditto M Mohan
CEO & Founder
Biogrithm Bioinformatics
Solutions Pvt Limited,
Bangalore.

INTRODUCTION

•Cervical cancer is a type of cancer that develops in the cervix, the lower part of the uterus.

•It is primarily caused by persistent infection with high-risk strains of Human Papillomavirus (HPV) and can be prevented through vaccination and regular screening.

•Cyclin-Dependent Kinase 1 (CDK1) is a protein enzyme crucial for regulating the cell cycle, including cell division. In cervical cancer, dysregulation of CDK1 activity can lead to uncontrolled cell proliferation, tumor growth, and progression.

•CDK1 has been identified as a potential therapeutic target in cervical cancer treatment, with efforts focused on developing inhibitors to disrupt its function and halt cancer cell proliferation.

AIM

The aim of this project is to use machine learning techniques to forecast pIC50 values based on the bioactivity components of the target protein CDK1

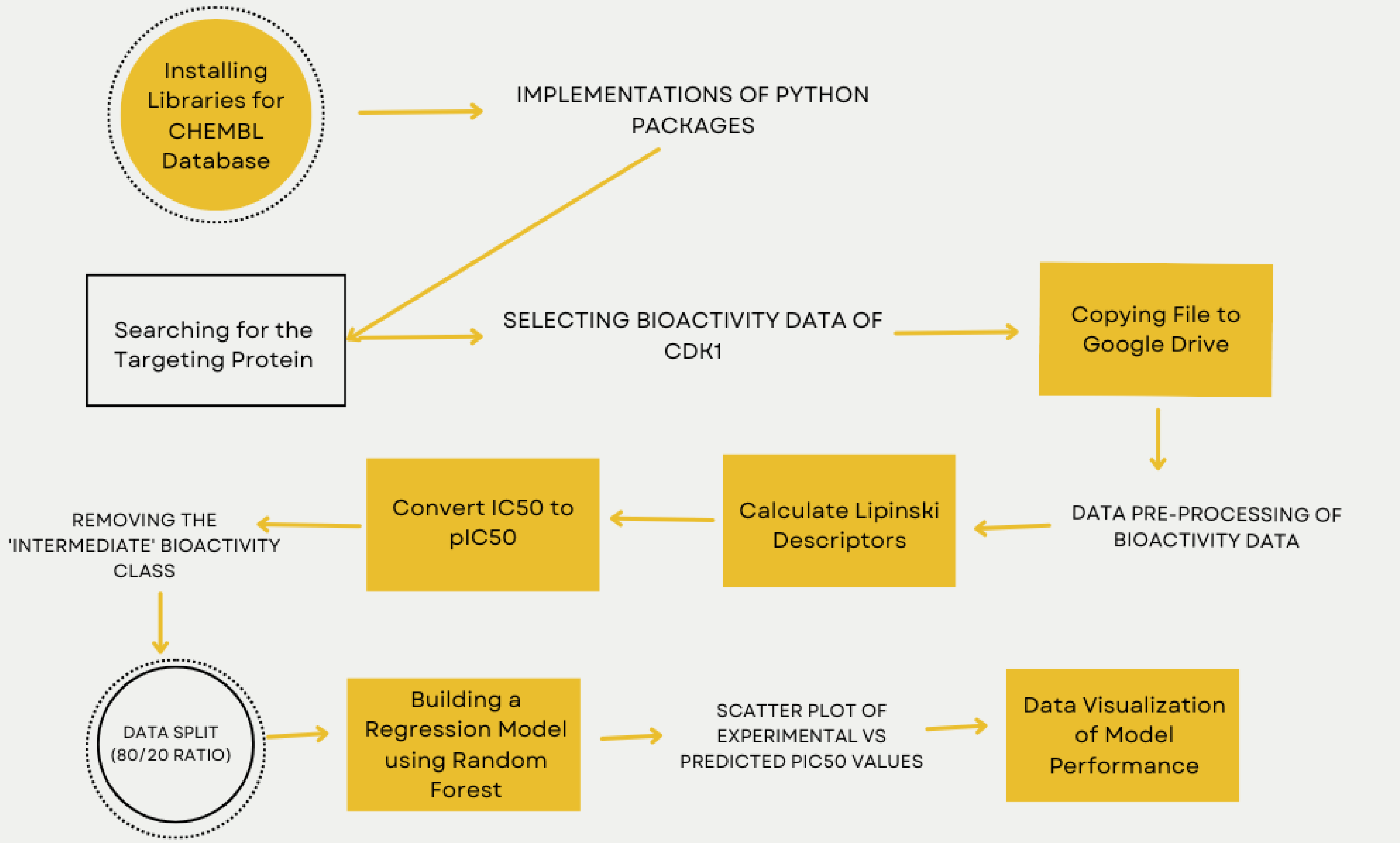
OBJECTIVE

•Developing a machine learning model to predict the pIC50 values of CDK1 protein

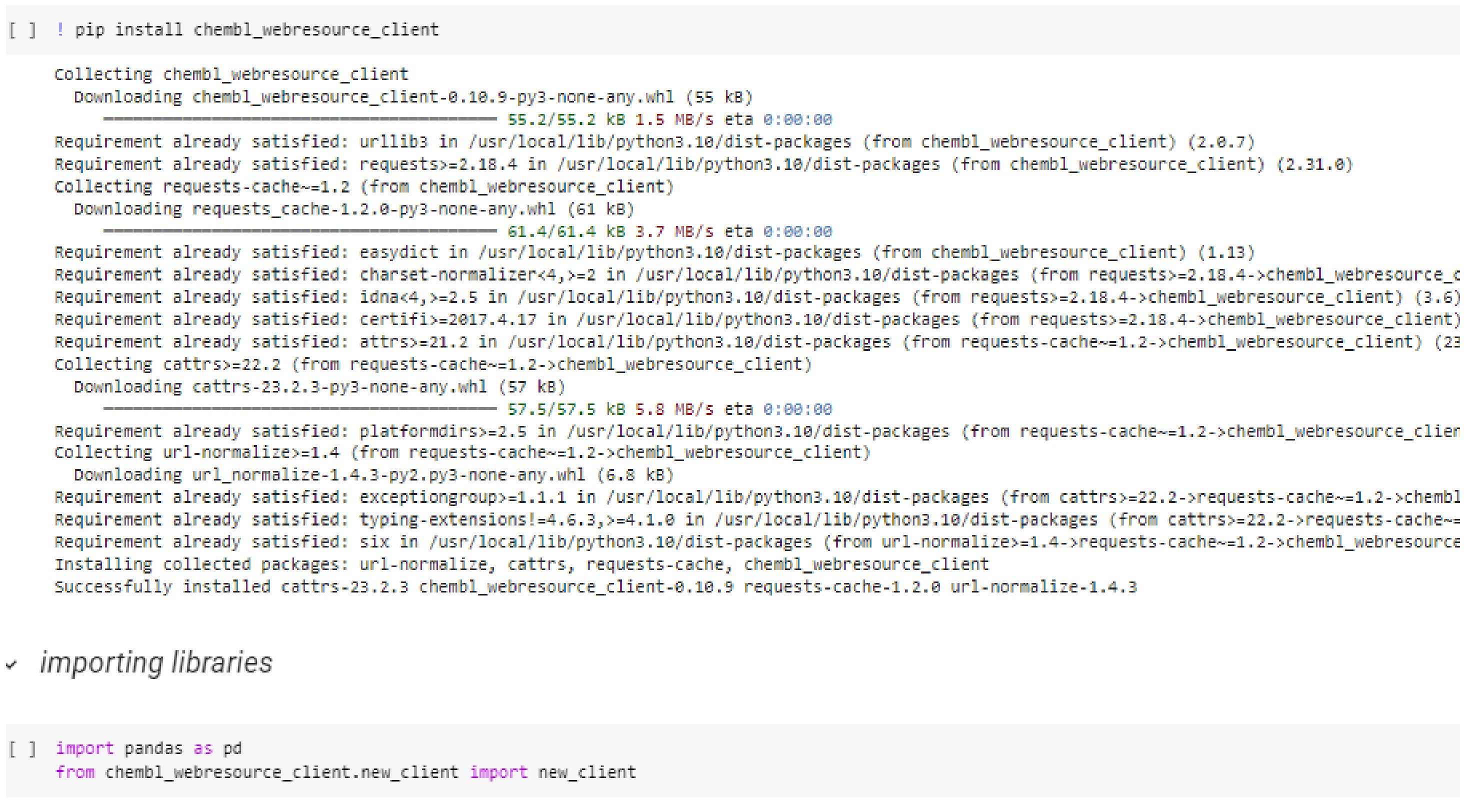
•Comparing different machine learning algorithms to the model to get the best forecasting algorithm for predicting pIC50 values.

•Analyzing the bioactivity molecules of the target protein CDK1 to determine the experimental pIC50 values, this will serve as the data set of inputs for a machine learning model

METHODOLOGY


Installing Libraries for CHEMBL Database
IMPLEMENTATIONS OF PYTHON PACKAGES
Searching for the Targeting Protein
SELECTING BIOACTIVITY DATA OF CDK1
Copying File to Google Drive
DATA PRE-PROCESSING OF BIOACTIVITY DATA
Calculate Lipinski Descriptors
Convert IC50 to pIC50
REMOVING THE 'INTERMEDIATE' BIOACTIVITY CLASS
DATA SPLIT (80/20 RATIO)
Building a Regression Model using Random Forest
SCATTER PLOT OF EXPERIMENTAL VS PREDICTED PIC50 VALUES
Data Visualization of Model Performance

RESULTS

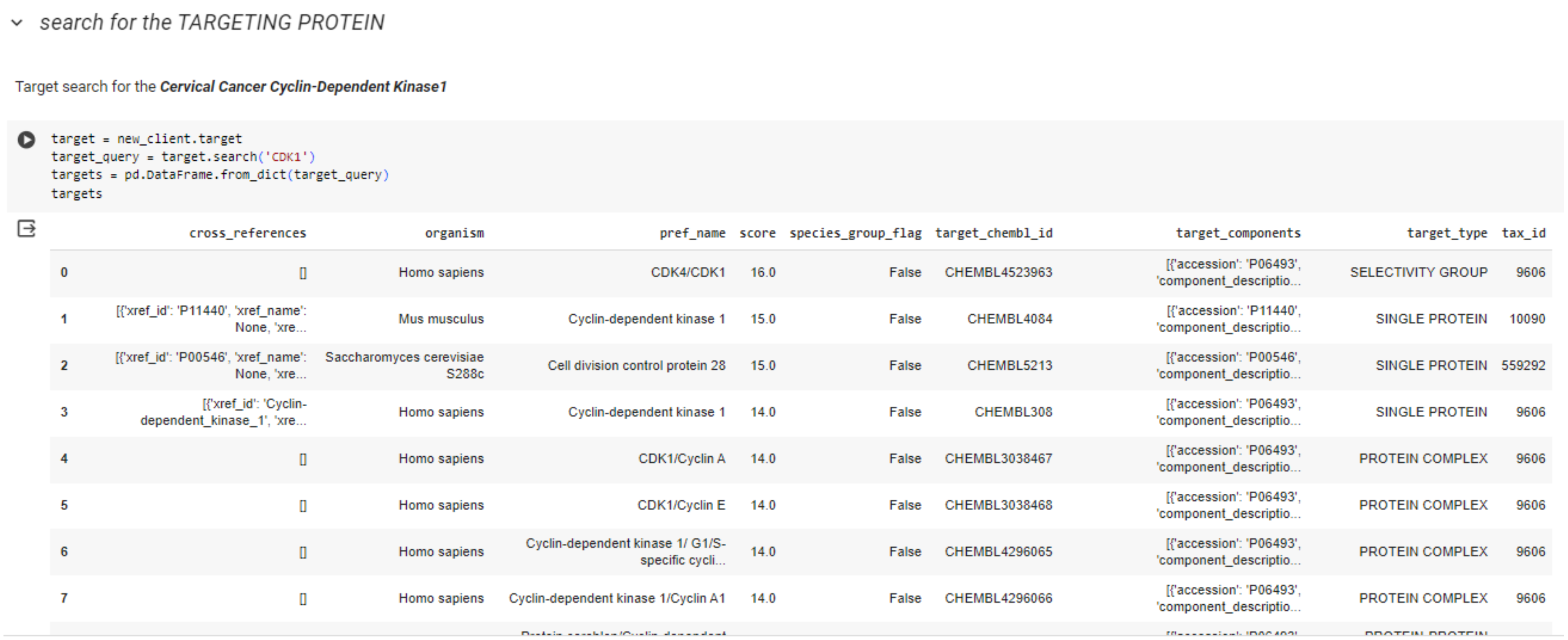


```
[ ]  ! pip install chembl_webresource_client

     Collecting chembl_webresource_client
       Downloading chembl_webresource_client-0.10.9-py3-none-any.whl (55 kB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.2/55.2 kB 1.5 MB/s eta 0:00:00
     Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from chembl_webresource_client) (2.0.7)
     Requirement already satisfied: requests>=2.18.4 in /usr/local/lib/python3.10/dist-packages (from chembl_webresource_client) (2.31.0)
     Collecting requests-cache~=1.2 (from chembl_webresource_client)
       Downloading requests_cache-1.2.0-py3-none-any.whl (61 kB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.4/61.4 kB 3.7 MB/s eta 0:00:00
     Requirement already satisfied: easydict in /usr/local/lib/python3.10/dist-packages (from chembl_webresource_client) (1.13)
     Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.18.4->chembl_webresource_c
     Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.18.4->chembl_webresource_client) (3.6)
     Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.18.4->chembl_webresource_client)
     Requirement already satisfied: attrs>=21.2 in /usr/local/lib/python3.10/dist-packages (from requests-cache~=1.2->chembl_webresource_client) (23
     Collecting cattrs>=22.2 (from requests-cache~=1.2->chembl_webresource_client)
       Downloading cattrs-23.2.3-py3-none-any.whl (57 kB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 5.8 MB/s eta 0:00:00
     Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests-cache~=1.2->chembl_webresource_clier
     Collecting url-normalize>=1.4 (from requests-cache~=1.2->chembl_webresource_client)
       Downloading url_normalize-1.4.3-py2.py3-none-any.whl (6.8 kB)
     Requirement already satisfied: exceptiongroup>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from cattrs>=22.2->requests-cache~=1.2->chembl
     Requirement already satisfied: typing-extensions!=4.6.3,>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from cattrs>=22.2->requests-cache~=
     Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from url-normalize>=1.4->requests-cache~=1.2->chembl_webresource
     Installing collected packages: url-normalize, cattrs, requests-cache, chembl_webresource_client
     Successfully installed cattrs-23.2.3 chembl_webresource_client-0.10.9 requests-cache-1.2.0 url-normalize-1.4.3
```

