In [3]:
```python
import pandas as pd
from chembl_webresource_client.new_client import new_client
```

In [5]:
```python
target = new_client.target
target_query = target.search('CHEMBL233')
targets = pd.DataFrame.from_dict(target_query)
targets
```

Out[5]:

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Homo sapiens	Mu opioid receptor	12.0	False	CHEMBL233	[{'accession': 'P35372', 'component_descriptio...	SINGLE PROTEIN	9606
1	[]	Homo sapiens	CCR5/mu opioid receptor complex	8.0	False	CHEMBL3301384	[{'accession': 'P51681', 'component_descriptio...	PROTEIN COMPLEX	9606
2	[]	Homo sapiens	Opioid receptors; mu & delta	7.0	False	CHEMBL2095149	[{'accession': 'P41143', 'component_descriptio...	SELECTIVITY GROUP	9606
3	[]	Homo sapiens	Opioid receptors; mu & kappa	7.0	False	CHEMBL2095156	[{'accession': 'P41145', 'component_descriptio...	SELECTIVITY GROUP	9606
4	[]	Homo sapiens	Mu opioid receptor/Alpha-2A adrenergic receptor	7.0	False	CHEMBL3883321	[{'accession': 'P08913', 'component_descriptio...	PROTEIN COMPLEX	9606
5	[]	Homo sapiens	Cannabinoid receptor 1/Mu-type opioid receptor	7.0	False	CHEMBL3885538	[{'accession': 'P21554', 'component_descriptio...	PROTEIN COMPLEX	9606
6	[]	Homo sapiens	Opioid receptors; mu/kappa/delta	6.0	False	CHEMBL2095181	[{'accession': 'P41145', 'component_descriptio...	PROTEIN FAMILY	9606

In [8]:
```python
selected_target = targets.target_chembl_id[0]
selected_target
```

Out[8]:
```
'CHEMBL233'
```

```
In [12]: activity = new_client.activity
         res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
```

```
In [15]: df = pd.DataFrame.from_dict(res)
```

```
In [18]: print(len(res))
```

```
2452
```

```
In [37]: df.head()
```

Out[37]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_
0	None	None	32544	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None	
1	None	None	33756	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None	
2	None	None	38634	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None	
3	None	None	38643	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None	
4	None	None	39754	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None	

5 rows × 46 columns

```
In [33]: df.tail()
```

Out[33]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession
2447	{'action_type': 'ANTAGONIST', 'description': '...	None	25631924	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2448	{'action_type': 'ANTAGONIST', 'description': '...	None	25631925	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2449	{'action_type': 'ANTAGONIST', 'description': '...	None	25631926	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2450	None	None	25778419	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465602	Selectivity interaction (BET inhibitor selecti...	B	None
2451	None	None	25787567	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474455	Selectivity interaction (Opioid Mu GTPgS agoni...	B	None

5 rows × 46 columns

In [35]:
```
df.info()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2452 entries, 0 to 2451
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   action_type                 66 non-null     object 
 1   activity_comment            1181 non-null   object 
 2   activity_id                 2452 non-null   int64  
 3   activity_properties         2452 non-null   object 
 4   assay_chembl_id             2452 non-null   object 
 5   assay_description           2452 non-null   object 
 6   assay_type                  2452 non-null   object 
 7   assay_variant_accession     0 non-null      object 
 8   assay_variant_mutation      0 non-null      object 
 9   bao_endpoint                2452 non-null   object 
 10  bao_format                  2452 non-null   object 
 11  bao_label                   2452 non-null   object 
 12  canonical_smiles            2419 non-null   object 
 13  data_validity_comment       7 non-null      object 
 14  data_validity_description   7 non-null      object 
 15  document_chembl_id          2452 non-null   object 
 16  document_journal            1339 non-null   object 
 17  document_year               1395 non-null   float64
 18  ligand_efficiency           674 non-null    object 
 19  molecule_chembl_id          2452 non-null   object 
 20  molecule_pref_name          982 non-null    object 
 21  parent_molecule_chembl_id   2452 non-null   object 
 22  pchembl_value               1187 non-null   object 
 23  potential_duplicate         2452 non-null   int64  
 24  qudt_units                  1531 non-null   object 
 25  record_id                   2452 non-null   int64  
 26  relation                    1496 non-null   object 
 27  src_id                      2452 non-null   int64  
 28  standard_flag               2452 non-null   int64  
 29  standard_relation           1496 non-null   object 
 30  standard_text_value         0 non-null      object 
 31  standard_type               2452 non-null   object 
 32  standard_units              1531 non-null   object 
 33  standard_upper_value        0 non-null      object 
 34  standard_value              1509 non-null   object 
 35  target_chembl_id            2452 non-null   object 
 36  target_organism             2452 non-null   object 
 37  target_pref_name            2452 non-null   object
 38  target_tax_id               2452 non-null   object
 39  text_value                  0 non-null      object
 40  toid                        0 non-null      object
 41  type                        2452 non-null   object
 42  units                       1469 non-null   object
 43  uo_units                    1531 non-null   object
 44  upper_value                 7 non-null      object
 45  value                       1509 non-null   object
dtypes: float64(1), int64(5), object(40)
memory usage: 881.3+ KB
```

```
Out[35]:  Index(['action_type', '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')
```

```
In [40]: df2 = df.dropna(subset=["standard_value", "canonical_smiles"])
```

```
In [42]: df2
```

Out[42]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession
0	None	None	32544	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
1	None	None	33756	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
2	None	None	38634	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
3	None	None	38643	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
4	None	None	39754	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
...	...	...	...	...	...	...	...	...
2447	{'action_type': 'ANTAGONIST', 'description': '...	None	25631924	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2448	{'action_type': 'ANTAGONIST', 'description': '...	None	25631925	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2449	{'action_type': 'ANTAGONIST', 'description': '...	None	25631926	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
2450	None	None	25778419	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465602	Selectivity interaction (BET inhibitor selecti...	B	None

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession
2451	None	None	25787567	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474455	Selectivity interaction (Opioid Mu GTPgS agoni...	B	None

1509 rows × 46 columns

```
In [46]: len(df2.canonical_smiles.unique())
```

```
Out[46]: 1301
```

```
In [51]: df2_nr = df2.drop_duplicates(subset="canonical_smiles", keep="first").reset_index(drop=True)
```

```
In [54]: df2_nr
```

Out[54]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession
0	None	None	32544	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
1	None	None	33756	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
2	None	None	38634	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
3	None	None	38643	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
4	None	None	39754	[]	CHEMBL749749	Binding affinity on cloned opioid receptor mu ...	B	None
...	...	...	...	...	...	...	...	...
1296	{'action_type': 'ANTAGONIST', 'description': '...	None	25631924	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
1297	{'action_type': 'ANTAGONIST', 'description': '...	None	25631925	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
1298	{'action_type': 'ANTAGONIST', 'description': '...	None	25631926	[]	CHEMBL5365899	Antagonist activity at human mu opioid recepto...	F	None
1299	None	None	25778419	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465602	Selectivity interaction (BET inhibitor selecti...	B	None

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession
1300	None	None	25787567	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474455	Selectivity interaction (Opioid Mu GTPgS agoni...	B	None

1301 rows × 46 columns

In [57]:
```
df2_nr.columns
```

Out[57]:
```
Index(['action_type', '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')
```

Data pre-processing of the bioactivity data

Combine the 3 columns (molecule_chembl_id,canonical_smiles,standard_value) and bioactivity_class into

In [62]:
```
selection = ['molecule_chembl_id','canonical_smiles','standard_value']
df3 = df2_nr[selection]
df3
```

Out[62]:

	molecule_chembl_id	canonical_smiles	standard_value
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0
...	...	...	...
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	210.0
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	110.0
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	29.0
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	100000.0
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0

1301 rows × 3 columns

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.

In [68]:

```
bioactivity_threshold = []
for i in df3.standard_value:
  if float(i) >= 10000:
    bioactivity_threshold.append("inactive")
  elif float(i) <= 1000:
    bioactivity_threshold.append("active")
  else:
    bioactivity_threshold.append("intermediate")
```

In [71]:
```
bioactivity_class = pd.Series(bioactivity_threshold, name='class')
df5 = pd.concat([df3, bioactivity_class], axis=1)
df5
```

Out[71]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0	intermediate
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0	intermediate
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0	active
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0	intermediate
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0	inactive
...	...	...	...	...
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	210.0	active
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	110.0	active
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	29.0	active
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	100000.0	inactive
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0	inactive

1301 rows × 4 columns

In [78]:
```
df5.to_csv('opioid receptor_03_bioactivity_data_curated.csv', index=False)
```

In [84]:
```
df = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\opioid receptor\opioid receptor_03_bioactivity_data_curated.csv")
df
```

Out[84]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0	intermediate
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0	intermediate
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0	active
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0	intermediate
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0	inactive
...	...	...	...	...
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	210.0	active
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	110.0	active
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	29.0	active
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	100000.0	inactive
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0	inactive

1301 rows × 4 columns

In [88]:
```
df_no_smiles = df.drop(columns='canonical_smiles')
```

In [91]:
```
smiles = []

for i in df.canonical_smiles.tolist():
  cpd = str(i).split('.')
  cpd_longest = max(cpd, key = len)
  smiles.append(cpd_longest)

smiles = pd.Series(smiles, name = 'canonical_smiles')
```

In [94]:
```
df_clean_smiles = pd.concat([df_no_smiles,smiles], axis=1)
df_clean_smiles
```

Out[94]:

	molecule_chembl_id	standard_value	class	canonical_smiles
0	CHEMBL423694	5250.0	intermediate	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1
1	CHEMBL278078	3480.0	intermediate	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1
2	CHEMBL13470	320.0	active	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...
3	CHEMBL127802	4130.0	intermediate	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1
4	CHEMBL126946	10000.0	inactive	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1
...	...	...	...	...
1296	CHEMBL5429557	210.0	active	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...
1297	CHEMBL5397104	110.0	active	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...
1298	CHEMBL5414957	29.0	active	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...
1299	CHEMBL2017291	100000.0	inactive	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...
1300	CHEMBL3643413	30000.0	inactive	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...

1301 rows × 4 columns

Calculate Lipinski descriptors

Christopher Lipinski, a scientist at Pfizer, came up with a set of rule-of-thumb for evaluating the druglikeness of compounds. Such druglikeness is based on the Absorption, Distribution, Metabolism and Excretion (ADME) that is also known as the pharmacokinetic profile. Lipinski analyzed all orally active FDA-approved drugs in the formulation of what is to be known as the Rule-of-Five or Lipinski's Rule.

The Lipinski's Rule stated the following:

Molecular weight < 500 Dalton Octanol-water partition coefficient (LogP) < 5 Hydrogen bond donors < 5 Hydrogen bond acceptors < 10

In [99]:
```python
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors, Lipinski
```

Calculate descriptors

In [105…
```python
# Inspired by: https://codeocean.com/explore/capsules?query=tag:data-curation

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
```

In [109…

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

Out[109…

	MW	LogP	NumHDonors	NumHAcceptors
0	378.516	4.36250	1.0	3.0
1	378.516	4.36250	1.0	3.0
2	449.639	4.84720	0.0	4.0
3	404.598	5.86640	0.0	2.0
4	366.480	4.49300	1.0	2.0
...	...	...	...	...
1296	352.478	2.95890	2.0	3.0
1297	352.478	2.95890	2.0	3.0
1298	386.923	3.61230	2.0	3.0
1299	415.453	4.16254	1.0	7.0
1300	450.465	2.88450	1.0	7.0

1301 rows × 4 columns

Combine DataFrames

In [115…

```
df_lipinski
```

Out[115…

	MW	LogP	NumHDonors	NumHAcceptors
0	378.516	4.36250	1.0	3.0
1	378.516	4.36250	1.0	3.0
2	449.639	4.84720	0.0	4.0
3	404.598	5.86640	0.0	2.0
4	366.480	4.49300	1.0	2.0
...	...	...	...	...
1296	352.478	2.95890	2.0	3.0
1297	352.478	2.95890	2.0	3.0
1298	386.923	3.61230	2.0	3.0
1299	415.453	4.16254	1.0	7.0
1300	450.465	2.88450	1.0	7.0

1301 rows × 4 columns

In [117…

```
df_lipinski.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1301 entries, 0 to 1300
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MW             1301 non-null   float64
 1   LogP           1301 non-null   float64
 2   NumHDonors     1301 non-null   float64
 3   NumHAcceptors  1301 non-null   float64
dtypes: float64(4)
memory usage: 40.8 KB
```

In [119…

```
df_lipinski.describe()
```

Out[119…

	MW	LogP	NumHDonors	NumHAcceptors
count	1301.000000	1301.000000	1301.000000	1301.000000
mean	460.980948	3.963422	2.023828	4.727902
std	181.132768	1.958788	2.685611	2.663670
min	115.220000	-8.073000	0.000000	1.000000
25%	365.433000	3.114900	1.000000	3.000000
50%	419.160000	3.950440	1.000000	4.000000
75%	496.604000	4.958700	2.000000	6.000000
max	1904.213000	12.487600	29.000000	26.000000

In [122…

```
df
```

Out[122…

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0	intermediate
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0	intermediate
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0	active
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0	intermediate
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0	inactive
...	...	...	...	...
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	210.0	active
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	110.0	active
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	29.0	active
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	100000.0	inactive
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0	inactive

1301 rows × 4 columns

In [124…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1301 entries, 0 to 1300
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  1301 non-null   object 
 1   canonical_smiles    1301 non-null   object 
 2   standard_value      1301 non-null   float64
 3   class               1301 non-null   object 
dtypes: float64(1), object(3)
memory usage: 40.8+ KB
```

In [126…

```
df.describe()
```

Out[126...

	standard_value
count	1.301000e+03
mean	9.888287e+03
std	6.473024e+04
min	-2.810000e+04
25%	5.800000e+01
50%	1.100000e+03
75%	9.600000e+03
max	2.000000e+06

combine the 2 DataFrame

In [132...

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

In [135...