importing libraries

```
[ ]  import pandas as pd
     from chembl_webresource_client.new_client import new_client
```

Utilizing the Python command to install necessary libraries for accessing the CHEMBL database



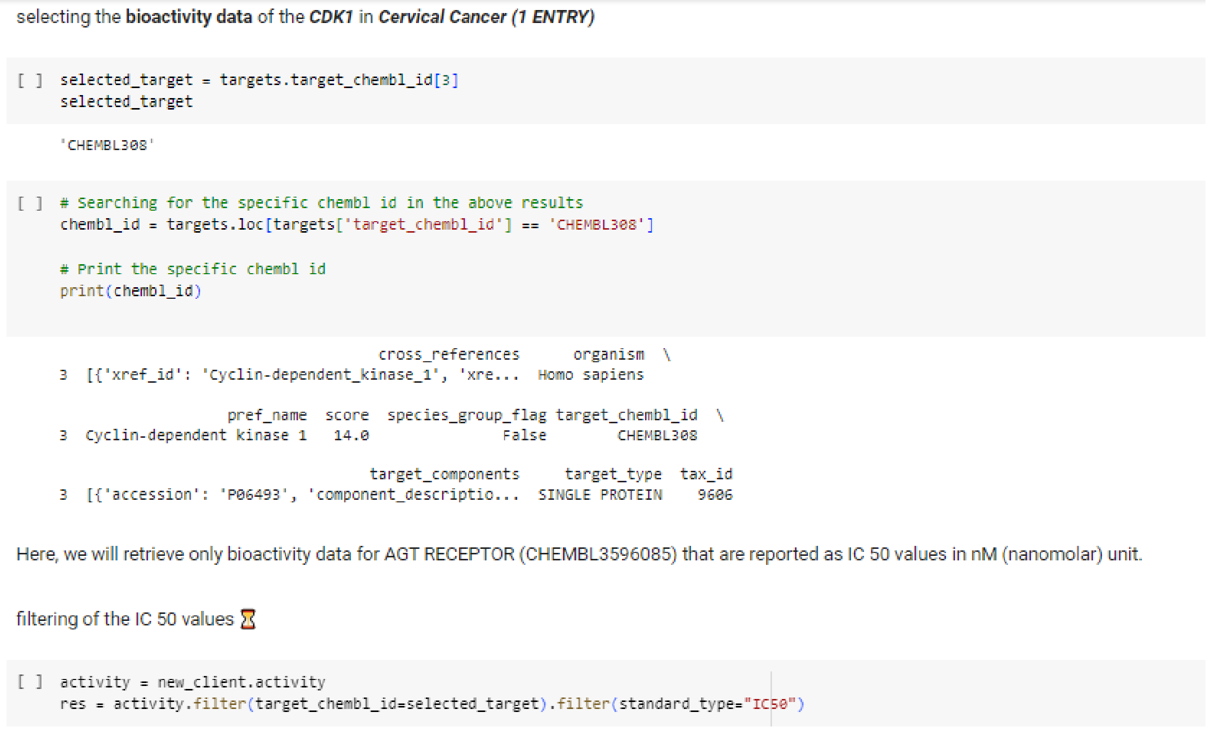
search for the TARGETING PROTEIN

Target search for the ***Cervical Cancer Cyclin-Dependent Kinase1***

```
target = new_client.target
target_query = target.search('CDK1')
targets = pd.DataFrame.from_dict(target_query)
targets
```

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Homo sapiens	CDK4/CDK1	16.0	False	CHEMBL4523963	[{'accession': 'P06493', 'component_descriptio...	SELECTIVITY GROUP	9606
1	[{'xref_id': 'P11440', 'xref_name': None, 'xre...	Mus musculus	Cyclin-dependent kinase 1	15.0	False	CHEMBL4084	[{'accession': 'P11440', 'component_descriptio...	SINGLE PROTEIN	10090
2	[{'xref_id': 'P00546', 'xref_name': None, 'xre...	Saccharomyces cerevisiae S288c	Cell division control protein 28	15.0	False	CHEMBL5213	[{'accession': 'P00546', 'component_descriptio...	SINGLE PROTEIN	559292
3	[{'xref_id': 'Cyclin-dependent_kinase_1', 'xre...	Homo sapiens	Cyclin-dependent kinase 1	14.0	False	CHEMBL308	[{'accession': 'P06493', 'component_descriptio...	SINGLE PROTEIN	9606
4	[]	Homo sapiens	CDK1/Cyclin A	14.0	False	CHEMBL3038467	[{'accession': 'P06493', 'component_descriptio...	PROTEIN COMPLEX	9606
5	[]	Homo sapiens	CDK1/Cyclin E	14.0	False	CHEMBL3038468	[{'accession': 'P06493', 'component_descriptio...	PROTEIN COMPLEX	9606
6	[]	Homo sapiens	Cyclin-dependent kinase 1/ G1/S-specific cycli...	14.0	False	CHEMBL4296065	[{'accession': 'P06493', 'component_descriptio...	PROTEIN COMPLEX	9606
7	[]	Homo sapiens	Cyclin-dependent kinase 1/Cyclin A1	14.0	False	CHEMBL4296066	[{'accession': 'P06493', 'component_descriptio...	PROTEIN COMPLEX	9606

Searching for the targeting protein (CDK1) in the CHEMBL database.


selecting the **bioactivity data** of the ***CDK1*** in ***Cervical Cancer (1 ENTRY)***

```
[ ]  selected_target = targets.target_chembl_id[3]
     selected_target
```

```
     'CHEMBL308'
```

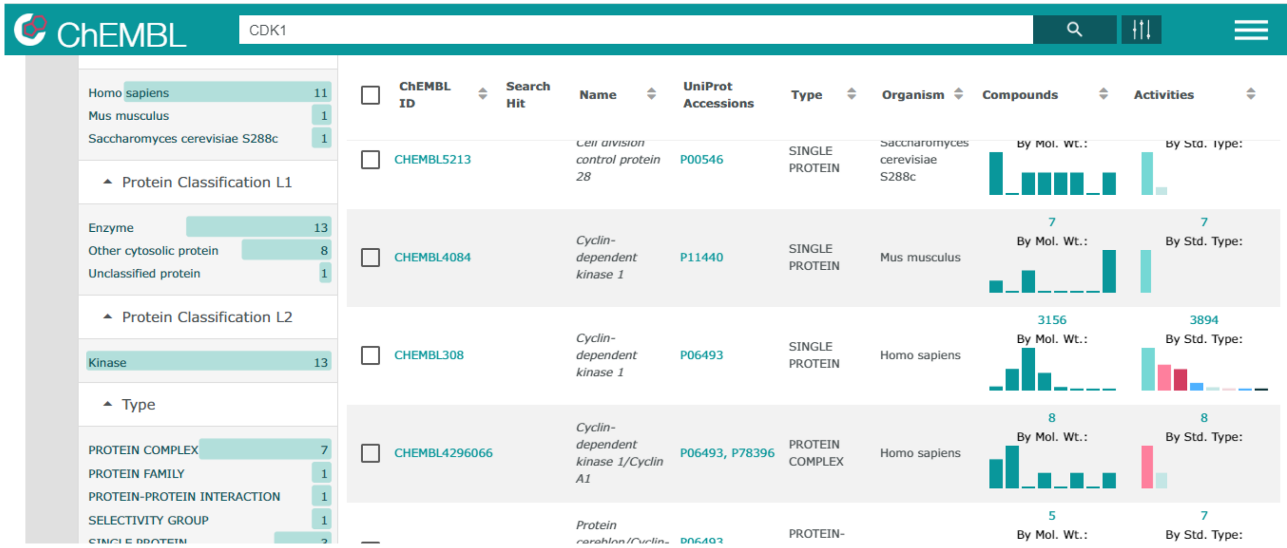
```
[ ]  # Searching for the specific chembl id in the above results
     chembl_id = targets.loc[targets['target_chembl_id'] == 'CHEMBL308']

     # Print the specific chembl id
     print(chembl_id)
```

```
                                   cross_references      organism  \
     3  [{'xref_id': 'Cyclin-dependent_kinase_1', 'xre...  Homo sapiens

                      pref_name   score  species_group_flag target_chembl_id  \
     3  Cyclin-dependent kinase 1   14.0               False        CHEMBL308

                                  target_components     target_type  tax_id
     3  [{'accession': 'P06493', 'component_descriptio...  SINGLE PROTEIN    9606
```

Here, we will retrieve only bioactivity data for AGT RECEPTOR (CHEMBL3596085) that are reported as IC 50 values in nM (nanomolar) unit.