```
df_combined
```

Out[135…

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	Nu
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0	intermediate	378.516	4.36250	1.0	
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0	intermediate	378.516	4.36250	1.0	
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0	active	449.639	4.84720	0.0	
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0	intermediate	404.598	5.86640	0.0	
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0	inactive	366.480	4.49300	1.0	
...	...	...	...	...	...	...	...	
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	210.0	active	352.478	2.95890	2.0	
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	110.0	active	352.478	2.95890	2.0	
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	29.0	active	386.923	3.61230	2.0	
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	100000.0	inactive	415.453	4.16254	1.0	
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0	inactive	450.465	2.88450	1.0	

1301 rows × 8 columns

1301 rows × 8 columns

In [137…

```
df_combined.head()
```

Out[137…

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHA
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	5250.0	intermediate	378.516	4.3625	1.0	
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	3480.0	intermediate	378.516	4.3625	1.0	
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	320.0	active	449.639	4.8472	0.0	
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	4130.0	intermediate	404.598	5.8664	0.0	
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	10000.0	inactive	366.480	4.4930	1.0	

In [139…

```
df_combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1301 entries, 0 to 1300
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  1301 non-null   object 
 1   canonical_smiles    1301 non-null   object 
 2   standard_value      1301 non-null   float64
 3   class               1301 non-null   object 
 4   MW                  1301 non-null   float64
 5   LogP                1301 non-null   float64
 6   NumHDonors          1301 non-null   float64
 7   NumHAcceptors       1301 non-null   float64
dtypes: float64(5), object(3)
memory usage: 81.4+ KB
```

In [141…

```
df_combined.describe()
```

Out[141…

	standard_value	MW	LogP	NumHDonors	NumHAcceptors
count	1.301000e+03	1301.000000	1301.000000	1301.000000	1301.000000
mean	9.888287e+03	460.980948	3.963422	2.023828	4.727902
std	6.473024e+04	181.132768	1.958788	2.685611	2.663670
min	-2.810000e+04	115.220000	-8.073000	0.000000	1.000000
25%	5.800000e+01	365.433000	3.114900	1.000000	3.000000
50%	1.100000e+03	419.160000	3.950440	1.000000	4.000000
75%	9.600000e+03	496.604000	4.958700	2.000000	6.000000
max	2.000000e+06	1904.213000	12.487600	29.000000	26.000000

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 Take the molar value and apply -log10 Delete the standard_value column and create a new pIC50 column

In [148…

```
# https://github.com/chaninlab/estrogen-receptor-alpha-qsar/blob/master/02_ER_alpha_RO5.ipynb

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
```

In [151…

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

Out[151…

```
count     1.301000e+03
mean      9.888287e+03
std       6.473024e+04
min      -2.810000e+04
25%       5.800000e+01
50%       1.100000e+03
75%       9.600000e+03
max       2.000000e+06
Name: standard_value, dtype: float64
```

In [154…

```
-np.log10( (10**-9)* 100000000 )
```

Out[154…

```
1.0
```

In [156…

```
-np.log10( (10**-9)* 10000000000 )
```

Out[156…

```
-1.0
```

In [162…

```
def norm_value(input):
    norm = []

    for i in input['standard_value']:
        if i > 100000000:
          i = 100000000
        norm.append(i)

    input['standard_value_norm'] = norm
    x = input.drop('standard_value', axis=1)

    return x
```

We will first apply the norm_value() function so that the values in the standard_value column is normalized.

In [165…

```
df_norm = norm_value(df_combined)
df_norm
```

Out[165…

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	st
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	intermediate	378.516	4.36250	1.0	3.0	
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	intermediate	378.516	4.36250	1.0	3.0	
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	active	449.639	4.84720	0.0	4.0	
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	intermediate	404.598	5.86640	0.0	2.0	
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	inactive	366.480	4.49300	1.0	2.0	
...	...	...	...	...	...	...	...	
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	active	352.478	2.95890	2.0	3.0	
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	352.478	2.95890	2.0	3.0	
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	386.923	3.61230	2.0	3.0	
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453	4.16254	1.0	7.0	
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465	2.88450	1.0	7.0	

1301 rows × 8 columns

In [168…

```
df_norm.standard_value_norm
```

```
Out[168…  0          5250.0
          1          3480.0
          2           320.0
          3          4130.0
          4         10000.0
                     ...
          1296        210.0
          1297        110.0
          1298         29.0
          1299     100000.0
          1300      30000.0
          Name: standard_value_norm, Length: 1301, dtype: float64
```

```
In [170…  df_norm.standard_value_norm.describe()
```

```
Out[170…  count     1.301000e+03
          mean      9.888287e+03
          std       6.473024e+04
          min      -2.810000e+04
          25%       5.800000e+01
          50%       1.100000e+03
          75%       9.600000e+03
          max       2.000000e+06
          Name: standard_value_norm, dtype: float64
```

```
In [184…  df_final = df_norm.drop('standard_value_norm', axis=1)
          df_final
```

Out[184…

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	
0	CHEMBL423694	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc(OC)c2)cc1	intermediate	378.516	4.36250	1.0	3.0	5.
1	CHEMBL278078	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccccc2OC)cc1	intermediate	378.516	4.36250	1.0	3.0	5.
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	active	449.639	4.84720	0.0	4.0	6.
3	CHEMBL127802	CCCCN1CCC(=C(c2ccccc2)c2ccc(C(=O)N(CC)CC)cc2)CC1	intermediate	404.598	5.86640	0.0	2.0	5.
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	inactive	366.480	4.49300	1.0	2.0	5.
...	...	...	...	...	...	...	...	
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	active	352.478	2.95890	2.0	3.0	6.
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	352.478	2.95890	2.0	3.0	6.
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	386.923	3.61230	2.0	3.0	7.
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453	4.16254	1.0	7.0	4.
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465	2.88450	1.0	7.0	4.

1301 rows × 8 columns

In [ ]:

In [188…

```
df_final.pIC50.describe()
```

Out[188…

```
count    1300.000000
mean        6.223332
std         1.394669
min         2.698970
25%         5.017729
50%         5.958607
75%         7.236572
max        10.903090
Name: pIC50, dtype: float64
```

Removing the 'intermediate' bioactivity class

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

In [194…

```
df_2class = df_final[df_final["class"]  != 'intermediate']
df_2class
```

Out[194…

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC
2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	active	449.639	4.84720	0.0	4.0	6.4948
4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	inactive	366.480	4.49300	1.0	2.0	5.0000
8	CHEMBL338089	CCN(CC)C(=O)c1ccc(C(=C2CCN(c3ccccc3)CC2)c2cccc...	inactive	424.588	6.27090	0.0	2.0	5.0000
11	CHEMBL126842	CCN(CC)C(=O)c1ccc(C(=C2CCN(C)CC2)c2ccccc2)cc1	inactive	362.517	4.69610	0.0	2.0	5.0000
17	CHEMBL129034	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc3ccccc23)cc1	active	398.550	5.50710	1.0	2.0	6.0877
...	...	...	...	...	...	...	...	
1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	active	352.478	2.95890	2.0	3.0	6.6777
1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	352.478	2.95890	2.0	3.0	6.9586
1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	386.923	3.61230	2.0	3.0	7.5376
1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453	4.16254	1.0	7.0	4.0000
1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465	2.88450	1.0	7.0	4.5228

956 rows × 8 columns

In [319…
```
df_2class.to_csv('opioid receptor_04_bioactivity_data_3class_pIC50.csv')
```

In [317…
```
df_2class.columns
```

Out[317…
```
Index(['molecule_chembl_id', 'canonical_smiles', 'class', 'MW', 'LogP',
       'NumHDonors', 'NumHAcceptors', 'pIC50'],
      dtype='object')
```

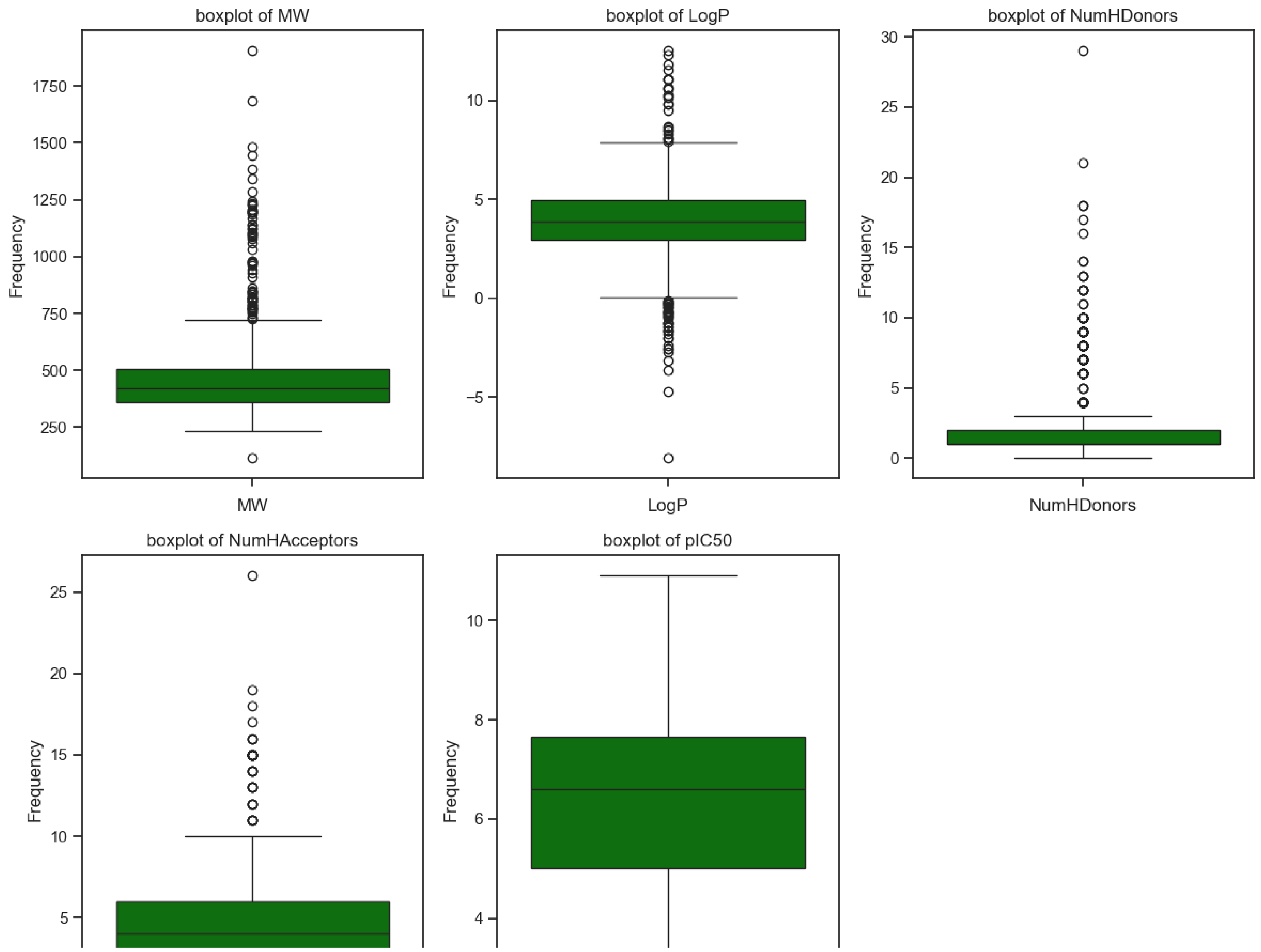
Exploratory Data Analysis (Chemical Space Analysis) via Lipinski descriptors

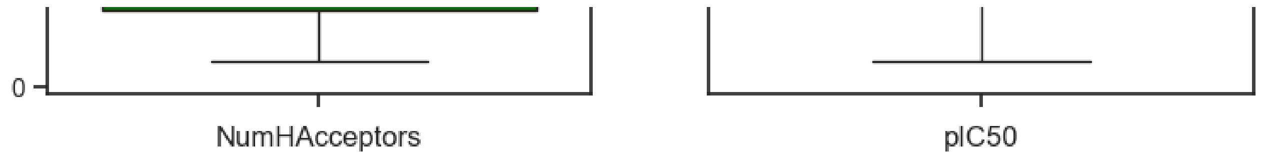
In [203…

```
import seaborn as sns
sns.set(style='ticks')
import matplotlib.pyplot as plt
```

In [210…

```
plt.figure(figsize=(12, 10))
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors', 'pIC50']
for i, column in enumerate(numeric_columns, 1):
 plt.subplot(2, 3, i) # 2 rows and 3 columns for better layout
 sns.boxplot(df_2class[column],color='green') # kde=True for kernel density estimate
 plt.title(f'boxplot of {column}')
 plt.xlabel(column)
 plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

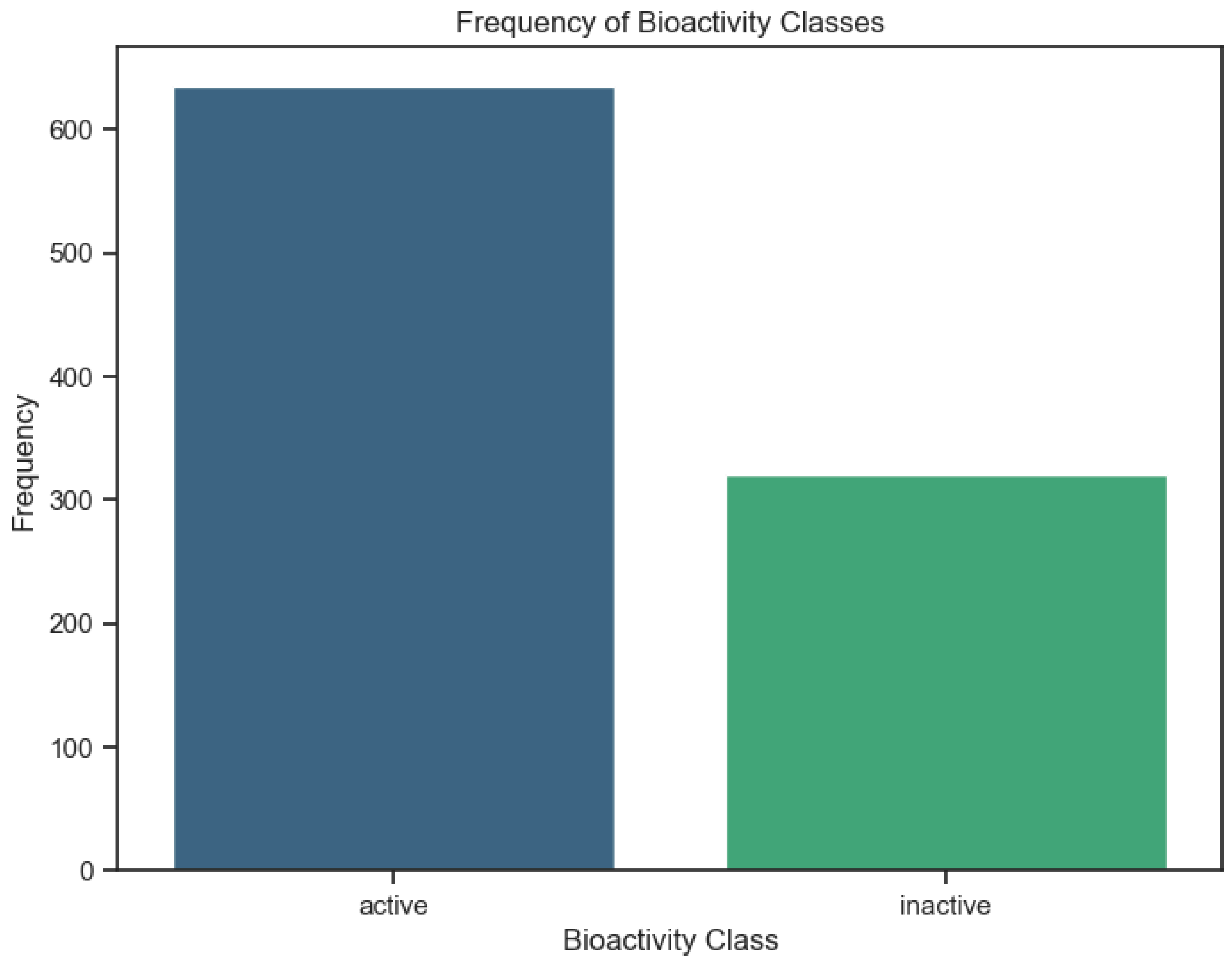

boxplot of MW
Frequency
MW
boxplot of LogP
Frequency
LogP
boxplot of NumHDonors
Frequency
NumHDonors
boxplot of NumHAcceptors
Frequency
boxplot of pIC50
Frequency



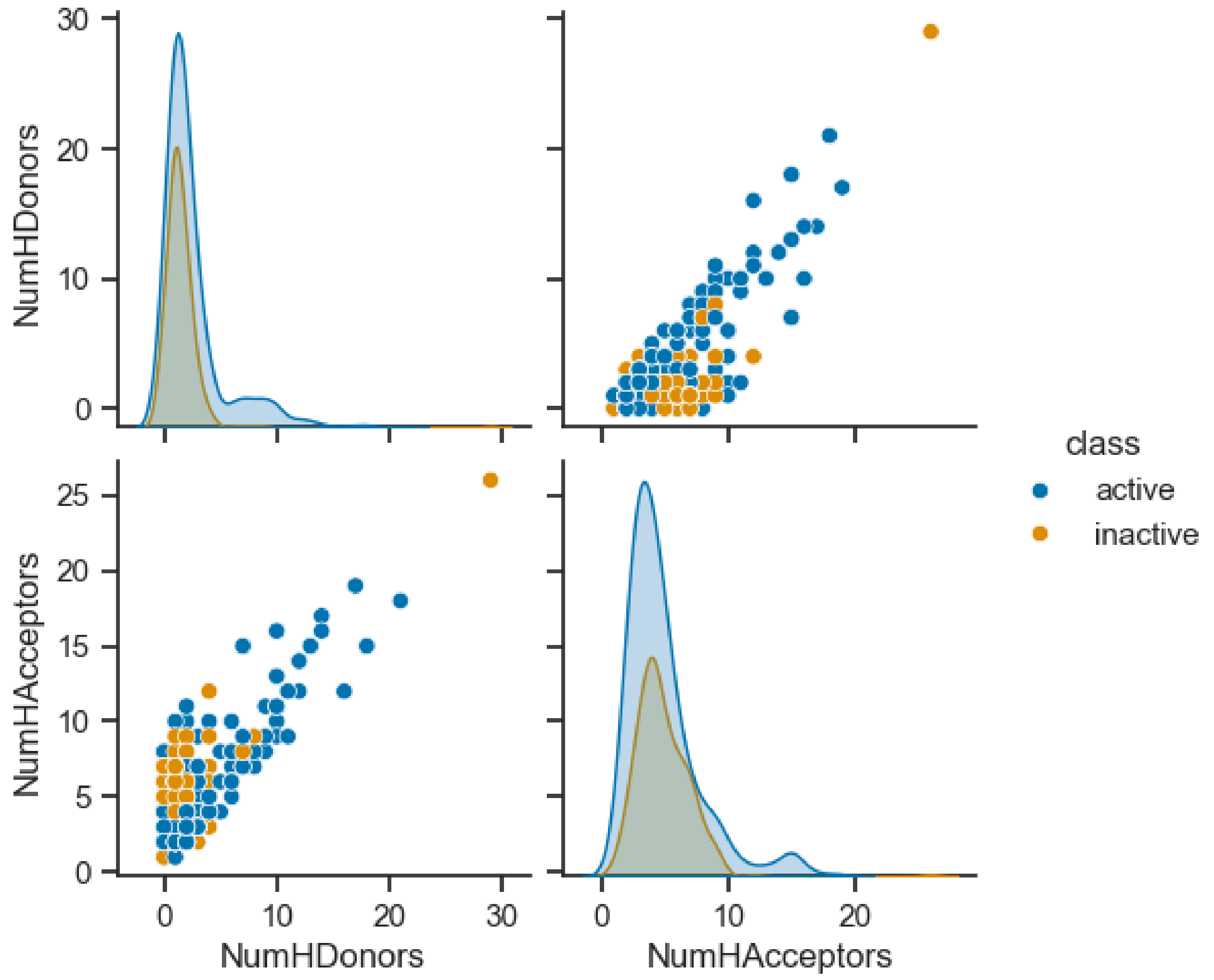
Frequency of Bioactivity Classes

In [213…

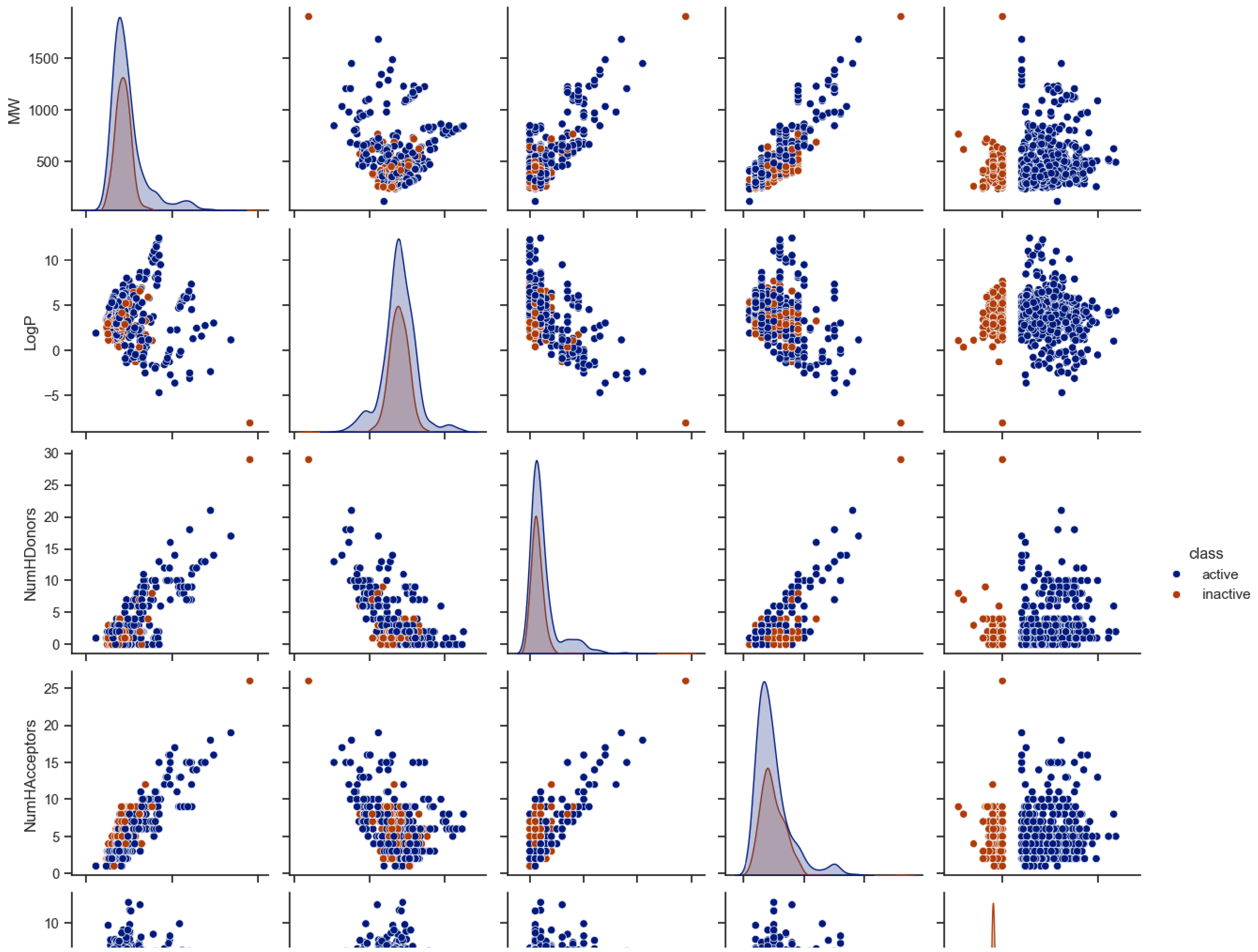
```
# Set figure size
plt.figure(figsize=(8, 6))
# Count plot with hue
sns.countplot(x='class', hue='class', data=df_2class, palette='viridis')
# Add labels and title
plt.xlabel('Bioactivity Class')
plt.ylabel('Frequency')
plt.title('Frequency of Bioactivity Classes')
# Show plot
plt.show()
```

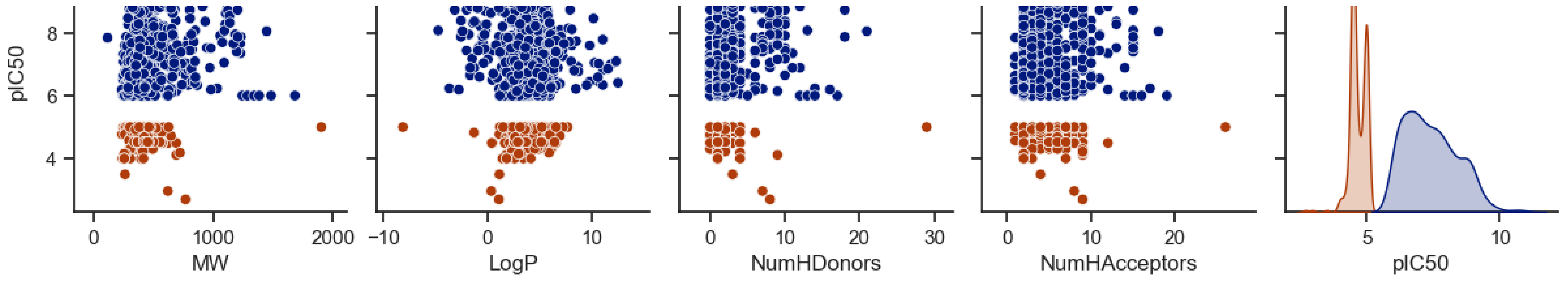


```
In [220…  sns.pairplot(df_2class, vars=['NumHDonors', 'NumHAcceptors'], hue='class', palette='colorblind')
          # Show plot
          plt.show()
```