filtering of the IC 50 values ⏳

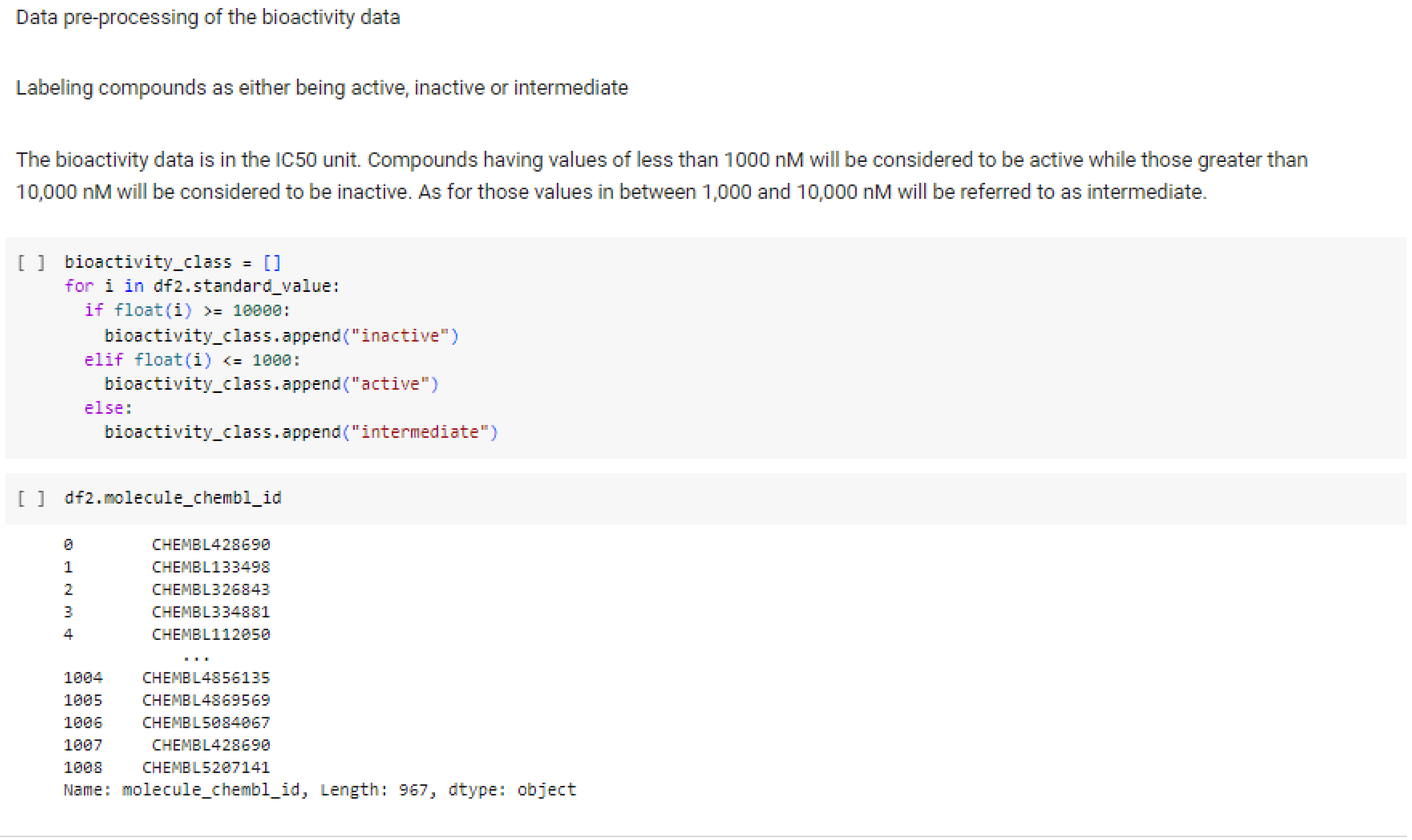
```
[ ]  activity = new_client.activity
     res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
```



Homo sapiens	11
Mus musculus	1
Saccharomyces cerevisiae S288c	1
Enzyme	13
Other cytosolic protein	8
Unclassified protein	1
Kinase	13
PROTEIN COMPLEX	7
PROTEIN FAMILY	1
PROTEIN-PROTEIN INTERACTION	1
SELECTIVITY GROUP	1
SINGLE PROTEIN	3

ChEMBL ID	Search Hit	Name	UniProt Accessions	Type	Organism	Compounds	Activities
CHEMBL5213		Cell division control protein 28	P00546	SINGLE PROTEIN	Saccharomyces cerevisiae S288c	By Mol. Wt.:	By Std. Type:
CHEMBL4084		Cyclin-dependent kinase 1	P11440	SINGLE PROTEIN	Mus musculus	7 By Mol. Wt.:	7 By Std. Type:
CHEMBL308		Cyclin-dependent kinase 1	P06493	SINGLE PROTEIN	Homo sapiens	3156 By Mol. Wt.:	3894 By Std. Type:
CHEMBL4296066		Cyclin-dependent kinase 1/Cyclin A1	P06493, P78396	PROTEIN COMPLEX	Homo sapiens	8 By Mol. Wt.:	8 By Std. Type:
		Protein cereblon/Cyclin-	P06493	PROTEIN-		5 By Mol. Wt.:	7 By Std. Type:

Filtering and selecting bioactivity data related to CDK1 in cervical cancer



Data pre-processing of the bioactivity data

Labeling compounds as either being active, inactive or intermediate

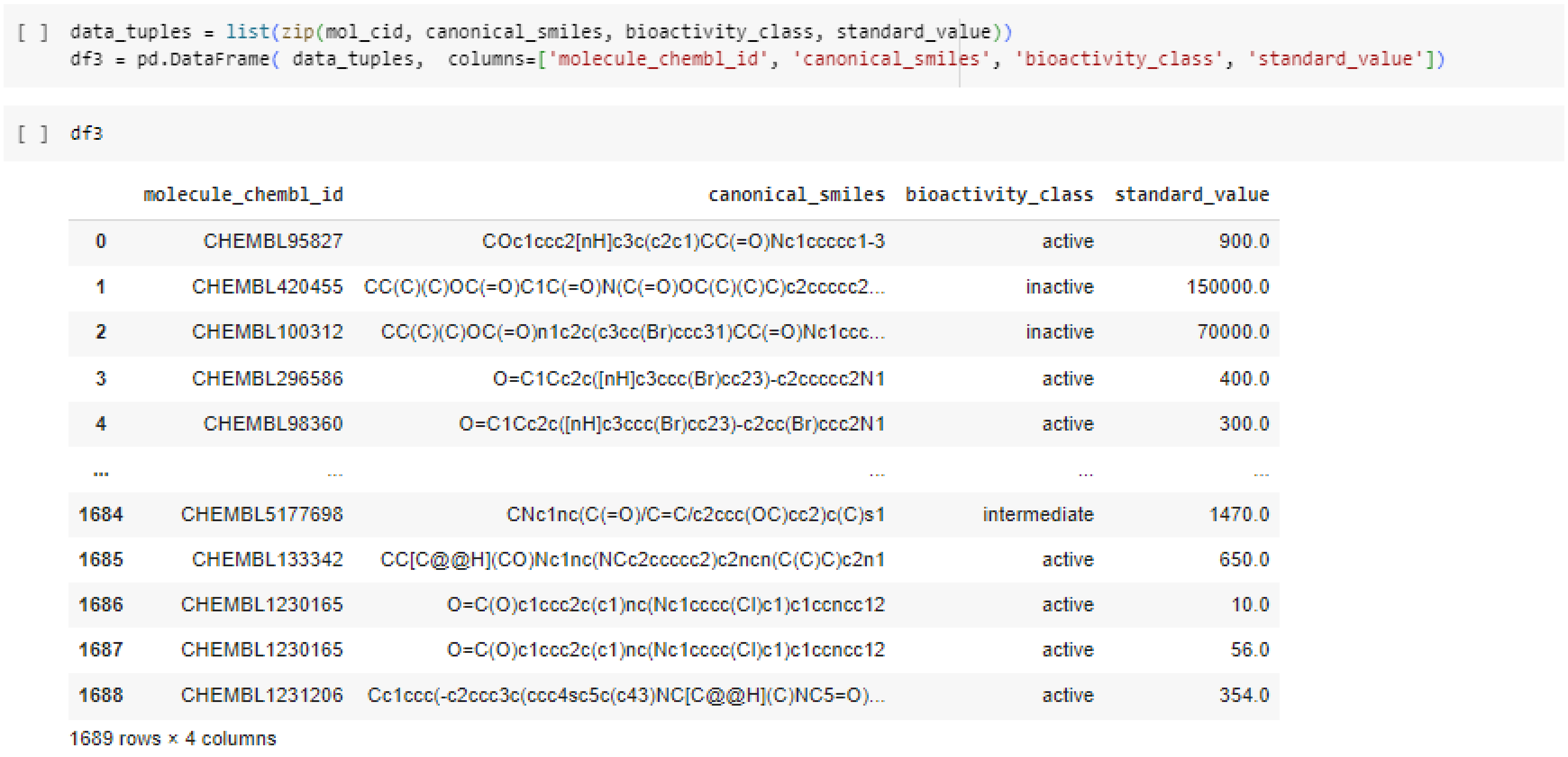
The bioactivity data is in the IC50 unit. Compounds having values of less than 1000 nM will be considered to be active while those greater than 10,000 nM will be considered to be inactive. As for those values in between 1,000 and 10,000 nM will be referred to as intermediate.

```
[ ]  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")
```

```
[ ]  df2.molecule_chembl_id
```

```
0        CHEMBL428690
1        CHEMBL133498
2        CHEMBL326843
3        CHEMBL334881
4        CHEMBL112050
             ...
1004    CHEMBL4856135
1005    CHEMBL4869569
1006    CHEMBL5084067
1007     CHEMBL428690
1008    CHEMBL5207141
Name: molecule_chembl_id, Length: 967, dtype: object
```

Data pre-processing : Labeling compounds as either being active, inactive or intermediate

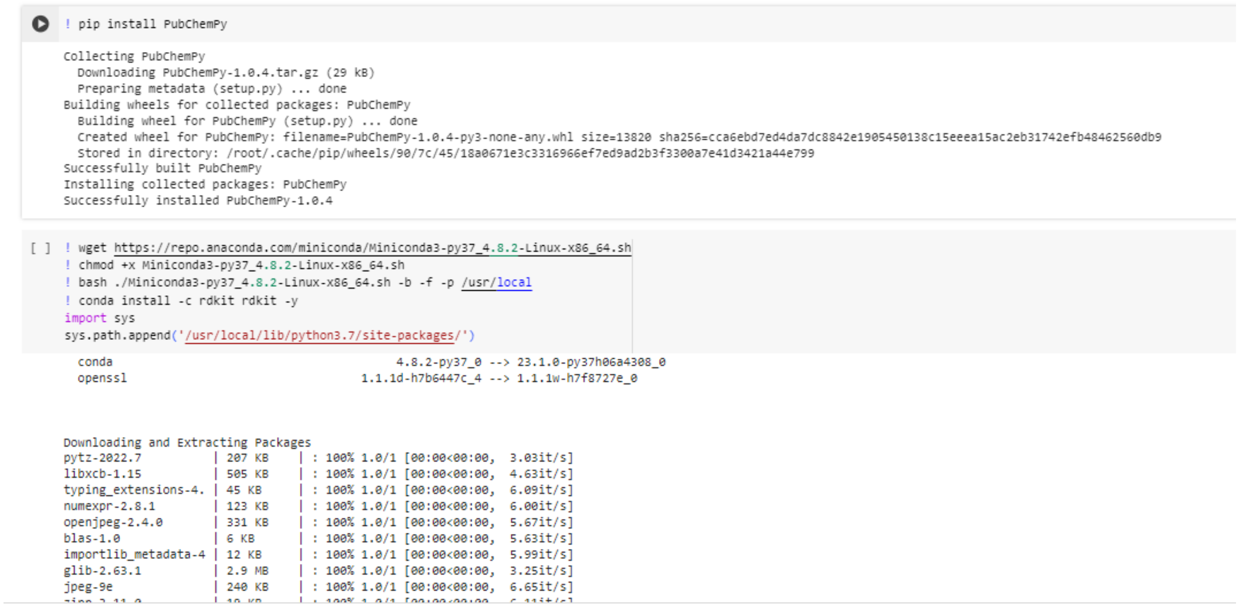


```
[ ] data_tuples = list(zip(mol_cid, canonical_smiles, bioactivity_class, standard_value))
    df3 = pd.DataFrame( data_tuples,  columns=['molecule_chembl_id', 'canonical_smiles', 'bioactivity_class', 'standard_value'])
```

```
[ ] df3
```

	molecule_chembl_id	canonical_smiles	bioactivity_class	standard_value
0	CHEMBL95827	COc1ccc2[nH]c3c(c2c1)CC(=O)Nc1ccccc1-3	active	900.0
1	CHEMBL420455	CC(C)(C)OC(=O)C1C(=O)N(C(=O)OC(C)(C)C)c2ccccc2...	inactive	150000.0
2	CHEMBL100312	CC(C)(C)OC(=O)n1c2c(c3cc(Br)ccc31)CC(=O)Nc1ccc...	inactive	70000.0
3	CHEMBL296586	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2ccccc2N1	active	400.0
4	CHEMBL98360	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2cc(Br)ccc2N1	active	300.0
...	...	...	...	...
1684	CHEMBL5177698	CNc1nc(C(=O)/C=C/c2ccc(OC)cc2)c(C)s1	intermediate	1470.0
1685	CHEMBL133342	CC[C@@H](CO)Nc1nc(NCc2ccccc2)c2ncn(C(C)C)c2n1	active	650.0
1686	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	active	10.0
1687	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	active	56.0
1688	CHEMBL1231206	Cc1ccc(-c2ccc3c(ccc4sc5c(c43)NC[C@@H](C)NC5=O)...	active	354.0