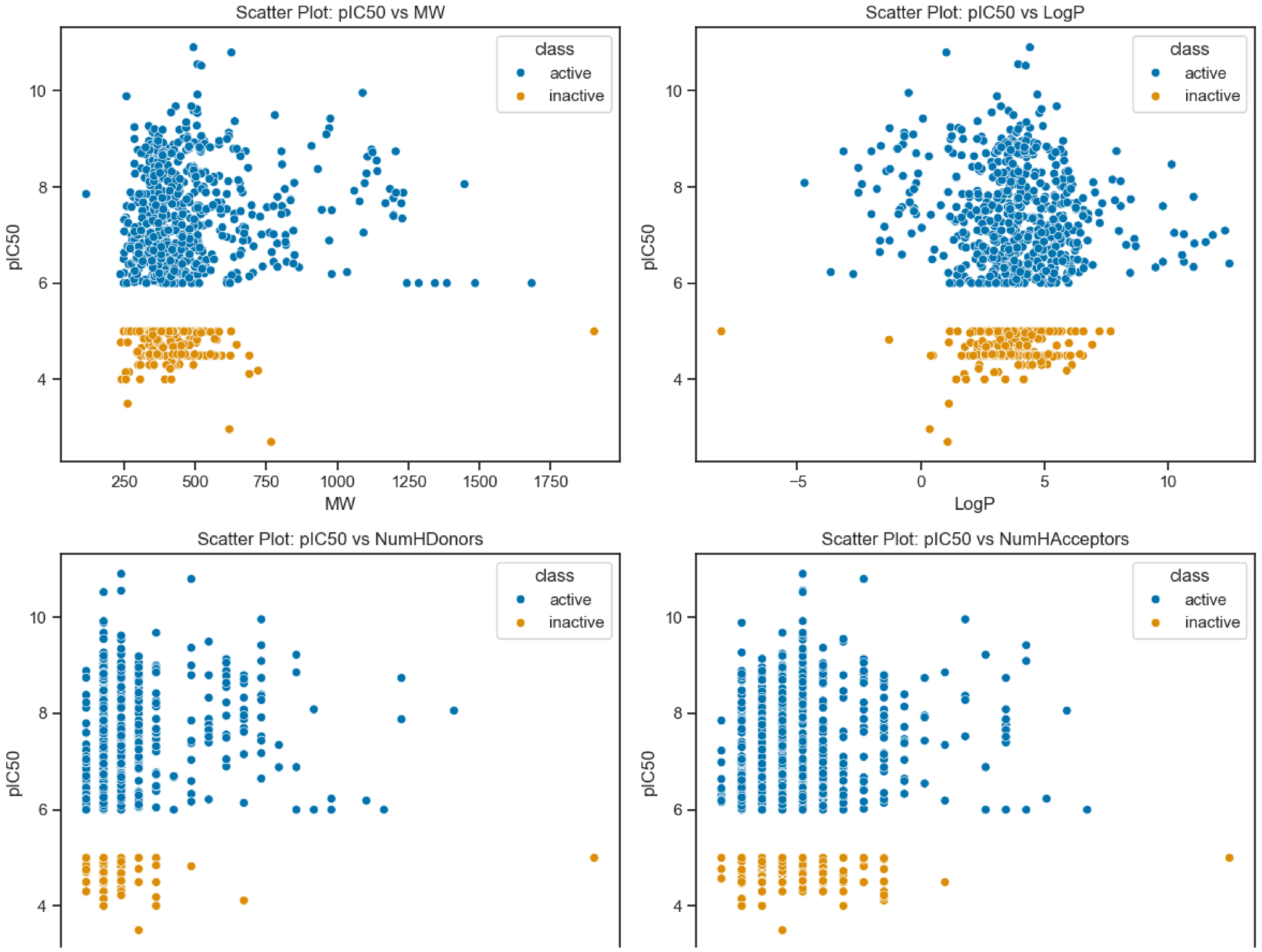
```
In [224…  sns.pairplot(df_2class, hue='class', palette='dark')
          # Show plot
          plt.show()
```

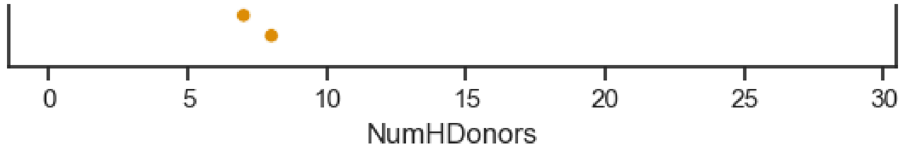

MW
1500
1000
500
LogP
10
5
0
−5
NumHDonors
30
25
20
15
10
5
0
NumHAcceptors
25
20
15
10
5
0
10
class
● active
● inactive

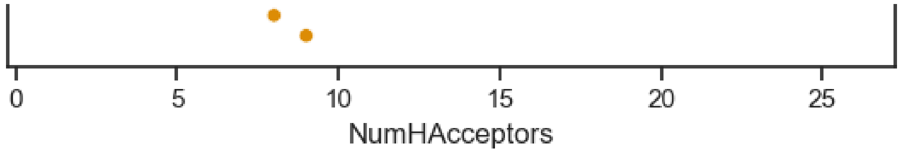


In [230…

```python
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors']
# Set figure size
plt.figure(figsize=(12, 10))
# Loop through numeric columns and create scatter plots
for i, column in enumerate(numeric_columns, 1):
 plt.subplot(2, 2, i) # 2 rows, 2 columns layout
 sns.scatterplot(x=df_2class[column], y=df_2class['pIC50'], hue=df_2class['class'], palette='colorblind')
 plt.title(f'Scatter Plot: pIC50 vs {column}')
 plt.xlabel(column)
 plt.ylabel('pIC50')
# Adjust layout
plt.tight_layout()
plt.show()
```

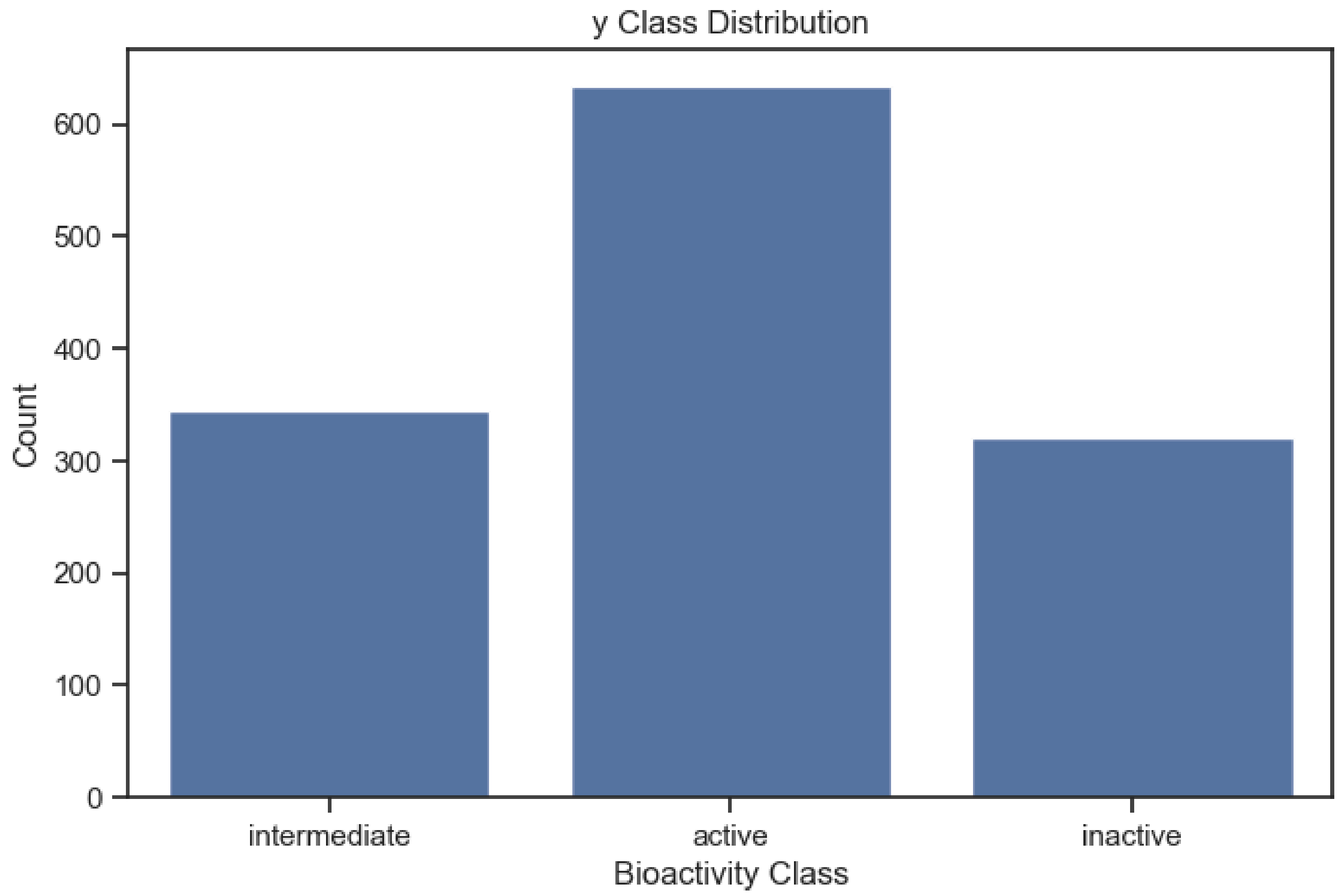

Scatter Plot: pIC50 vs MW
pIC50
MW
class
active
inactive
Scatter Plot: pIC50 vs LogP
pIC50
LogP
class
active
inactive
Scatter Plot: pIC50 vs NumHDonors
pIC50
class
active
inactive
Scatter Plot: pIC50 vs NumHAcceptors
pIC50
class
active
inactive





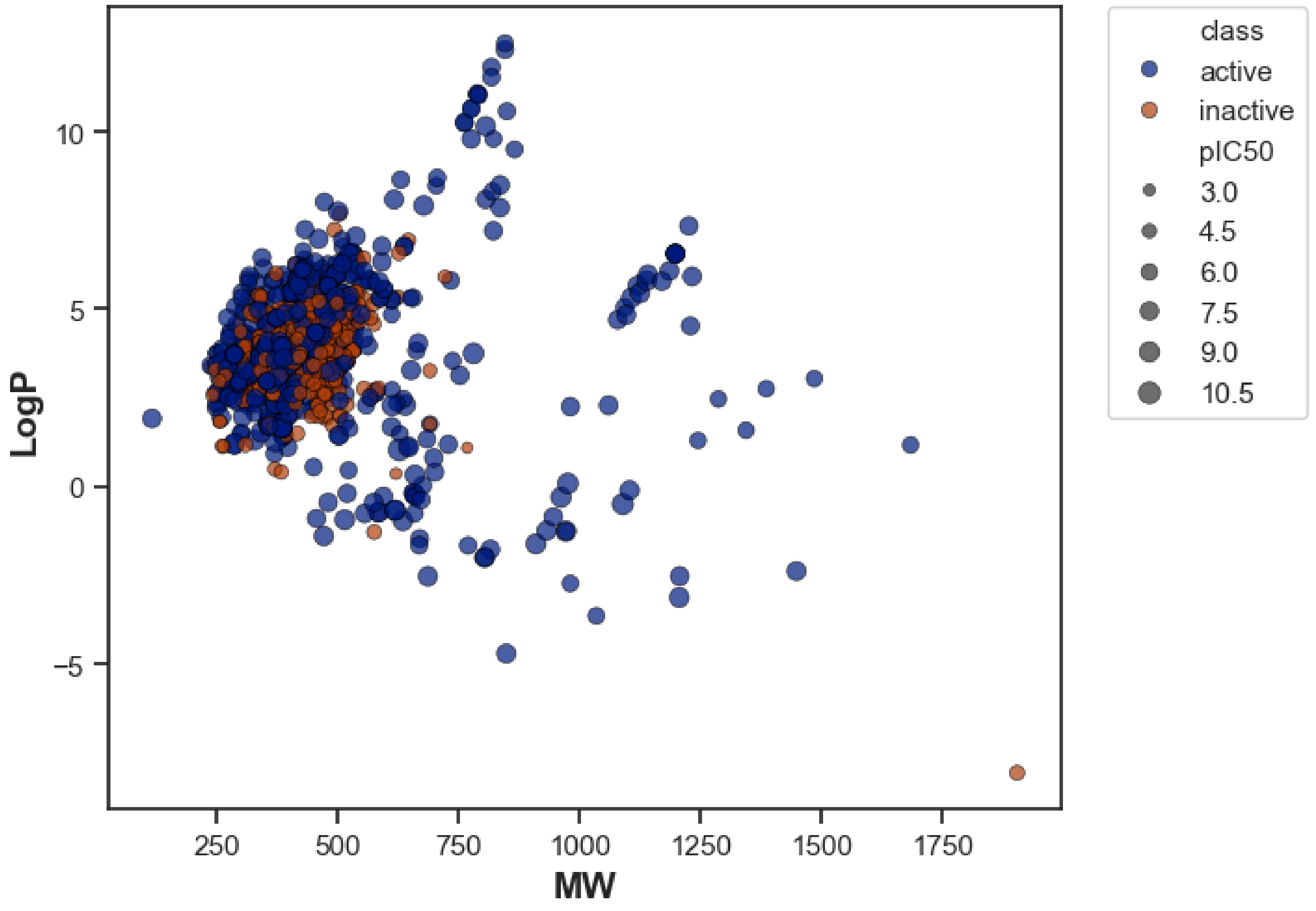
In [233…

```
plt.figure(figsize=(8, 5))
sns.countplot(x="class", data=df, )
plt.xlabel("Bioactivity Class")
plt.ylabel("Count")
plt.title("y Class Distribution")
plt.show()
```