1689 rows × 4 columns

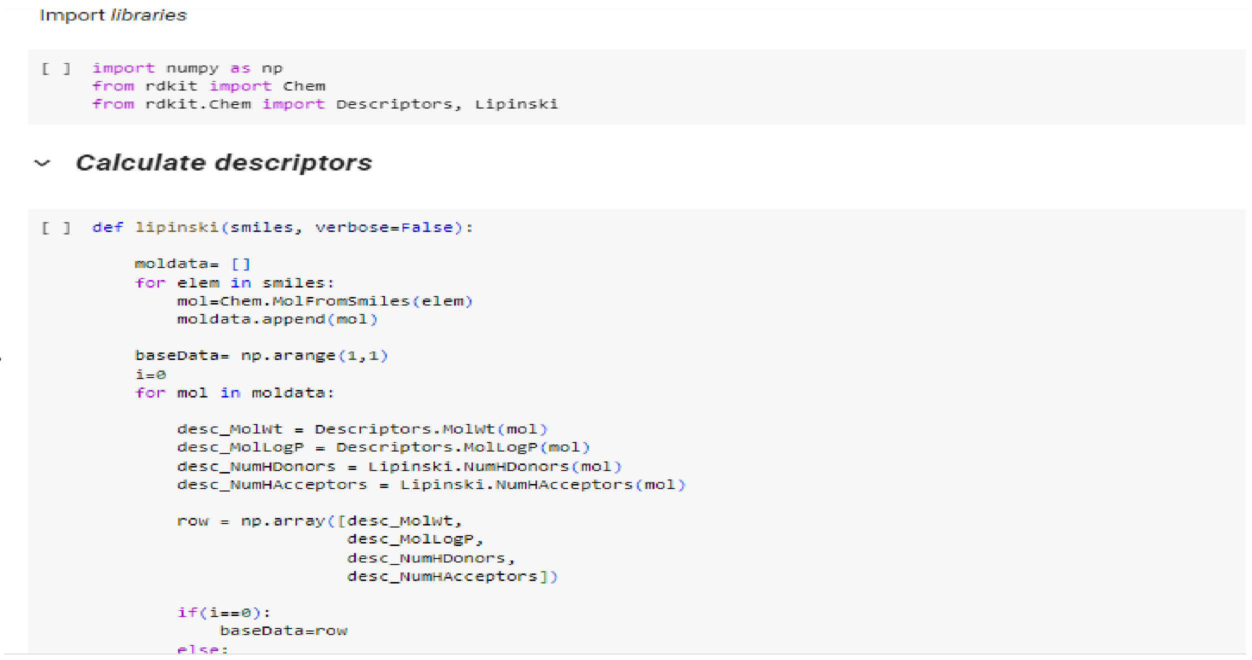


```
! pip install PubChemPy

Collecting PubChemPy
  Downloading PubChemPy-1.0.4.tar.gz (29 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: PubChemPy
  Building wheel for PubChemPy (setup.py) ... done
  Created wheel for PubChemPy: filename=PubChemPy-1.0.4-py3-none-any.whl size=13820 sha256=cca6ebd7ed4da7dc8842e1905450138c15eeea15ac2eb31742efb48462560db9
  Stored in directory: /root/.cache/pip/wheels/90/7c/45/18a0671e3c3316966ef7ed9ad2b3f3300a7e41d3421a44e799
Successfully built PubChemPy
Installing collected packages: PubChemPy
Successfully installed PubChemPy-1.0.4
```

```
[ ] ! wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.2-Linux-x86_64.sh
    ! chmod +x Miniconda3-py37_4.8.2-Linux-x86_64.sh
    ! bash ./Miniconda3-py37_4.8.2-Linux-x86_64.sh -b -f -p /usr/local
    ! conda install -c rdkit rdkit -y
    import sys
    sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

```
  conda                                  4.8.2-py37_0 --> 23.1.0-py37h06a4308_0
  openssl                             1.1.1d-h7b6447c_4 --> 1.1.1w-h7f8727e_0


Downloading and Extracting Packages
pytz-2022.7          | 207 KB    | : 100% 1.0/1 [00:00<00:00,  3.03it/s]
libxcb-1.15          | 505 KB    | : 100% 1.0/1 [00:00<00:00,  4.63it/s]
typing_extensions-4. | 45 KB     | : 100% 1.0/1 [00:00<00:00,  6.09it/s]
numexpr-2.8.1        | 123 KB    | : 100% 1.0/1 [00:00<00:00,  6.00it/s]
openjpeg-2.4.0       | 331 KB    | : 100% 1.0/1 [00:00<00:00,  5.67it/s]
blas-1.0             | 6 KB      | : 100% 1.0/1 [00:00<00:00,  5.63it/s]
importlib_metadata-4 | 12 KB     | : 100% 1.0/1 [00:00<00:00,  5.99it/s]
glib-2.63.1          | 2.9 MB    | : 100% 1.0/1 [00:00<00:00,  3.25it/s]
jpeg-9e              | 240 KB    | : 100% 1.0/1 [00:00<00:00,  6.65it/s]
```

Computing Lipinski descriptors for each compound in the dataset to characterize their drug-likeness properties



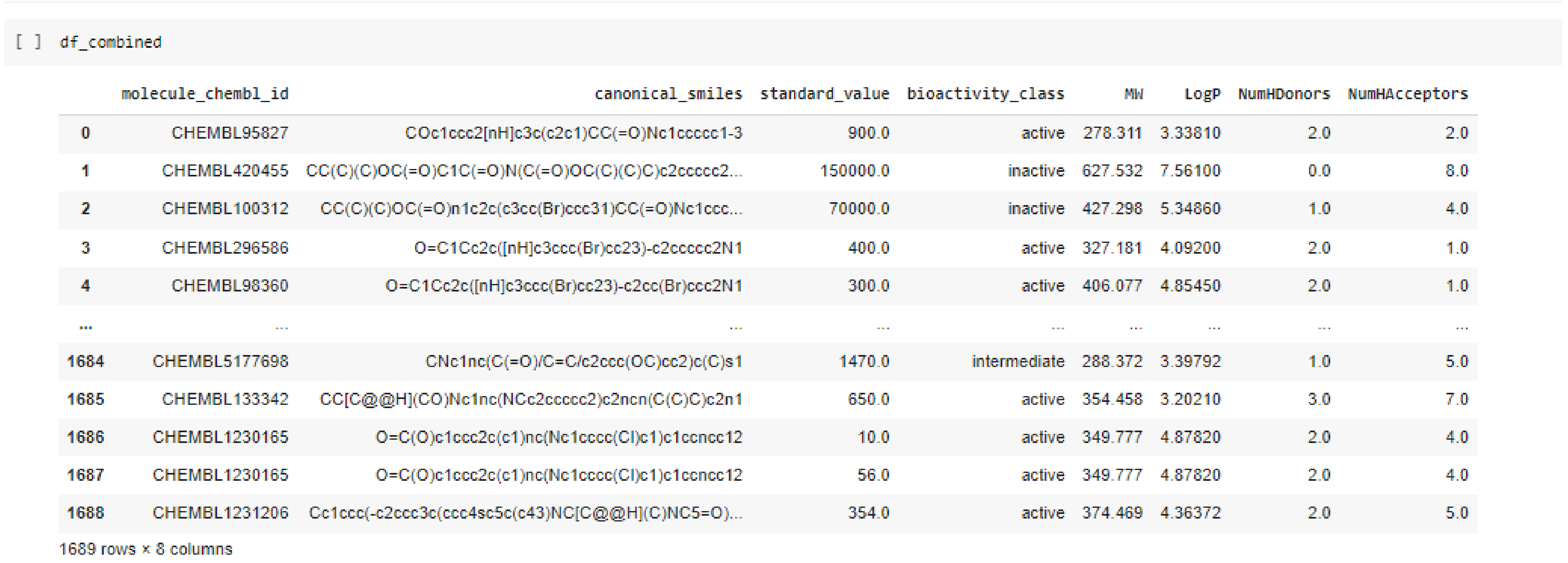
Import *libraries*

```
[ ] import numpy as np
    from rdkit import Chem
    from rdkit.Chem import Descriptors, Lipinski
```

Calculate descriptors

```
[ ] 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:
```

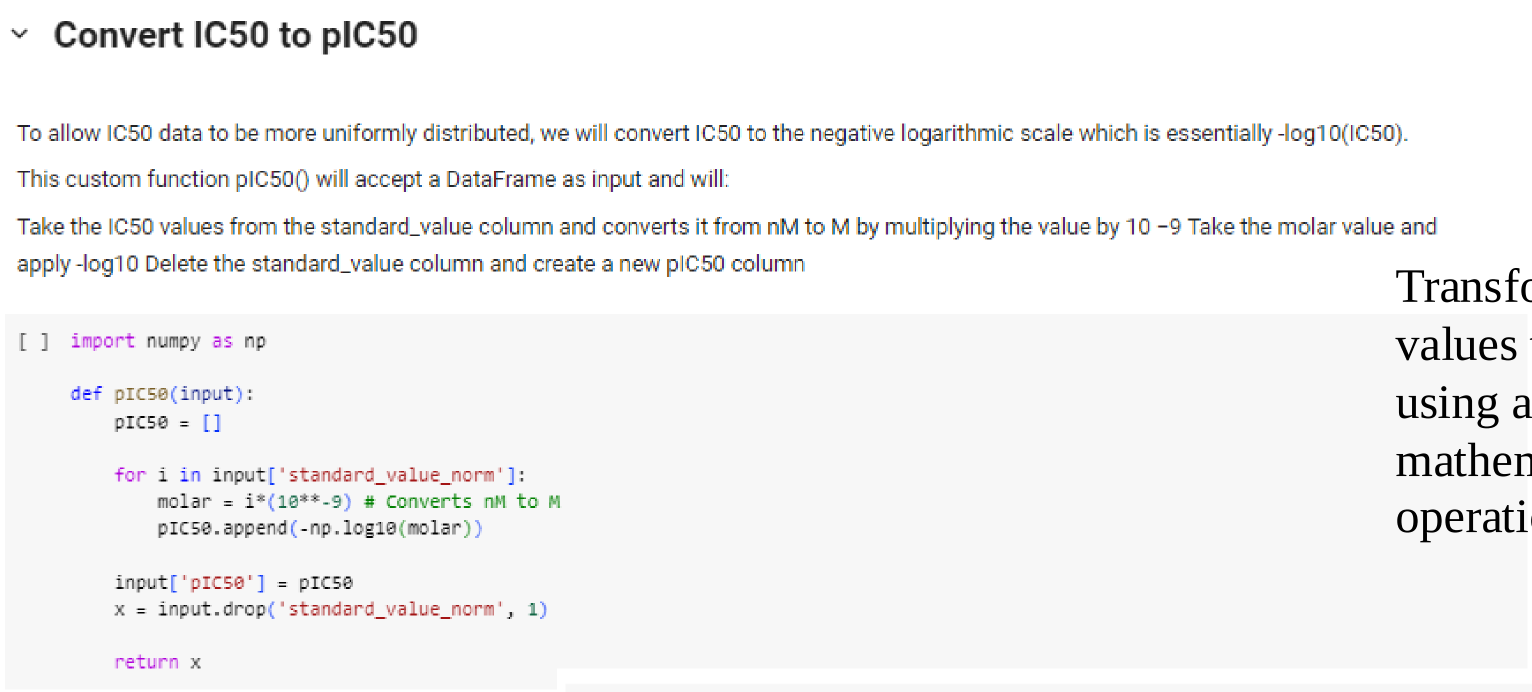
Calculate additional molecular descriptors to capture diverse chemical properties of the compounds.



```
[ ] df_combined
```

	molecule_chembl_id	canonical_smiles	standard_value	bioactivity_class	MW	LogP	NumHDonors	NumHAcceptors
0	CHEMBL95827	COc1ccc2[nH]c3c(c2c1)CC(=O)Nc1ccccc1-3	900.0	active	278.311	3.33810	2.0	2.0
1	CHEMBL420455	CC(C)(C)OC(=O)C1C(=O)N(C(=O)OC(C)(C)C)c2ccccc2...	150000.0	inactive	627.532	7.56100	0.0	8.0
2	CHEMBL100312	CC(C)(C)OC(=O)n1c2c(c3cc(Br)ccc31)CC(=O)Nc1ccc...	70000.0	inactive	427.298	5.34860	1.0	4.0
3	CHEMBL296586	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2ccccc2N1	400.0	active	327.181	4.09200	2.0	1.0
4	CHEMBL98360	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2cc(Br)ccc2N1	300.0	active	406.077	4.85450	2.0	1.0
...	...	...	...	...	...	...	...	...
1684	CHEMBL5177698	CNc1nc(C(=O)/C=C/c2ccc(OC)cc2)c(C)s1	1470.0	intermediate	288.372	3.39792	1.0	5.0
1685	CHEMBL133342	CC[C@@H](CO)Nc1nc(NCc2ccccc2)c2ncn(C(C)C)c2n1	650.0	active	354.458	3.20210	3.0	7.0
1686	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	10.0	active	349.777	4.87820	2.0	4.0
1687	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	56.0	active	349.777	4.87820	2.0	4.0
1688	CHEMBL1231206	Cc1ccc(-c2ccc3c(ccc4sc5c(c43)NC[C@@H](C)NC5=O)...	354.0	active	374.469	4.36372	2.0	5.0

1689 rows × 8 columns



Convert IC50 to pIC50

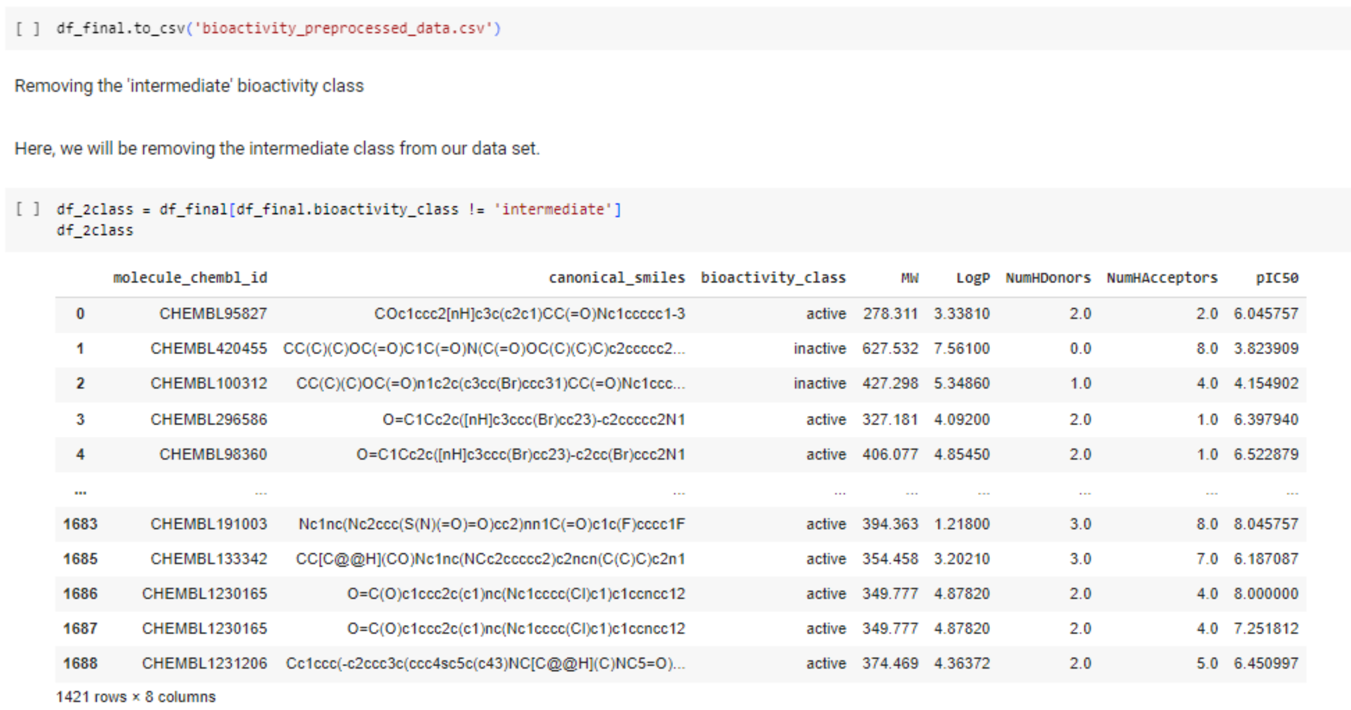
To allow IC50 data to be more uniformly distributed, we will convert IC50 to the negative logarithmic scale which is essentially -log10(IC50).

This custom function pIC50() will accept a DataFrame as input and will:

Take the IC50 values from the standard_value column and converts it from nM to M by multiplying the value by 10 −9 Take the molar value and apply -log10 Delete the standard_value column and create a new pIC50 column

```
[ ]  import numpy as np

     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
```

Transforming IC50 values to pIC50 values using appropriate mathematical operations.



```
[ ]  df_final.to_csv('bioactivity_preprocessed_data.csv')
```

Removing the 'intermediate' bioactivity class

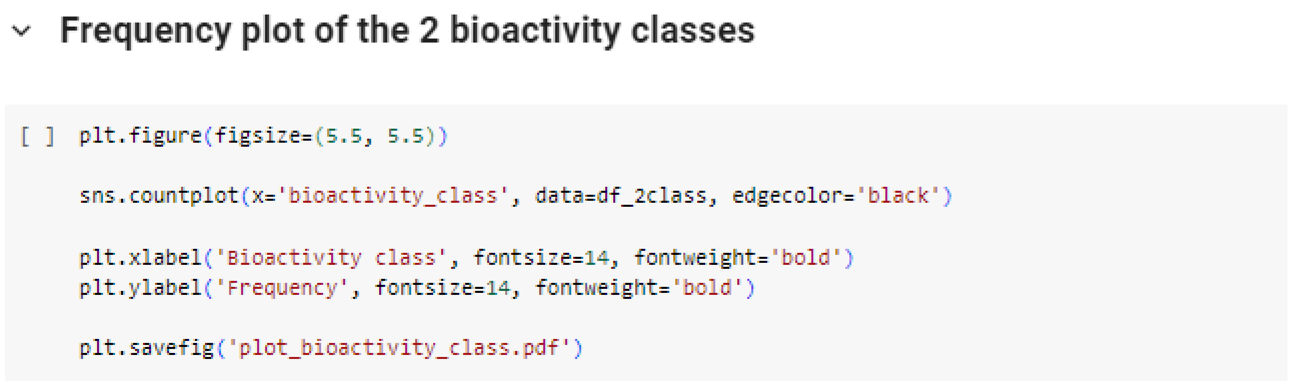
Here, we will be removing the intermediate class from our data set.

```
[ ]  df_2class = df_final[df_final.bioactivity_class != 'intermediate']
     df_2class
```

	molecule_chembl_id	canonical_smiles	bioactivity_class	MW	LogP	NumHDonors	NumHAcceptors	pIC50
0	CHEMBL95827	COc1ccc2[nH]c3c(c2c1)CC(=O)Nc1ccccc1-3	active	278.311	3.33810	2.0	2.0	6.045757
1	CHEMBL420455	CC(C)(C)OC(=O)C1C(=O)N(C(=O)OC(C)(C)C)c2ccccc2...	inactive	627.532	7.56100	0.0	8.0	3.823909
2	CHEMBL100312	CC(C)(C)OC(=O)n1c2c(c3cc(Br)ccc31)CC(=O)Nc1ccc...	inactive	427.298	5.34860	1.0	4.0	4.154902
3	CHEMBL296586	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2ccccc2N1	active	327.181	4.09200	2.0	1.0	6.397940
4	CHEMBL98360	O=C1Cc2c([nH]c3ccc(Br)cc23)-c2cc(Br)ccc2N1	active	406.077	4.85450	2.0	1.0	6.522879
...	...	...	...	...	...	...	...	...
1683	CHEMBL191003	Nc1nc(Nc2ccc(S(N)(=O)=O)cc2)nn1C(=O)c1c(F)cccc1F	active	394.363	1.21800	3.0	8.0	8.045757
1685	CHEMBL133342	CC[C@@H](CO)Nc1nc(NCc2ccccc2)c2ncn(C(C)C)c2n1	active	354.458	3.20210	3.0	7.0	6.187087
1686	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	active	349.777	4.87820	2.0	4.0	8.000000
1687	CHEMBL1230165	O=C(O)c1ccc2c(c1)nc(Nc1cccc(Cl)c1)c1ccncc12	active	349.777	4.87820	2.0	4.0	7.251812
1688	CHEMBL1231206	Cc1ccc(-c2ccc3c(ccc4sc5c(c43)NC[C@@H](C)NC5=O)...	active	374.469	4.36372	2.0	5.0	6.450997

1421 rows × 8 columns

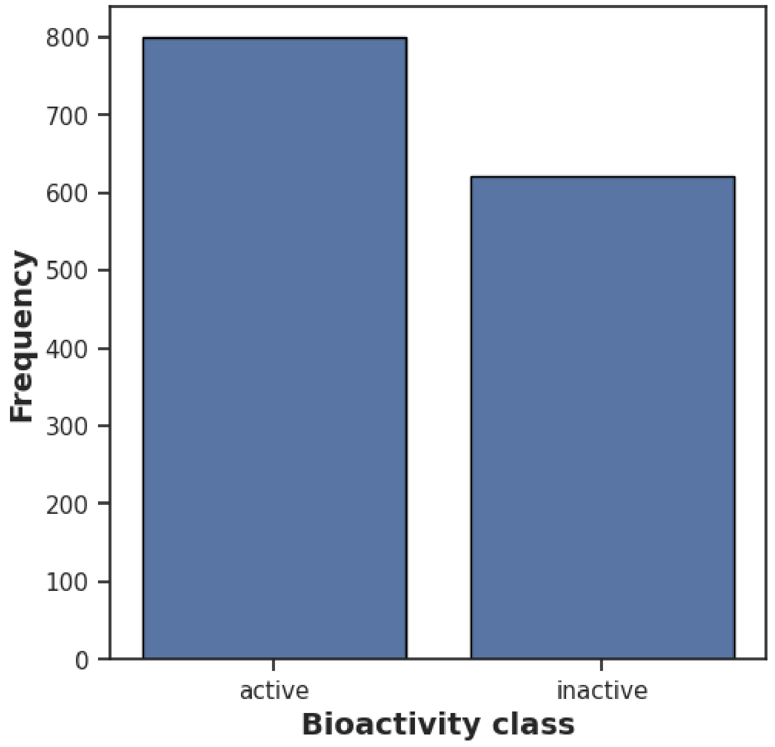
Exclude compounds categorized under the 'intermediate' bioactivity class to focus on potent compounds.

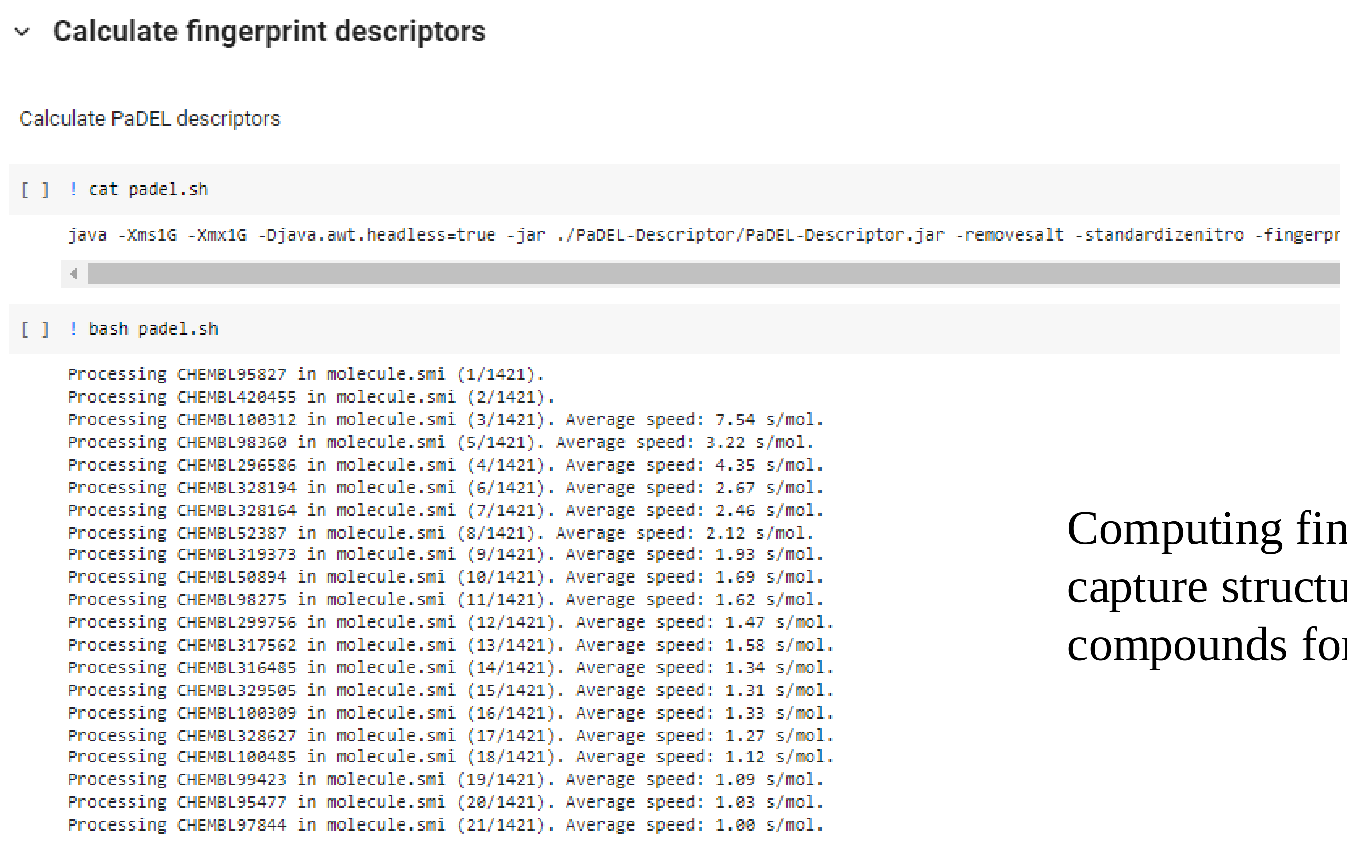


Frequency plot of the 2 bioactivity classes