In [236…

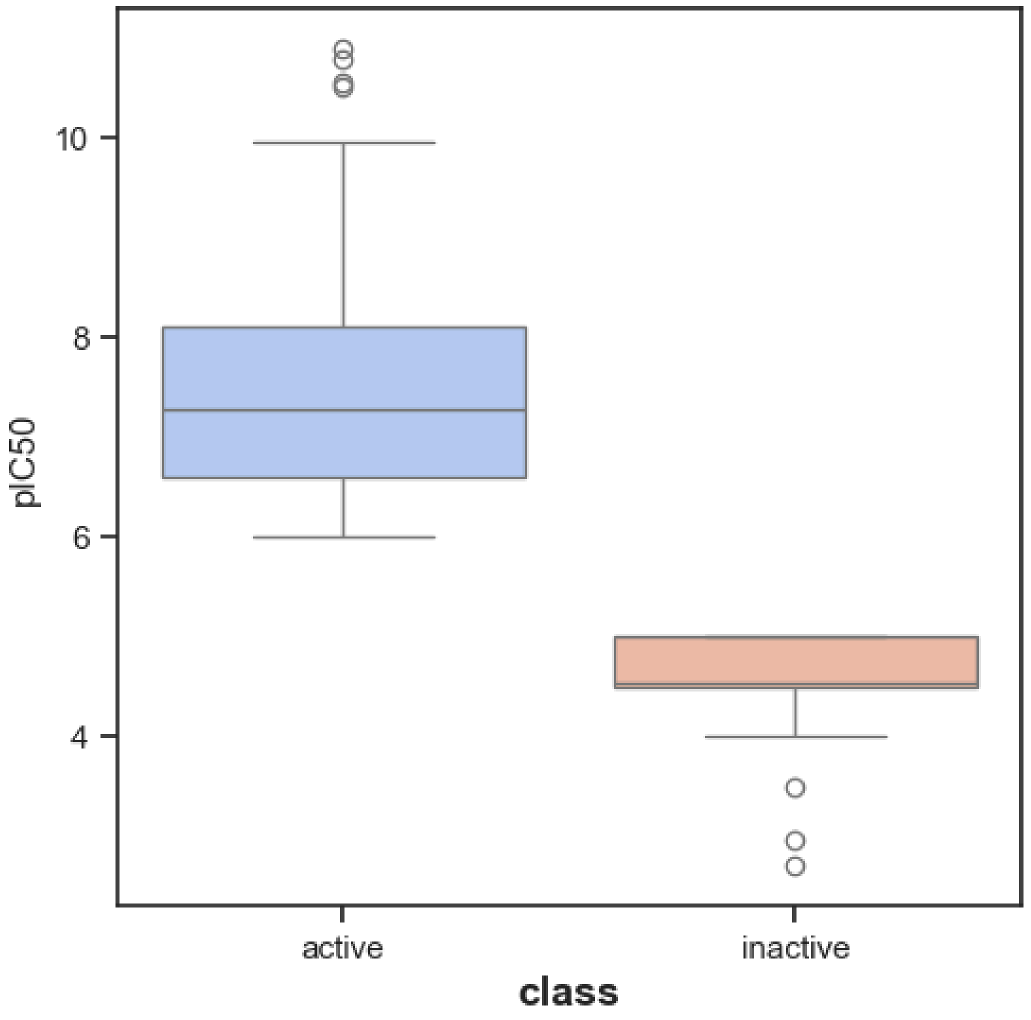
```
plt.figure(figsize=(6.6, 5.6))
sns.scatterplot(x='MW', y='LogP', data=df_2class, hue='class', size='pIC50',palette='dark', edgecolor='black', alpha=0.7)
plt.xlabel('MW', fontsize=14, fontweight='bold')
plt.ylabel('LogP', fontsize=14, fontweight='bold')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0)
plt.show()
```



In [241…

```
plt.figure(figsize=(5.5, 5.5))
sns.boxplot(x = 'class', y = 'pIC50', data = df_2class, hue = 'class' ,palette = 'coolwarm')
plt.xlabel('class', fontsize=14, fontweight='bold')
```

Out[241… Text(0.5, 0, 'class')



Statistical analysis | Mann-Whitney U Test

In [249…

```
# https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/
from numpy.random import seed
from scipy.stats import mannwhitneyu

def mannwhitney(descriptor, verbose=False):
    # Seed the random number generator for reproducibility
    seed(1)
    # Select relevant columns
    selection = [descriptor, 'class']
    df = df_2class[selection]

    # Split active and inactive classes
    active = df[df['class'] == 'active'][descriptor]
    inactive = df[df['class'] == 'inactive'][descriptor]

    # Perform Mann-Whitney U test
    stat, p = mannwhitneyu(active, inactive)

    # Interpret the result
    alpha = 0.05
    interpretation = 'Same distribution (fail to reject H0)' if p > alpha else 'Different distribution (reject H0)'

    # Store results in a DataFrame
    results = pd.DataFrame({
        'Descriptor': [descriptor],
        'Statistics': [stat],
        'p': [p],
        'alpha': [alpha],
        'Interpretation': [interpretation]
    })

    # Save results to CSV
    filename = f'mannwhitneyu_{descriptor}.csv'
    results.to_csv(filename, index=False)

    return results
```

In [252…

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

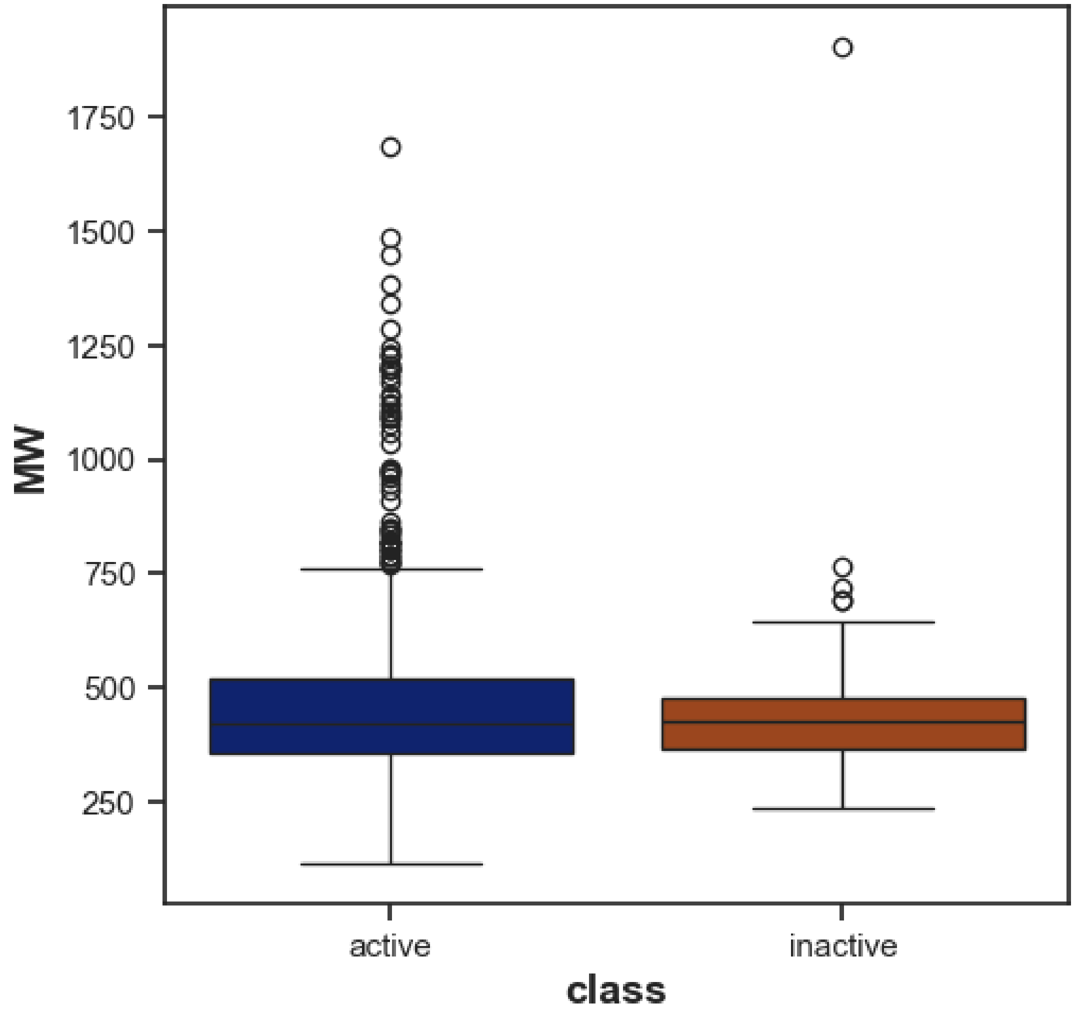
Out[252…

	Descriptor	Statistics	p	alpha	Interpretation
0	pIC50	NaN	NaN	0.05	Different distribution (reject H0)

MW

In [258…
```
plt.figure(figsize=(5.5, 5.5))
sns.boxplot(x = 'class', y = 'MW', data = df_2class,hue = 'class' , palette = 'dark')
plt.xlabel('class', fontsize=14, fontweight='bold')
plt.ylabel('MW', fontsize=14, fontweight='bold')
```

Out[258…
```
Text(0, 0.5, 'MW')
```



In [261…
```
mannwhitney('MW')
```

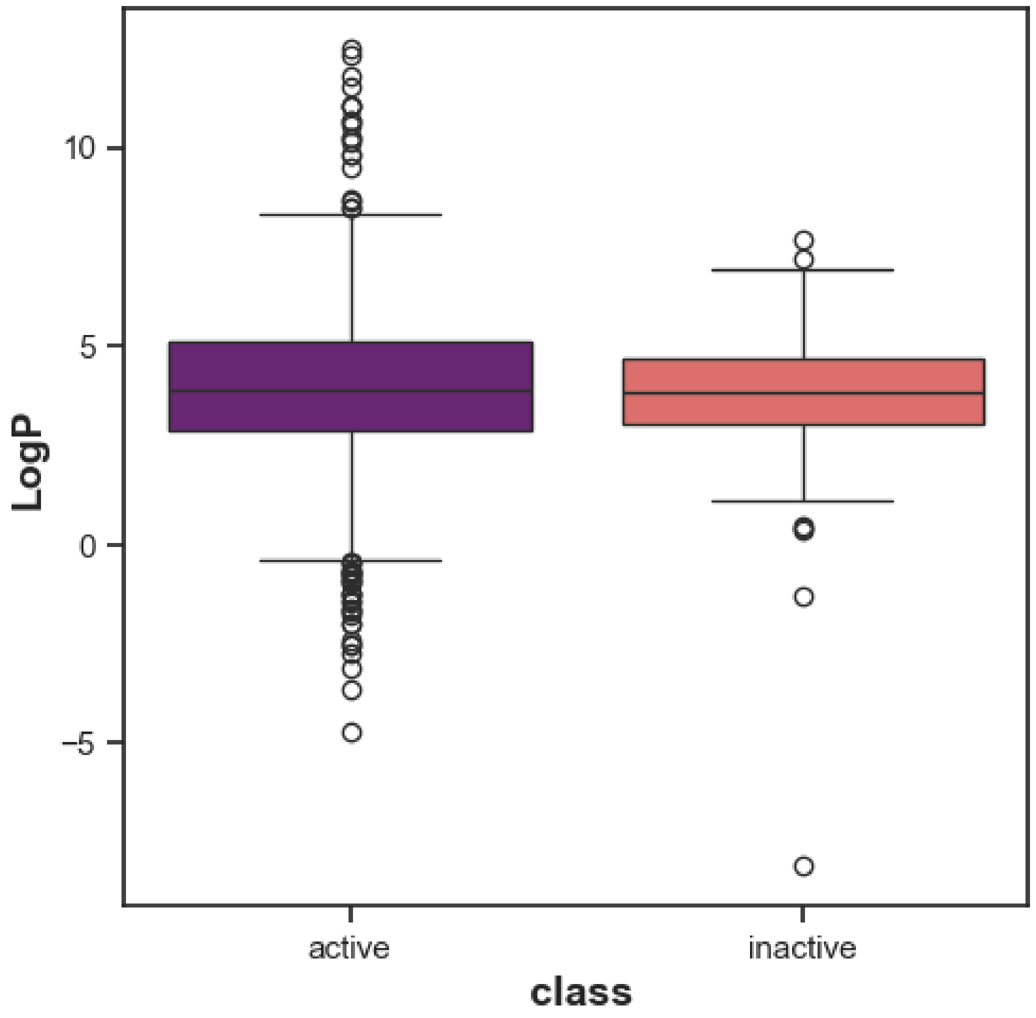
```
Out[261...
```

	Descriptor	Statistics	p	alpha	Interpretation
0	MW	108504.0	0.102365	0.05	Same distribution (fail to reject H0)

LogP

```
In [269...
plt.figure(figsize=(5.5, 5.5))

sns.boxplot(x = 'class', y = 'LogP', data = df_2class,hue = 'class' , palette = 'magma')

plt.xlabel(' class', fontsize=14, fontweight='bold')
plt.ylabel('LogP', fontsize=14, fontweight='bold')
```

```
Out[269... Text(0, 0.5, 'LogP')
```



Statistical analysis | Mann-Whitney U Test

In [272…
```
mannwhitney('LogP')
```

Out[272…

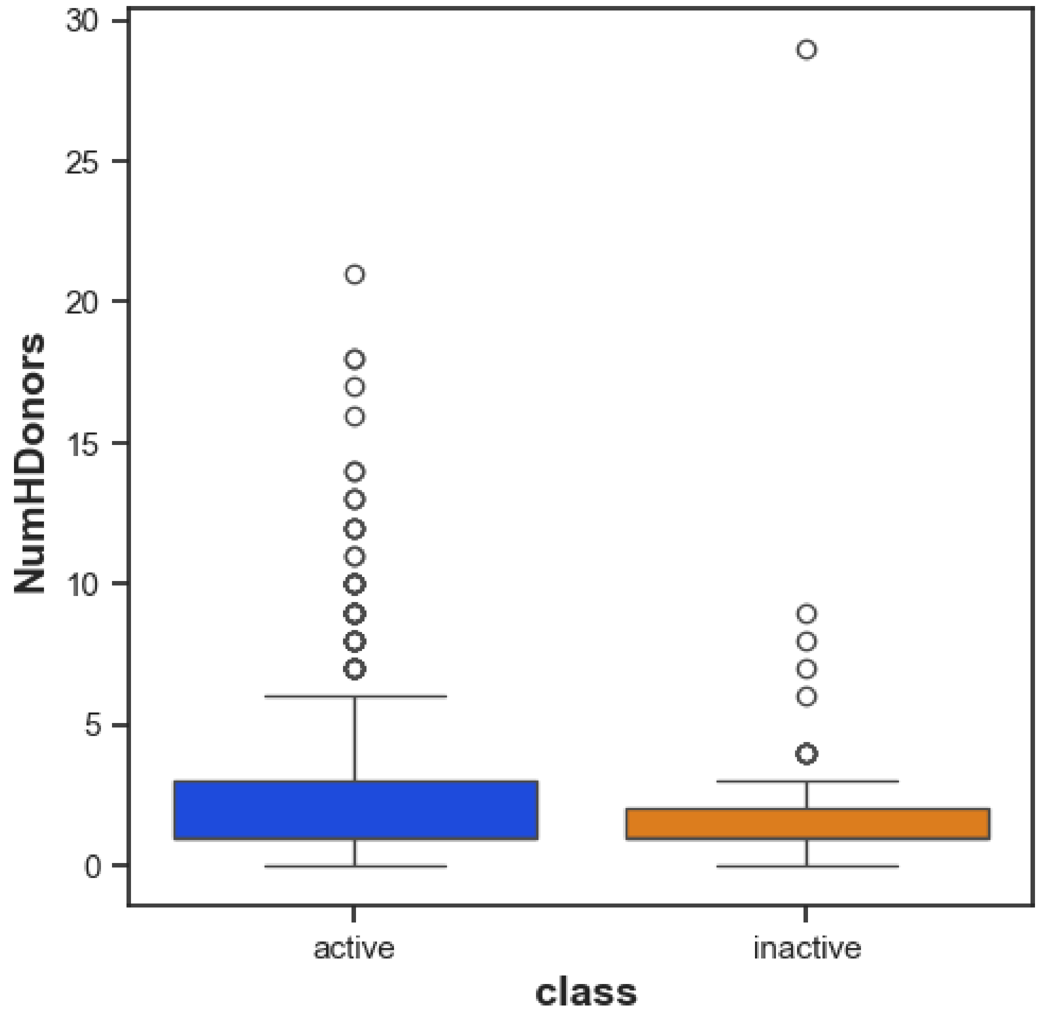
	Descriptor	Statistics	p	alpha	Interpretation
0	LogP	105271.5	0.405549	0.05	Same distribution (fail to reject H0)

NumHDonors

```
In [283…
```

```python
plt.figure(figsize=(5.5, 5.5))
plt.figure(figsize=(5.5, 5.5))
sns.boxplot(x = 'class', y = 'NumHDonors', data = df_2class,hue = 'class' , palette = 'bright')
plt.xlabel(' class', fontsize=14, fontweight='bold')
plt.ylabel('NumHDonors', fontsize=14, fontweight='bold')
```

```
Out[283… Text(0, 0.5, 'NumHDonors')
<Figure size 550x550 with 0 Axes>
```



Statistical analysis | Mann-Whitney U Test

In [293…
```
mannwhitney('NumHDonors')
```

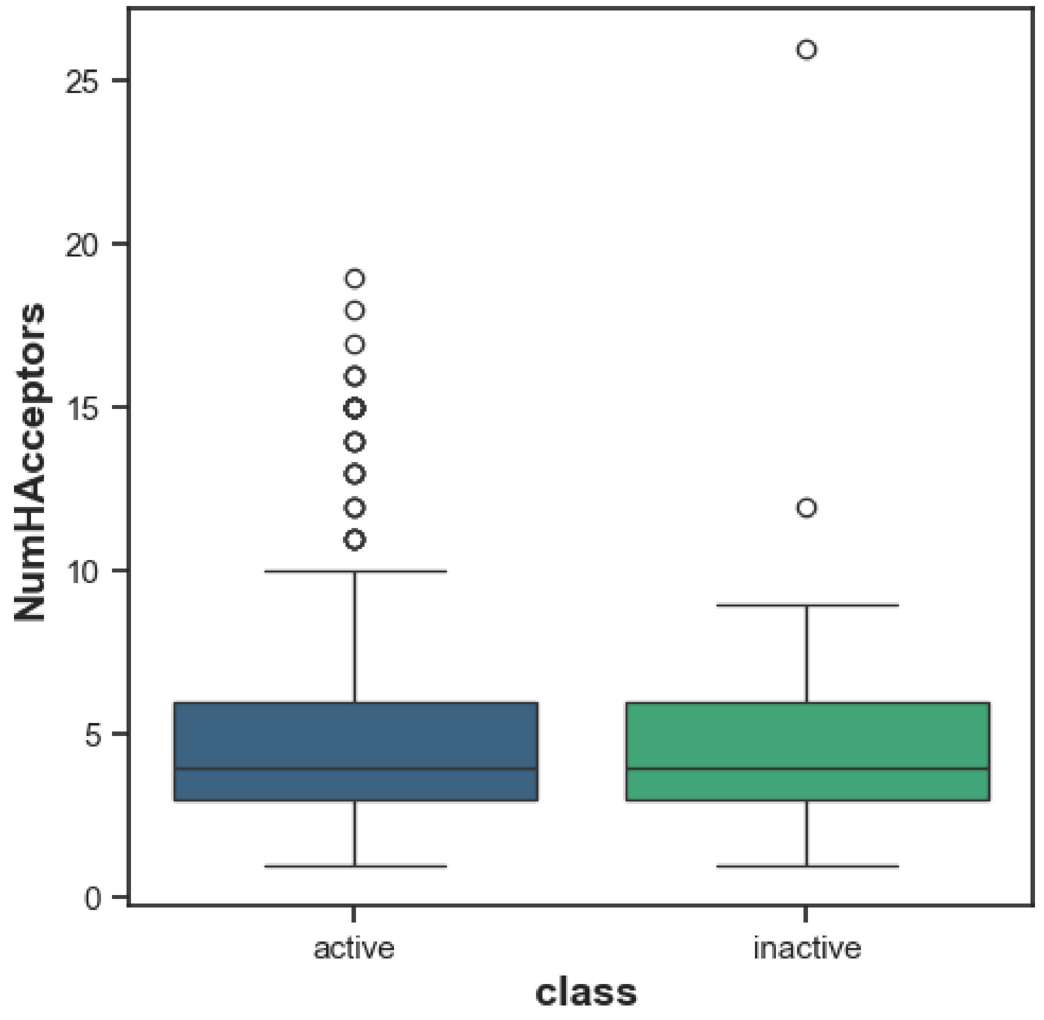
Out[293…

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHDonors	123067.0	4.233636e-08	0.05	Different distribution (reject H0)

NumHAcceptors

```
In [300…
plt.figure(figsize=(5.5, 5.5))

sns.boxplot(x = 'class', y = 'NumHAcceptors', data = df_2class,hue = 'class' , palette = 'viridis')

plt.xlabel(' class', fontsize=14, fontweight='bold')
plt.ylabel('NumHAcceptors', fontsize=14, fontweight='bold')
```

```
Out[300…
Text(0, 0.5, 'NumHAcceptors')
```



Statistical analysis | Mann-Whitney U Test

```
In [309…  mannwhitney('NumHAcceptors')
```

Out[309…

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHAcceptors	91430.0	0.008387	0.05	Different distribution (reject H0)

Box Plots pIC50 values Taking a look at pIC50 values, the actives and inactives displayed statistically significant difference, which is to be expected since threshold values (IC50 < 1,000 nM = Actives while IC50 > 10,000 nM = Inactives, corresponding to pIC50 > 6 = Actives and pIC50 < 5 =

Inactives) were used to define actives and inactives.

Lipinski's descriptors Of the 4 Lipinski's descriptors (MW, LogP, NumHDonors and NumHAcceptors), only NumHDonors and NumHAcceptors exhibited difference between the actives and inactives while the other descriptors (only (MW, LogP shows statistically significant same difference between actives and inactives.

In [323…
```
df3 = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\opioid receptor\opioid receptor_04_bioactivity_data_3class_pIC50.csv")
```

In [325…
```
df3
```

Out[325...

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAccept
0	2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)...	active	449.639	4.84720	0.0	
1	4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	inactive	366.480	4.49300	1.0	
2	8	CHEMBL338089	CCN(CC)C(=O)c1ccc(C(=C2CCN(c3ccccc3)CC2)c2cccc...	inactive	424.588	6.27090	0.0	
3	11	CHEMBL126842	CCN(CC)C(=O)c1ccc(C(=C2CCN(C)CC2)c2ccccc2)cc1	inactive	362.517	4.69610	0.0	
4	17	CHEMBL129034	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc3ccccc23)cc1	active	398.550	5.50710	1.0	
...	...	...	...	...	...	...	...	
951	1296	CHEMBL5429557	CN(C)[C@H](CNC(=O)[C@H]1C[C@]1(C)c1ccccc1)Cc1c...	active	352.478	2.95890	2.0	
952	1297	CHEMBL5397104	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	352.478	2.95890	2.0	
953	1298	CHEMBL5414957	CN(C)[C@H](CNC(=O)[C@@H]1C[C@]1(C)c1ccccc1)Cc1...	active	386.923	3.61230	2.0	
954	1299	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453	4.16254	1.0	
955	1300	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465	2.88450	1.0	

956 rows × 9 columns

In [328...

```
df3.head()
```

Out[328…

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors
0	2	CHEMBL13470	C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)…	active	449.639	4.8472	0.0	4.0
1	4	CHEMBL126946	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1	inactive	366.480	4.4930	1.0	2.0
2	8	CHEMBL338089	CCN(CC)C(=O)c1ccc(C(=C2CCN(c3ccccc3)CC2)c2cccc…	inactive	424.588	6.2709	0.0	2.0
3	11	CHEMBL126842	CCN(CC)C(=O)c1ccc(C(=C2CCN(C)CC2)c2ccccc2)cc1	inactive	362.517	4.6961	0.0	2.0
4	17	CHEMBL129034	CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc3ccccc23)cc1	active	398.550	5.5071	1.0	2.0

In [332…

```
df3.describe()
```

Out[332…

	Unnamed: 0	MW	LogP	NumHDonors	NumHAcceptors	pIC50
count	956.000000	956.000000	956.000000	956.000000	956.000000	955.000000
mean	677.293933	468.168969	3.875962	2.146444	4.843096	6.481876
std	364.371668	190.384243	2.008802	2.721929	2.745861	1.539089
min	2.000000	115.220000	-8.073000	0.000000	1.000000	2.698970
25%	388.250000	362.469250	2.958900	1.000000	3.000000	5.000000
50%	678.000000	423.794500	3.868980	1.000000	4.000000	6.596879
75%	998.250000	507.030000	4.937450	2.000000	6.000000	7.657577
max	1300.000000	1904.213000	12.487600	29.000000	26.000000	10.903090

In [345…

```
selection = ['canonical_smiles','molecule_chembl_id']
df3_selection = df3[selection]
df3_selection.to_csv('molecule.smi', sep='\t', index=False, header=False)
```

```
In [347…  with open('molecule.smi', 'r') as file:
           for _ in range(5):
               print(file.readline().strip())
```

```
C=CCN1C[C@H](C)N([C@H](c2ccc(C(=O)N(CC)CC)cc2)c2cccc(OC)c2)C[C@H]1C     CHEMBL13470
CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2ccc(F)cc2)cc1   CHEMBL126946
CCN(CC)C(=O)c1ccc(C(=C2CCN(c3ccccc3)CC2)c2ccccc2)cc1    CHEMBL338089
CCN(CC)C(=O)c1ccc(C(=C2CCN(C)CC2)c2ccccc2)cc1   CHEMBL126842
CCN(CC)C(=O)c1ccc(C(=C2CCNCC2)c2cccc3ccccc23)cc1        CHEMBL129034
```

```
In [349…  with open('molecule.smi', 'r') as file:
           line_count = sum(1 for line in file)
          print("Total number of lines:", line_count)
```

```
Total number of lines: 956
```

Preparing the X and Y Data Matrices

```
In [396…  X = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\New folder\archive\descriptors_output.csv")
```

```
In [397…  X
```

Out[397…

	Name	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	Pubche
0	CHEMBL130478	1	1	0	0	0	0	0	0	
1	CHEMBL336538	1	1	1	0	0	0	0	0	
2	CHEMBL339995	1	1	1	0	0	0	0	0	
3	CHEMBL341437	1	1	1	0	0	0	0	0	
4	CHEMBL130098	1	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
6941	CHEMBL253998	1	1	1	0	0	0	0	0	
6942	CHEMBL502	1	1	1	0	0	0	0	0	
6943	CHEMBL3085398	1	1	1	0	0	0	0	0	
6944	CHEMBL13045	1	1	1	0	0	0	0	0	
6945	CHEMBL417799	1	1	0	0	0	0	0	0	

6946 rows × 882 columns

In [399…

```
X.head()
```

Out[399...

	Name	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP
0	CHEMBL130478	1	1	0	0	0	0	0	0	
1	CHEMBL336538	1	1	1	0	0	0	0	0	
2	CHEMBL339995	1	1	1	0	0	0	0	0	
3	CHEMBL341437	1	1	1	0	0	0	0	0	
4	CHEMBL130098	1	1	0	0	0	0	0	0	

5 rows × 882 columns

In [402...

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6946 entries, 0 to 6945
Columns: 882 entries, Name to PubchemFP880
dtypes: int64(881), object(1)
memory usage: 46.7+ MB
```

In [403...

```
X.dtypes
```

Out[403...

```
Name             object
PubchemFP0        int64
PubchemFP1        int64
PubchemFP2        int64
PubchemFP3        int64
                  ...
PubchemFP876      int64
PubchemFP877      int64
PubchemFP878      int64
PubchemFP879      int64
PubchemFP880      int64
Length: 882, dtype: object
```

In [404...