```
[ ]  plt.figure(figsize=(5.5, 5.5))

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

     plt.xlabel('Bioactivity class', fontsize=14, fontweight='bold')
     plt.ylabel('Frequency', fontsize=14, fontweight='bold')

     plt.savefig('plot_bioactivity_class.pdf')
```

Created a frequency plot to visualize the distribution of compounds across different bioactivity classes.





Calculate fingerprint descriptors

Calculate PaDEL descriptors

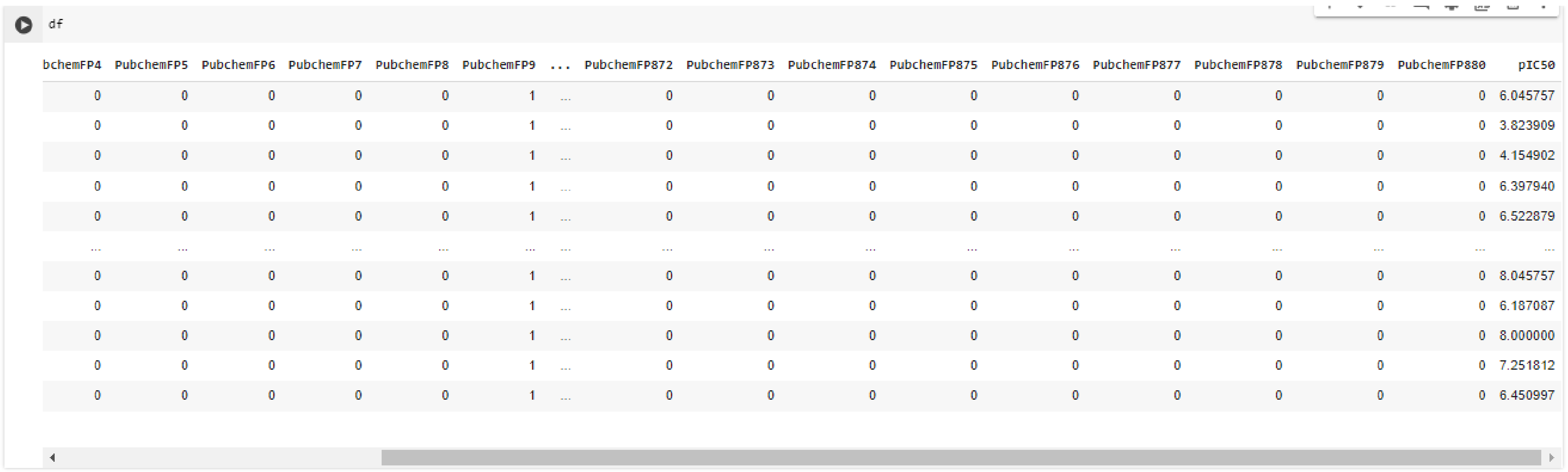
```
[ ] ! cat padel.sh
```

```
java -Xms1G -Xmx1G -Djava.awt.headless=true -jar ./PaDEL-Descriptor/PaDEL-Descriptor.jar -removesalt -standardizenitro -fingerpr
```

```
[ ] ! bash padel.sh
```

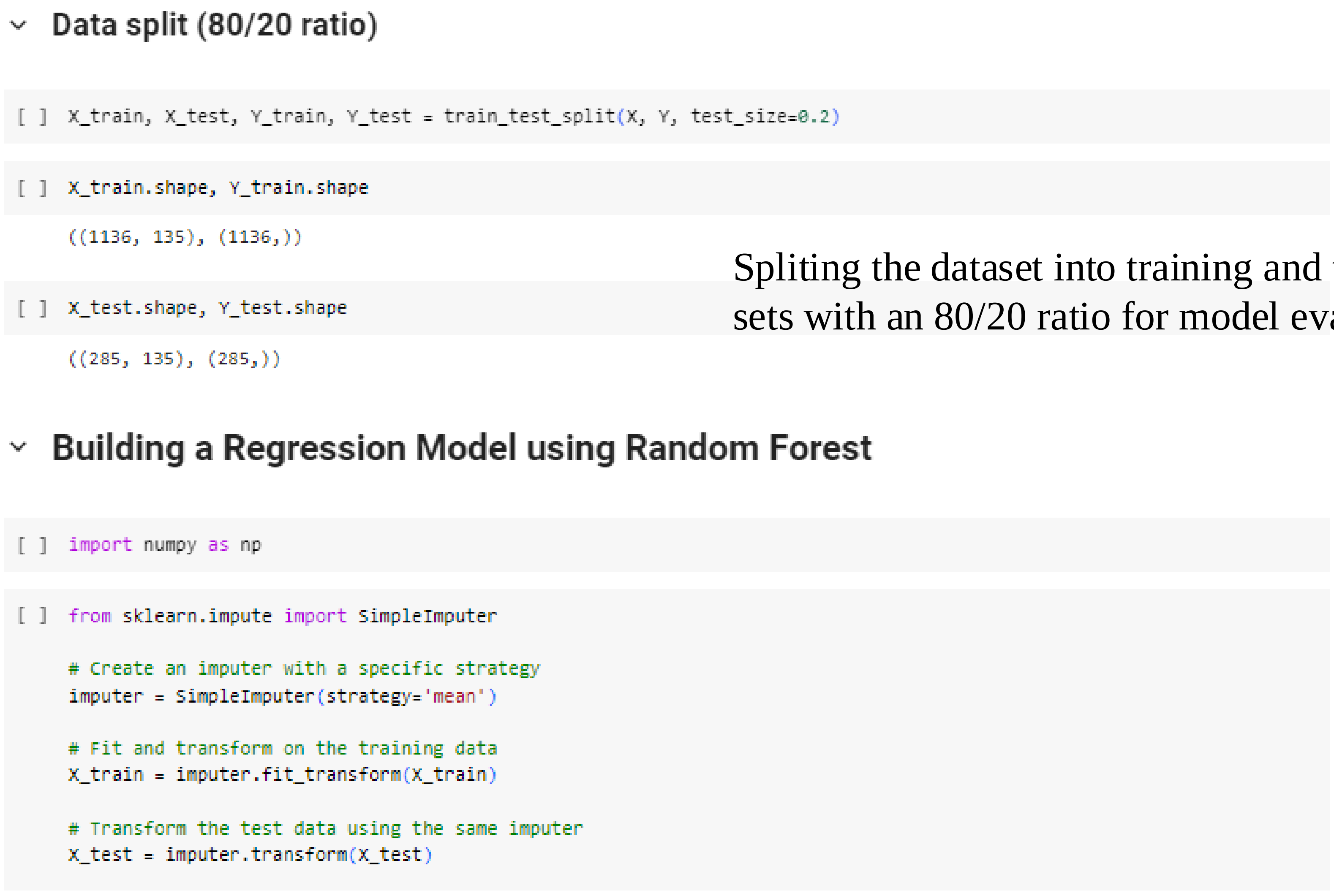
```
Processing CHEMBL95827 in molecule.smi (1/1421).
Processing CHEMBL420455 in molecule.smi (2/1421).
Processing CHEMBL100312 in molecule.smi (3/1421). Average speed: 7.54 s/mol.
Processing CHEMBL98360 in molecule.smi (5/1421). Average speed: 3.22 s/mol.
Processing CHEMBL296586 in molecule.smi (4/1421). Average speed: 4.35 s/mol.
Processing CHEMBL328194 in molecule.smi (6/1421). Average speed: 2.67 s/mol.
Processing CHEMBL328164 in molecule.smi (7/1421). Average speed: 2.46 s/mol.
Processing CHEMBL52387 in molecule.smi (8/1421). Average speed: 2.12 s/mol.
Processing CHEMBL319373 in molecule.smi (9/1421). Average speed: 1.93 s/mol.
Processing CHEMBL50894 in molecule.smi (10/1421). Average speed: 1.69 s/mol.
Processing CHEMBL98275 in molecule.smi (11/1421). Average speed: 1.62 s/mol.
Processing CHEMBL299756 in molecule.smi (12/1421). Average speed: 1.47 s/mol.
Processing CHEMBL317562 in molecule.smi (13/1421). Average speed: 1.58 s/mol.
Processing CHEMBL316485 in molecule.smi (14/1421). Average speed: 1.34 s/mol.
Processing CHEMBL329505 in molecule.smi (15/1421). Average speed: 1.31 s/mol.
Processing CHEMBL100309 in molecule.smi (16/1421). Average speed: 1.33 s/mol.
Processing CHEMBL328627 in molecule.smi (17/1421). Average speed: 1.27 s/mol.
Processing CHEMBL100485 in molecule.smi (18/1421). Average speed: 1.12 s/mol.
Processing CHEMBL99423 in molecule.smi (19/1421). Average speed: 1.09 s/mol.
Processing CHEMBL95477 in molecule.smi (20/1421). Average speed: 1.03 s/mol.
Processing CHEMBL97844 in molecule.smi (21/1421). Average speed: 1.00 s/mol.
```

Computing fingerprint descriptors to capture structural features of compounds for further analysis.