```
X = X.drop(columns=['Name'])
X
```

Out[404…

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	Pubcheml
0	1	1	0	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	0	
2	1	1	1	0	0	0	0	0	0	
3	1	1	1	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
6941	1	1	1	0	0	0	0	0	0	
6942	1	1	1	0	0	0	0	0	0	
6943	1	1	1	0	0	0	0	0	0	
6944	1	1	1	0	0	0	0	0	0	
6945	1	1	0	0	0	0	0	0	0	

6946 rows × 881 columns

Y variable

In [409…

```
y = df3['class']
```

In [410…

```
y = y.map({'active': 1, 'inactive': 0})
```

Split dataset

In [414…

```
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (6946, 881)
Shape of y: (956,)
```

In [418...
```
X = X.iloc[:y.shape[0], :]  # Trim X to match y
```

In [421...
```
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (956, 881)
Shape of y: (956,)
```

In [424...
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [427...
```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [430...
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Logistic Regression Model

In [433...
```
lr = LogisticRegression(max_iter=600)
```

In [436...
```
lr.fit(X_train, y_train)
```

Out[436...

```
▾ LogisticRegression
LogisticRegression(max_iter=600)
```

In [439...
```
y_pred_lr = lr.predict(X_test)
y_pred_lr
```

```
Out[439…  array([1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
                 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1], dtype=int64)
```

```
In [444…  print(classification_report(y_test, y_pred_lr))
```

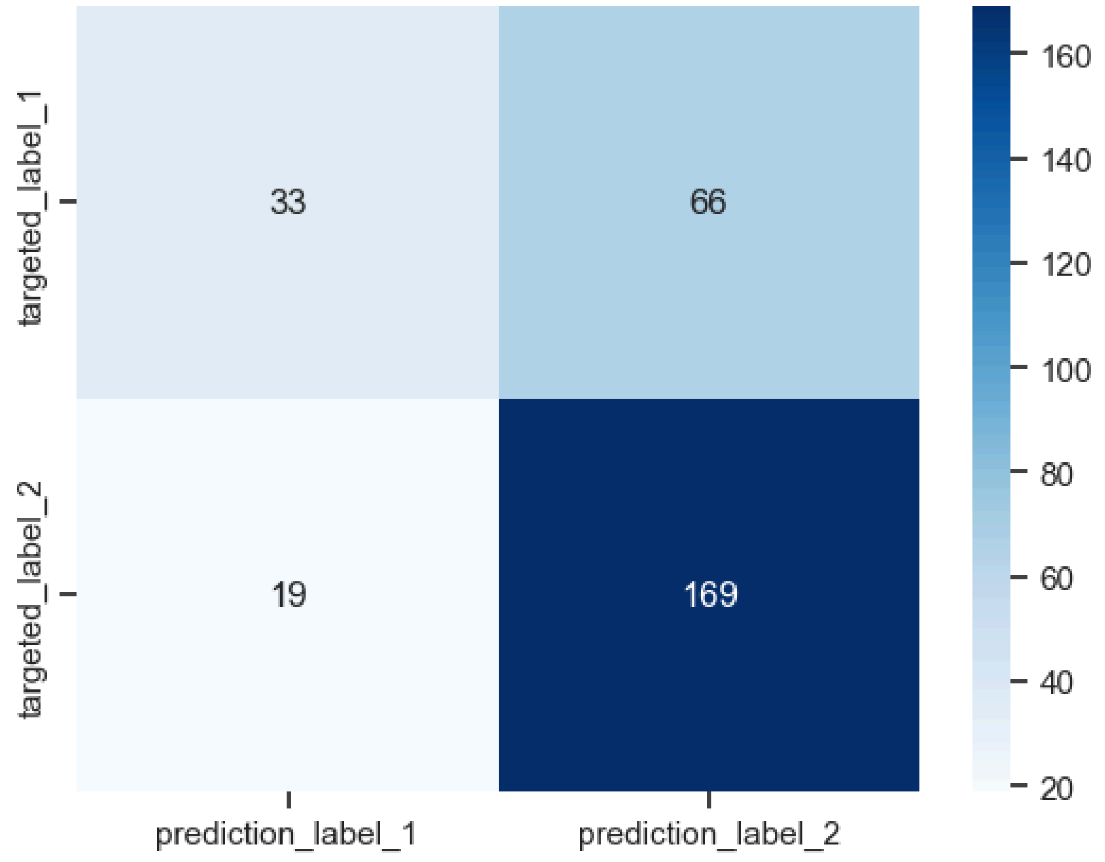
```
              precision    recall  f1-score   support

           0       0.63      0.33      0.44        99
           1       0.72      0.90      0.80       188

    accuracy                           0.70       287
   macro avg       0.68      0.62      0.62       287
weighted avg       0.69      0.70      0.67       287
```

```
In [447…  accuracy = accuracy_score(y_test, y_pred_lr)
          print(f"Logistic Regression Model Accuracy: {accuracy * 100:.2f}%")
```

```
Logistic Regression Model Accuracy: 70.38%
```

```
In [450…  cm = confusion_matrix(y_test, y_pred_lr)
          ax = sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
          ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
          ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

```
Out[450…  [Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



Logistic Regression Model Accuracy: 70.38%

In [457…
```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(X_train,y_train)
```

Out[457…
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [460…
```
y_pred_DT = DT.predict(X_test)
y_pred_DT
```

```
Out[460…  array([1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
                 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
                 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
                 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
                 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
                 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
                 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
                 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
                 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
                 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                 1], dtype=int64)
```

```
In [463…  print(classification_report(y_test,y_pred_DT))
```

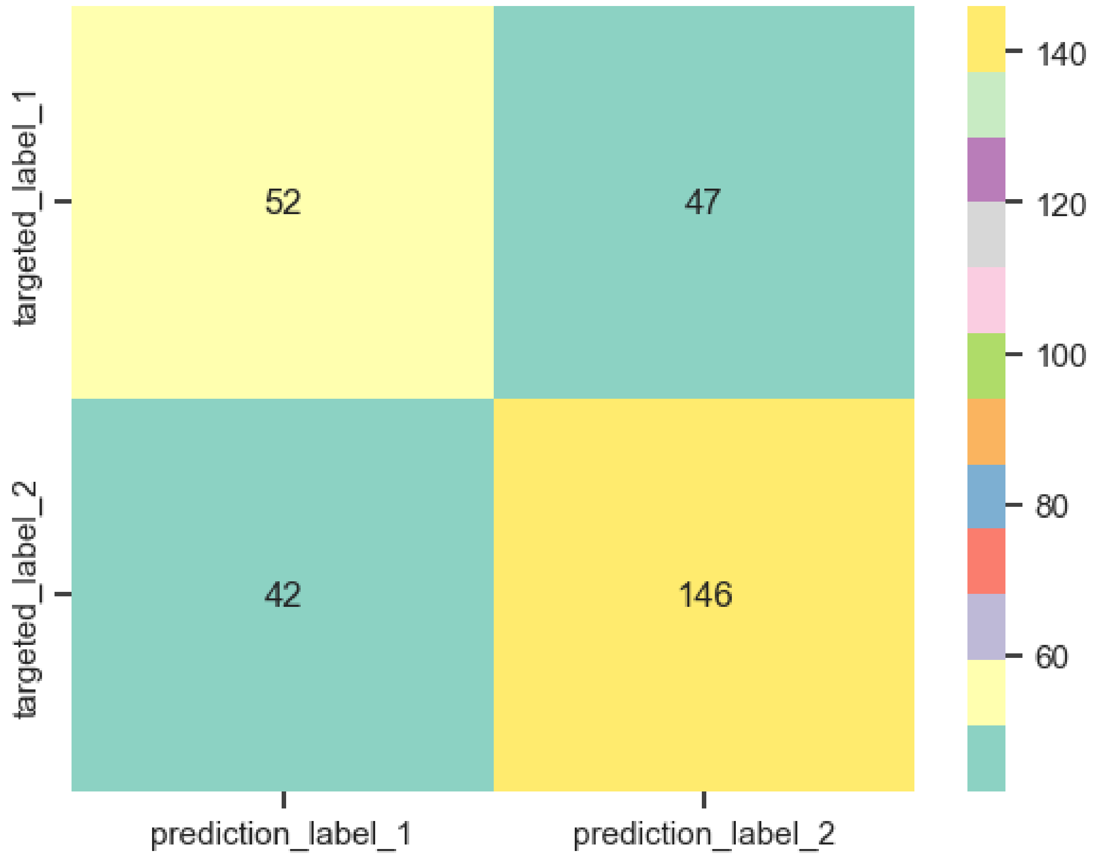
```
              precision    recall  f1-score   support

           0       0.55      0.53      0.54        99
           1       0.76      0.78      0.77       188

    accuracy                           0.69       287
   macro avg       0.65      0.65      0.65       287
weighted avg       0.69      0.69      0.69       287
```

```
In [466…  accuracy = accuracy_score(y_test, y_pred_DT)
          print(f"DecisionTreeClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
DecisionTreeClassifier Model Accuracy: 68.99%
```

```
In [475…  cm = confusion_matrix(y_test, y_pred_DT)
          ax = sns.heatmap(cm, annot=True, fmt='d', cmap="Set3")
          ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
          ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

```
Out[475…  [Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



DecisionTreeClassifier Model Accuracy: 68.99%

RandomForestClassifier

```
In [484…
rnf = RandomForestClassifier()
rnf.fit(X_train,y_train)
```

```
Out[484…
▾ RandomForestClassifier
RandomForestClassifier()
```

```
In [485…  y_pred_rnf = rnf.predict(X_test)
          y_pred_rnf
```

```
Out[485…  array([1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
                 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
                 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
                 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
                 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1], dtype=int64)
```

```
In [486…  print(classification_report(y_test,y_pred_rnf))
```

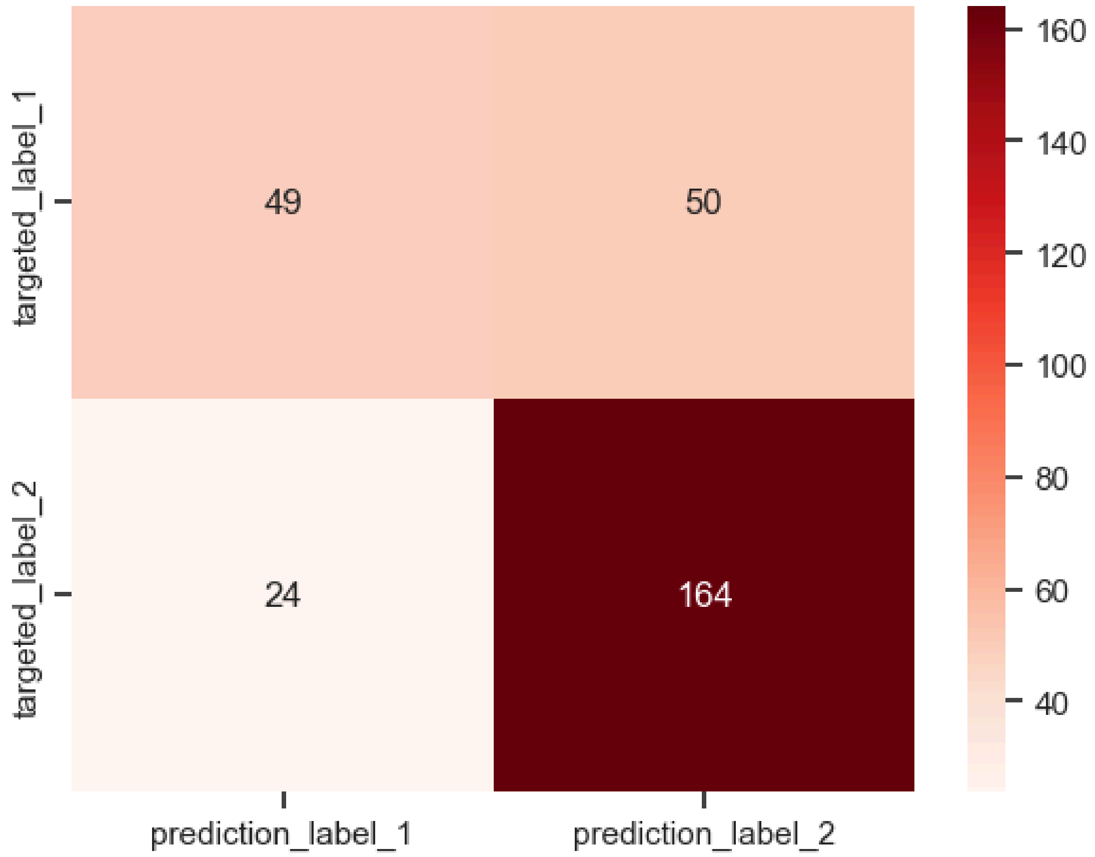
```
              precision    recall  f1-score   support

           0       0.67      0.49      0.57        99
           1       0.77      0.87      0.82       188

    accuracy                           0.74       287
   macro avg       0.72      0.68      0.69       287
weighted avg       0.73      0.74      0.73       287
```

```
In [499…  accuracy = accuracy_score(y_test, y_pred_rnf)
          print(f"RandomForestClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
RandomForestClassifier Model Accuracy: 74.22%
```

```
In [493…  cm = confusion_matrix(y_test, y_pred_rnf)
          ax = sns.heatmap(cm, annot=True, fmt='d', cmap="Reds")
          ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
          ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

```
Out[493…  [Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



RandomForestClassifier Model Accuracy: 74.22%

K-Nearest Neighbors (KNN)

In [511…
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
```

Out[511…
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
In [513…  y_pred_knn = knn.predict(X_test)
          y_pred_knn
```

```
Out[513…  array([1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
                 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
                 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
                 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
                 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
                 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1], dtype=int64)
```

```
In [515…  print(classification_report(y_test,y_pred_knn))
```