```
df
```

bchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	PubchemFP9	...	PubchemFP872	PubchemFP873	PubchemFP874	PubchemFP875	PubchemFP876	PubchemFP877	PubchemFP878	PubchemFP879	PubchemFP880	pIC50
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	6.045757
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	3.823909
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	4.154902
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	6.397940
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	6.522879
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	8.045757
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	6.187087
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	8.000000
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	7.251812
0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	6.450997



Data split (80/20 ratio)

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

```
[ ] X_train.shape, Y_train.shape
```

```
((1136, 135), (1136,))
```

```
[ ] X_test.shape, Y_test.shape
```

```
((285, 135), (285,))
```

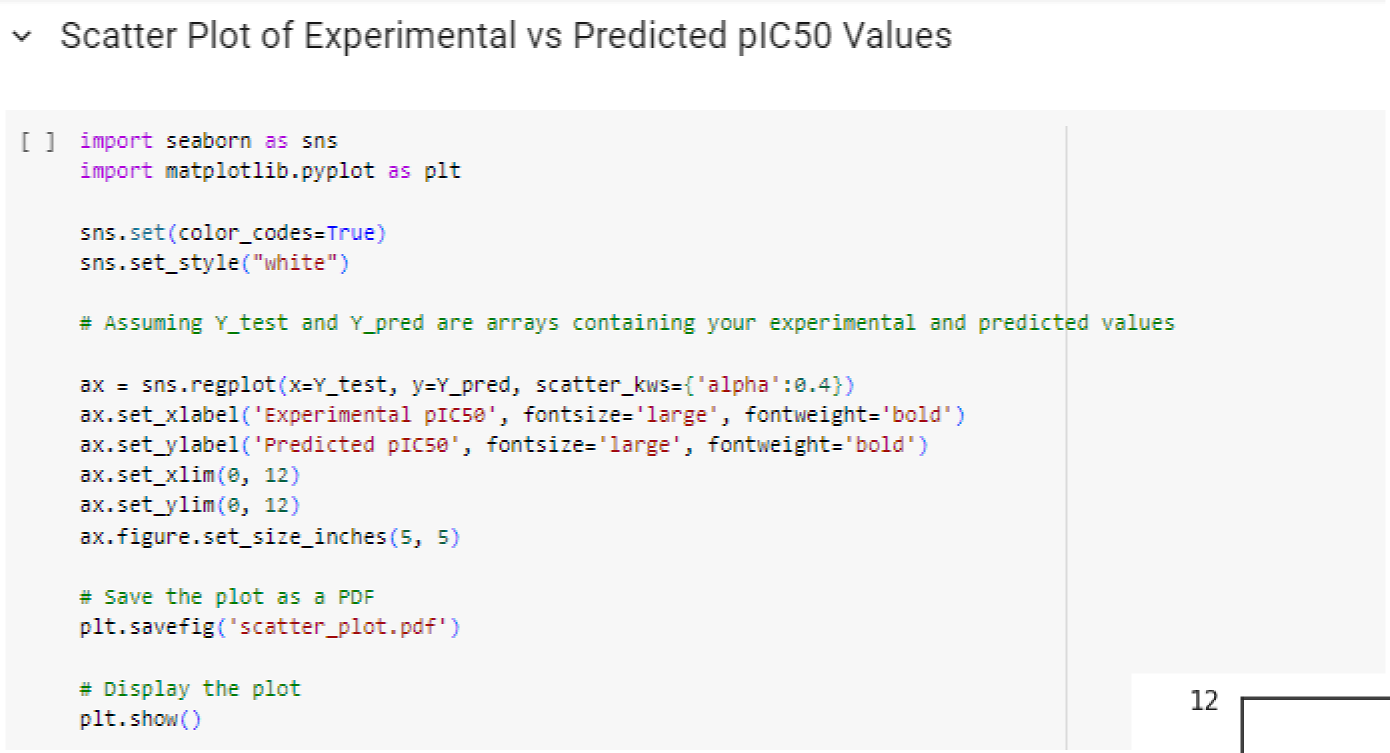
Spliting the dataset into training and testing sets with an 80/20 ratio for model evaluation.

Building a Regression Model using Random Forest

```
[ ] import numpy as np
```

```
[ ] from sklearn.impute import SimpleImputer

    # Create an imputer with a specific strategy
    imputer = SimpleImputer(strategy='mean')

    # Fit and transform on the training data
    X_train = imputer.fit_transform(X_train)

    # Transform the test data using the same imputer
    X_test = imputer.transform(X_test)
```

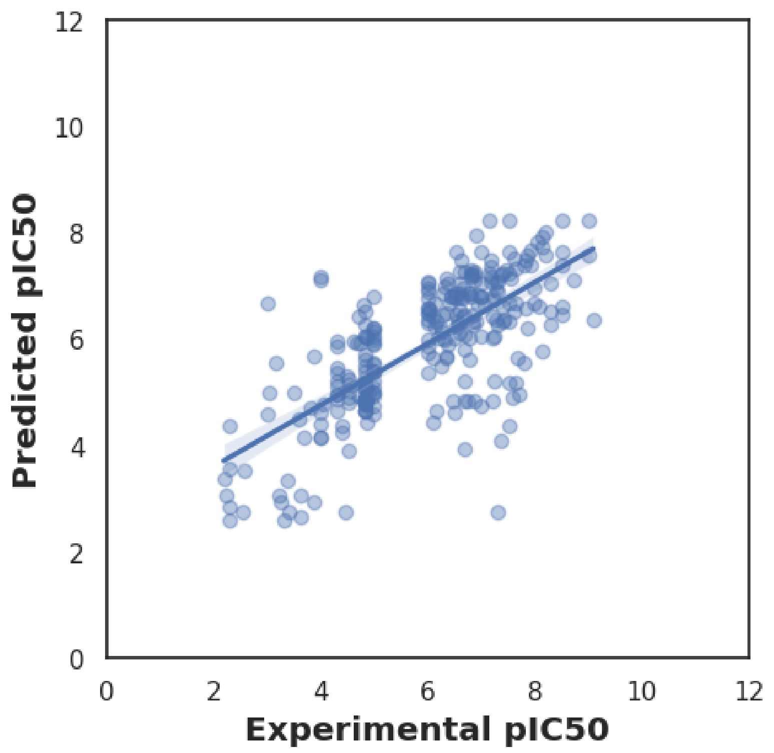
Train a Random Forest regression model using the training data to predict compound potency



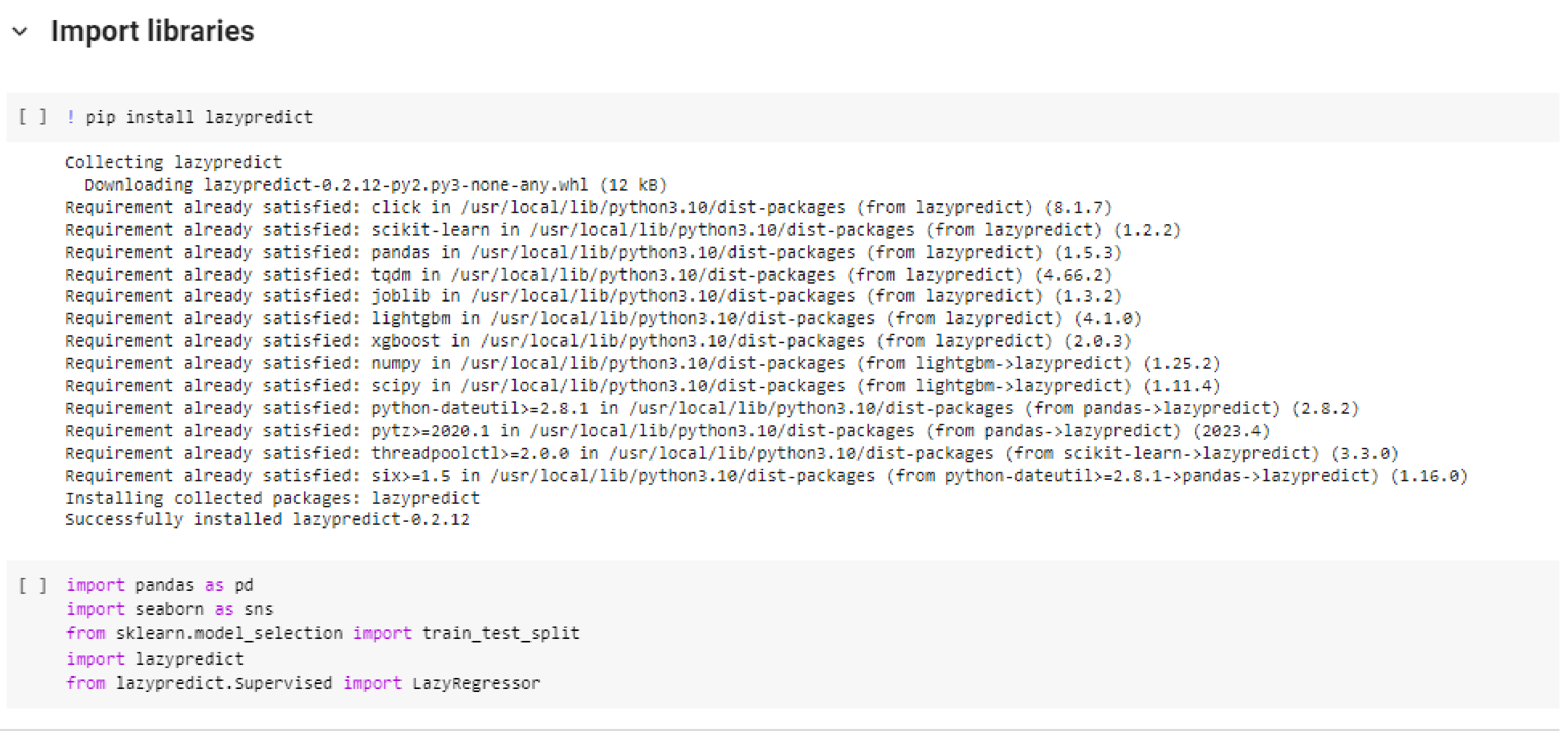
Scatter Plot of Experimental vs Predicted pIC50 Values

```
[ ] import seaborn as sns
    import matplotlib.pyplot as plt

    sns.set(color_codes=True)
    sns.set_style("white")

    # Assuming Y_test and Y_pred are arrays containing your experimental and predicted values

    ax = sns.regplot(x=Y_test, y=Y_pred, scatter_kws={'alpha':0.4})
    ax.set_xlabel('Experimental pIC50', fontsize='large', fontweight='bold')
    ax.set_ylabel('Predicted pIC50', fontsize='large', fontweight='bold')
    ax.set_xlim(0, 12)
    ax.set_ylim(0, 12)
    ax.figure.set_size_inches(5, 5)

    # Save the plot as a PDF
    plt.savefig('scatter_plot.pdf')

    # Display the plot
    plt.show()
```

Visualizing the model's performance by comparing predicted pIC50 values with experimental values.



The scatter plot of the predicted values against the actual values forms a diagonal shape, it indicates that the model's predictions perfectly align with the actual values. In the context of random forest regression, this alignment suggests that the model has accurately captured the underlying relationships in the data and is making precise predictions.



Import libraries

```
[ ] ! pip install lazypredict
```

```
Collecting lazypredict
  Downloading lazypredict-0.2.12-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from lazypredict) (8.1.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.5.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.66.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.3.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.1.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.11.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->lazypredict) (3.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->lazypredict) (1.16.0)
Installing collected packages: lazypredict
Successfully installed lazypredict-0.2.12
```

```
[ ] import pandas as pd
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    import lazypredict
    from lazypredict.Supervised import LazyRegressor
```

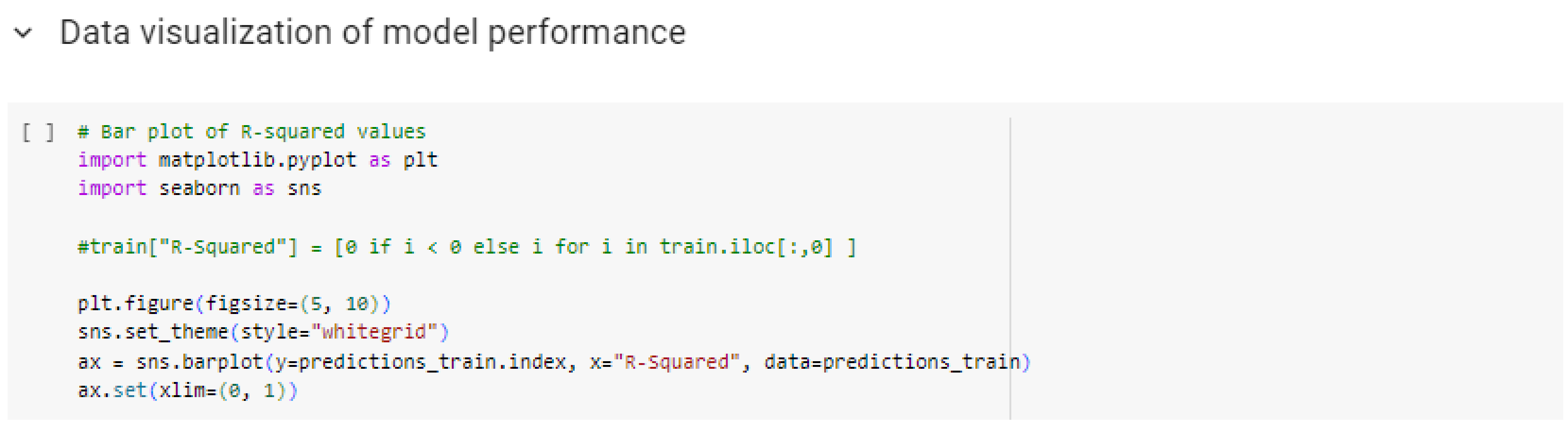
The Lazy Predict package is a Python library that streamlines the process of evaluating multiple machine learning algorithms by providing a comprehensive overview of their performance.

Perform data splitting using 80/20 ratio

```
[ ]  X_train, X_test1, Y_train, Y_test = train_test_split(X, Y, test_size=0.5, random_state=42)
```

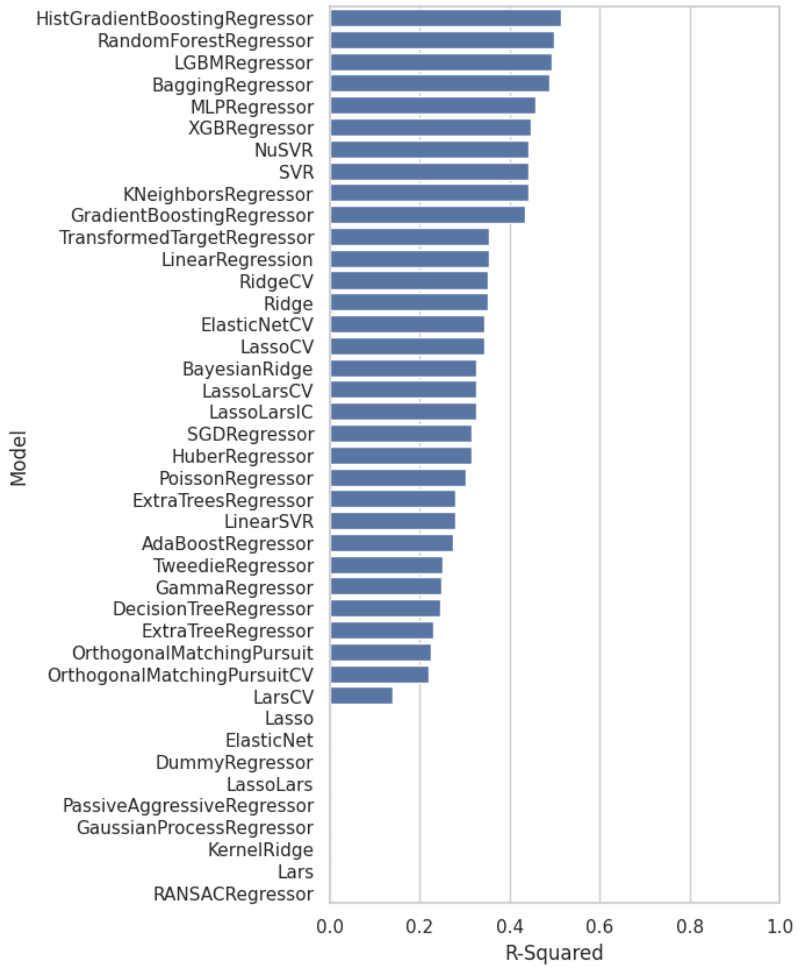
Compare ML algorithms

```
[ ]  from lazypredict.Supervised import LazyRegressor
     from sklearn.model_selection import train_test_split

     # Assuming you have imported necessary libraries and data, and split it into training and testing sets
     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

     # Define and build the lazyclassifier
     clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)

     # Fit the model
     models_train, predictions_train = clf.fit(X_train, X_test, Y_train, Y_test)

     # Print the models and their corresponding performance metrics
     print(models_train)

     # Access the predictions of individual models
     print(predictions_train)
```



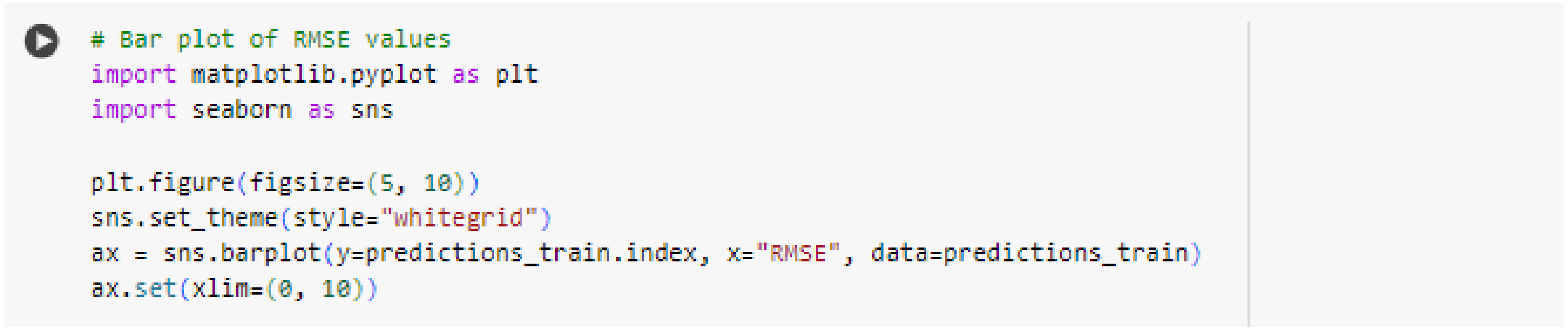
Data visualization of model performance

```
[ ] # Bar plot of R-squared values
    import matplotlib.pyplot as plt
    import seaborn as sns

    #train["R-Squared"] = [0 if i < 0 else i for i in train.iloc[:,0] ]

    plt.figure(figsize=(5, 10))
    sns.set_theme(style="whitegrid")
    ax = sns.barplot(y=predictions_train.index, x="R-Squared", data=predictions_train)
    ax.set(xlim=(0, 1))
```



•Comparing the R-squared values of different models to assess their predictive accuracy

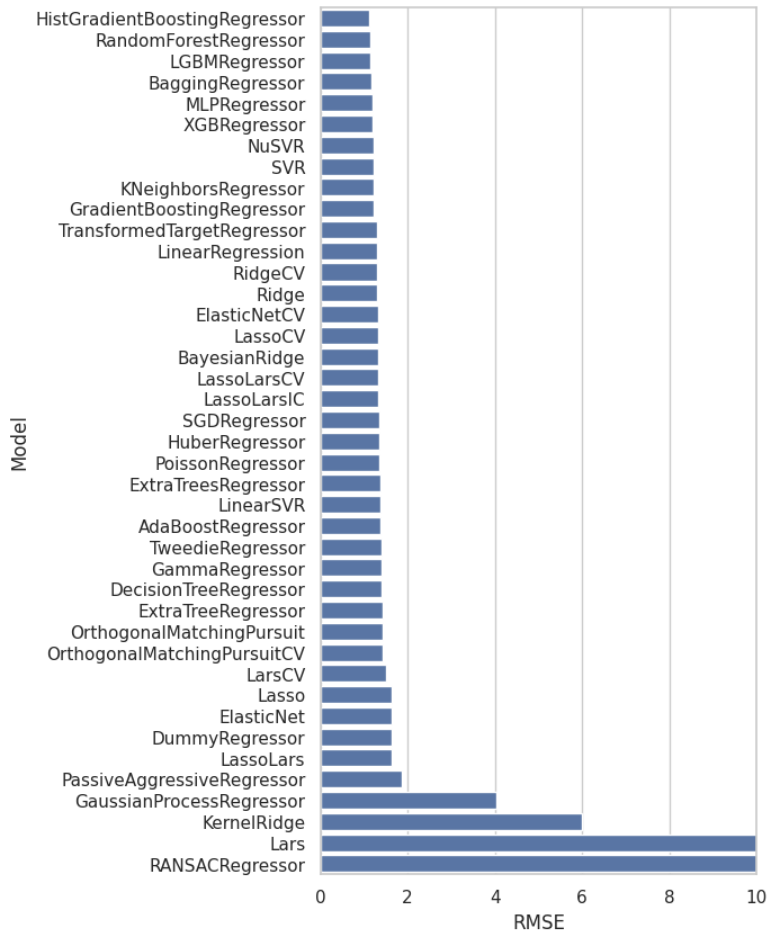
•The greater the R-squared number, the better the model's ability to explain data variation.



```
# Bar plot of RMSE values
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(5, 10))
sns.set_theme(style="whitegrid")
ax = sns.barplot(y=predictions_train.index, x="RMSE", data=predictions_train)
ax.set(xlim=(0, 10))
```

•Compare the Root Mean Square Error (RMSE) values of various models to evaluate their error rates.

•A lower RMSE generally indicates a better-fitting model



Score Table

Model	Adjusted R-Squared value	R-Squared Value	RMSE	Time Taken
HistGradientBoostingRegressor	0.07	0.51	1.13	0.88
RandomforestRegressor	0.05	0.50	1.15	1.24
LGBMRegressor	0.04	0.49	1.15	0.20
BaggingRegressor	0.02	0.49	1.16	0.15
MLPRegressor	-0.04	0.46	1.20	3.85
XGBRegressor	-0.05	0.45	1.21	0.24
NuSVR	-0.06	0.44	1.21	0.21
SVR	-0.06	0.44	1.21	0.25
KNeighborsRegressor	-0.06	0.44	1.21	0.11
GradientBoostingRegressor	-0.08	0.43	1.22	0.53
TransformedTargetRegressor	-0.23	0.35	1.31	0.04
LinerRegression	-0.23	0.35	1.31	0.07
RidgeCV	-0.24	0.35	1.31	0.05
Ridge	-0.24	0.35	1.31	0.02
ElasticNetCV	-0.25	0.35	1.31	5.06
LassoCV	-0.25	0.34	1.32	3.69
BayesianRidge	-0.29	0.33	1.33	0.09
LassoLarsCV	-0.29	0.33	1.33	0.26
LassoLarsIC	-0.29	0.33	1.33	0.10
SGDRegressor	-0.30	0.32	1.34	0.04
HuberRegressor	-0.30	0.32	1.34	0.12
PoissonRegressor	-0.33	0.30	1.35	0.43
ExtraTreesRegressor	-0.37	0.28	1.38	1.50
LinerSVR	-0.38	0.28	1.38	0.30
AdaBoostRegressor	-0.38	0.27	1.38	0.18
TweedieRegressor	-0.42	0.25	1.40	0.08
GammRegressor	-0.43	0.25	1.41	0.14
DecisionTreeRegressor	-0.44	0.25	1.41	0.10
ExtraTreeRegressor	-0.47	0.23	1.42	0.09
OrthogonalMatchingPursuit	-0.47	0.23	1.43	0.03
OrthogonalMatchingPursuitCV	-0.48	0.22	1.43	0.05
LarsCV	-0.64	0.14	1.50	0.28
Lasso	-0.91	-0.00	1.62	0.04
ElasticNet	-0.91	-0.00	1.62	0.02
DummyRegressor	-0.91	-0.00	1.62	0.03
LassoLars	-0.91	-0.00	1.62	0.05
PassiveAggressiveRegressor	-1.51	-0.32	1.86	0.04
GaussianProcessRegressor	-10.80	-5.91	4.04	0.41
KernelRidge	-25.11	-12.70	6.01	0.13
Lars	-1121.72	-588.03	39.40	0.11
RANSACRegressor	-579978062925424322674688.00	-304284265408057151848448.00	895576856490.44	1.22

CONCLUSION

•This study analyzed bioactive compounds linked to CDK1, initially identifying IC50 values and transforming them into PIC50 values for interpretation.

•Using these values as benchmarks, a predictive machine learning model was developed to forecast PIC50 values based on CDK1 bioactivity. Various machine learning algorithms were employed for model training and testing, assessing accuracy through scatter plots.

•Transitioning the model to a web-based application aims to facilitate accessibility, offering user-friendly interfaces for drug discovery efforts targeting CDK1.

•This application could include features like data validation and visualization tools for comprehensive analysis.

THANK YOU