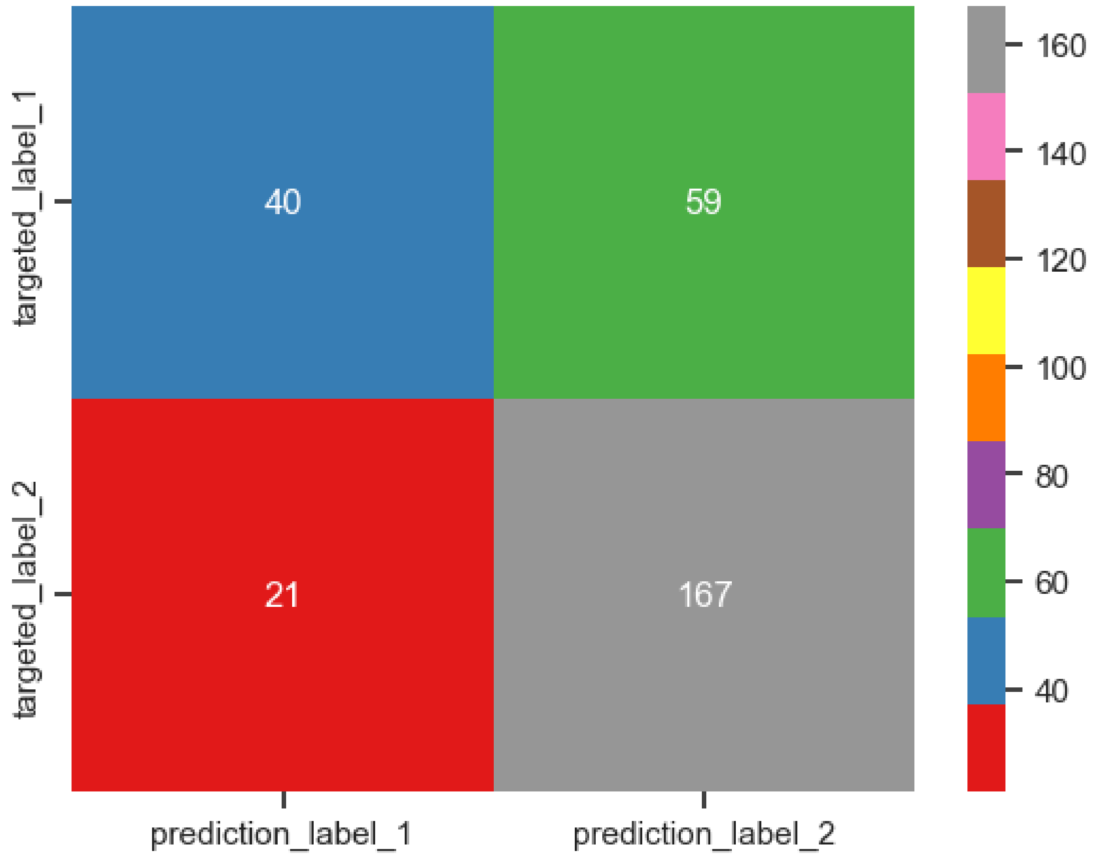
```
              precision    recall  f1-score   support

           0       0.66      0.40      0.50        99
           1       0.74      0.89      0.81       188

    accuracy                           0.72       287
   macro avg       0.70      0.65      0.65       287
weighted avg       0.71      0.72      0.70       287
```

```
In [519…  accuracy = accuracy_score(y_test, y_pred_knn)
          print(f"KNeighborsClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
KNeighborsClassifier Model Accuracy: 72.13%
```

```
In [530…  cm = confusion_matrix(y_test, y_pred_knn)
          ax = sns.heatmap(cm, annot=True, fmt='d', cmap="Set1")
          ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
          ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

```
Out[530…  [Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



KNeighborsClassifier Model Accuracy: 72.13%

In [536…
```
df3_X = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\New folder\archive\descriptors_output.csv")
```

In [539…
```
df3_X
```

Out[539…

	Name	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	Pubche
0	CHEMBL130478	1	1	0	0	0	0	0	0	
1	CHEMBL336538	1	1	1	0	0	0	0	0	
2	CHEMBL339995	1	1	1	0	0	0	0	0	
3	CHEMBL341437	1	1	1	0	0	0	0	0	
4	CHEMBL130098	1	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
6941	CHEMBL253998	1	1	1	0	0	0	0	0	
6942	CHEMBL502	1	1	1	0	0	0	0	0	
6943	CHEMBL3085398	1	1	1	0	0	0	0	0	
6944	CHEMBL13045	1	1	1	0	0	0	0	0	
6945	CHEMBL417799	1	1	0	0	0	0	0	0	

6946 rows × 882 columns

In [542…

```
df3_X = df3_X.drop(columns=['Name'])
df3_X
```

```
Out[542…
```

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	Pubcheml
0	1	1	0	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	0	
2	1	1	1	0	0	0	0	0	0	
3	1	1	1	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
6941	1	1	1	0	0	0	0	0	0	
6942	1	1	1	0	0	0	0	0	0	
6943	1	1	1	0	0	0	0	0	0	
6944	1	1	1	0	0	0	0	0	0	
6945	1	1	0	0	0	0	0	0	0	

6946 rows × 881 columns

```
In [565…
```

```
Y = df3['pIC50']
Y
```

Out[565…
```
0      6.49
1      5.00
2      5.00
3      5.00
4      6.09
       ...
951    6.68
952    6.96
953    7.54
954    4.00
955    4.52
Name: pIC50, Length: 956, dtype: float64
```

In [567…
```python
dataset3 = pd.concat([df3_X,df3_Y], axis=1)
dataset3
```

Out[567…

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	Pubcheml
0	1	1	0	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	0	
2	1	1	1	0	0	0	0	0	0	
3	1	1	1	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
6941	1	1	1	0	0	0	0	0	0	
6942	1	1	1	0	0	0	0	0	0	
6943	1	1	1	0	0	0	0	0	0	
6944	1	1	1	0	0	0	0	0	0	
6945	1	1	0	0	0	0	0	0	0	

6946 rows × 882 columns

```
In [569…  from sklearn.model_selection import train_test_split
          import lazypredict
          from lazypredict.Supervised import LazyRegressor

In [570…  X.shape

Out[570…  (956, 133)

In [572…  # Remove low variance features
          from sklearn.feature_selection import VarianceThreshold
          selection = VarianceThreshold(threshold=(.8 * (1 - .8)))
          X = selection.fit_transform(X)
          X.shape

Out[572…  (956, 133)

In [575…  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

In [597…  print("X_train shape:", X_train.shape)
          print("Y_train shape:", Y_train.shape)

         X_train shape: (764, 133)
         Y_train shape: (764,)

In [603…  from sklearn.impute import SimpleImputer
          import numpy as np

          # Convert Y_train to NumPy array and reshape
          imputer = SimpleImputer(strategy="median")
          Y_train = imputer.fit_transform(Y_train.values.reshape(-1, 1)).ravel()

In [605…  clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
          models, predictions = clf.fit(X_train, X_test, Y_train, Y_test)

          print(models)

          98%|█████████▊| 41/42 [00:33<00:00,  2.91it/s]
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.002196 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 399
[LightGBM] [Info] Number of data points in the train set: 764, number of used features: 133
[LightGBM] [Info] Start training from score 6.471968
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
100%|██████████| 42/42 [00:33<00:00,  1.25it/s]
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
                                          Adjusted R-Squared  \
Model
GradientBoostingRegressor                              -1.82
RandomForestRegressor                                  -1.89
NuSVR                                                  -1.91
BaggingRegressor                                       -1.94
LassoCV                                                -1.94
ElasticNetCV                                           -1.95
HistGradientBoostingRegressor                          -1.95
PoissonRegressor                                       -1.97
LassoLarsIC                                            -1.99
KNeighborsRegressor                                    -2.00
LGBMRegressor                                          -2.00
BayesianRidge                                          -2.00
TweedieRegressor                                       -2.02
MLPRegressor                                           -2.02
GammaRegressor                                         -2.02
SVR                                                    -2.02
AdaBoostRegressor                                      -2.03
LassoLarsCV                                            -2.05
RidgeCV                                                -2.10
OrthogonalMatchingPursuit                              -2.14
OrthogonalMatchingPursuitCV                            -2.14
LarsCV                                                 -2.16
Ridge                                                  -2.18
LinearRegression                                       -2.20
TransformedTargetRegressor                             -2.20
SGDRegressor                                           -2.22
XGBRegressor                                           -2.22
QuantileRegressor                                      -2.29
```

```
DummyRegressor                                     -2.30
LassoLars                                          -2.30
Lasso                                              -2.30
ElasticNet                                         -2.30
ExtraTreesRegressor                                -2.40
DecisionTreeRegressor                              -2.40
HuberRegressor                                     -2.42
ExtraTreeRegressor                                 -2.50
LinearSVR                                          -2.71
PassiveAggressiveRegressor                         -6.39
GaussianProcessRegressor                          -15.44
KernelRidge                                       -54.35
Lars                                -101317328680848368.00
RANSACRegressor            -2608506083726857186207334 4.00

                                              R-Squared            RMSE  \
Model
GradientBoostingRegressor                          0.15            1.53
RandomForestRegressor                              0.12            1.55
NuSVR                                              0.12            1.56
BaggingRegressor                                   0.11            1.56
LassoCV                                            0.11            1.56
ElasticNetCV                                       0.11            1.56
HistGradientBoostingRegressor                      0.10            1.57
PoissonRegressor                                   0.10            1.57
LassoLarsIC                                        0.09            1.58
KNeighborsRegressor                                0.09            1.58
LGBMRegressor                                      0.09            1.58
BayesianRidge                                      0.09            1.58
TweedieRegressor                                   0.08            1.58
MLPRegressor                                       0.08            1.58
GammaRegressor                                     0.08            1.58
SVR                                                0.08            1.59
AdaBoostRegressor                                  0.08            1.59
LassoLarsCV                                        0.07            1.59
RidgeCV                                            0.06            1.61
OrthogonalMatchingPursuit                          0.05            1.62
OrthogonalMatchingPursuitCV                        0.05            1.62
LarsCV                                             0.04            1.62
Ridge                                              0.03            1.63
LinearRegression                                   0.03            1.63
TransformedTargetRegressor                         0.03            1.63
```

```
SGDRegressor                                             0.02             1.63
XGBRegressor                                             0.02             1.64
QuantileRegressor                                       -0.00             1.65
DummyRegressor                                          -0.00             1.65
LassoLars                                               -0.00             1.65
Lasso                                                   -0.00             1.65
ElasticNet                                              -0.00             1.65
ExtraTreesRegressor                                     -0.03             1.68
DecisionTreeRegressor                                   -0.03             1.68
HuberRegressor                                          -0.04             1.69
ExtraTreeRegressor                                      -0.06             1.71
LinearSVR                                               -0.13             1.76
PassiveAggressiveRegressor                              -1.25             2.48
GaussianProcessRegressor                                -3.99             3.70
KernelRidge                                            -15.81             6.78
Lars                                   -30766518657011544.00     290138521.39
RANSACRegressor                -7921117950584173300285440.00 4655426293023.44

                               Time Taken
Model
GradientBoostingRegressor            0.91
RandomForestRegressor                2.23
NuSVR                                0.16
BaggingRegressor                     0.25
LassoCV                              6.30
ElasticNetCV                         8.45
HistGradientBoostingRegressor        1.95
PoissonRegressor                     0.11
LassoLarsIC                          0.30
KNeighborsRegressor                  0.06
LGBMRegressor                        0.56
BayesianRidge                        0.14
TweedieRegressor                     0.07
MLPRegressor                         3.77
GammaRegressor                       0.07
SVR                                  0.22
AdaBoostRegressor                    0.28
LassoLarsCV                          0.16
RidgeCV                              0.24
OrthogonalMatchingPursuit            0.05
OrthogonalMatchingPursuitCV          0.10
LarsCV                               0.65
```

```
Ridge                                 0.05
LinearRegression                      0.16
TransformedTargetRegressor            0.20
SGDRegressor                          0.05
XGBRegressor                          0.41
QuantileRegressor                     0.36
DummyRegressor                        0.03
LassoLars                             0.04
Lasso                                 0.04
ElasticNet                            0.04
ExtraTreesRegressor                   2.61
DecisionTreeRegressor                 0.06
HuberRegressor                        0.29
ExtraTreeRegressor                    0.06
LinearSVR                             0.25
PassiveAggressiveRegressor            0.04
GaussianProcessRegressor              0.27
KernelRidge                           0.08
Lars                                  0.15
RANSACRegressor                       1.28
```

In [613…

```python
from sklearn.datasets import make_regression

X, Y = make_regression(n_samples=500, n_features=10, noise=0.1, random_state=42)
# Ensure Y is 1D
Y = np.ravel(Y)
# Split Data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
# Check Data Validity
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
# Initialize LazyRegressor
clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
# Train & Evaluate Models
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)
# Display Results
if models_test.empty:
 print("No models were evaluated! Check the data formatting.")
else:
 print(models_test)
 models_test.to_csv("model_comparison.csv") # Save results if needed
```

```
(400, 10) (100, 10) (400,) (100,)
```

```
 98%|█████████▊| 41/42 [00:06<00:00,  6.49it/s]
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000540 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1337
[LightGBM] [Info] Number of data points in the train set: 400, number of used features: 10
[LightGBM] [Info] Start training from score 2.408682
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100%|██████████| 42/42 [00:06<00:00,  6.64it/s]
'tuple' object has no attribute '__name__'
Invalid Regressor(s)
100%|██████████| 42/42 [00:06<00:00,  6.02it/s]
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000289 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1337
[LightGBM] [Info] Number of data points in the train set: 400, number of used features: 10
[LightGBM] [Info] Start training from score 2.408682
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100%|██████████| 42/42 [00:06<00:00,  6.05it/s]
```

```
                               Adjusted R-Squared  R-Squared   RMSE  \
Model                                                               
BayesianRidge                                1.00       1.00   0.10   
LassoLarsCV                                  1.00       1.00   0.10   
LinearRegression                             1.00       1.00   0.10   
TransformedTargetRegressor                   1.00       1.00   0.10   
LarsCV                                       1.00       1.00   0.10   
Lars                                         1.00       1.00   0.10   
RANSACRegressor                              1.00       1.00   0.10   
LassoLarsIC                                  1.00       1.00   0.10   
HuberRegressor                               1.00       1.00   0.10   
SGDRegressor                                 1.00       1.00   0.10   
RidgeCV                                      1.00       1.00   0.11   
PassiveAggressiveRegressor                   1.00       1.00   0.11   
LinearSVR                                    1.00       1.00   0.11   
LassoCV                                      1.00       1.00   0.29   
Ridge                                        1.00       1.00   0.38   
KernelRidge                                  1.00       1.00   2.39   
Lasso                                        1.00       1.00   3.12   
LassoLars                                    1.00       1.00   3.12   
ElasticNetCV                                 0.99       0.99  12.09   
LGBMRegressor                                0.89       0.90  44.70   
GradientBoostingRegressor                    0.87       0.89  47.40   
HistGradientBoostingRegressor                0.87       0.89  47.48   
OrthogonalMatchingPursuitCV                  0.87       0.88  48.06   
ElasticNet                                   0.86       0.88  49.50   
ExtraTreesRegressor                          0.81       0.83  57.80   
RandomForestRegressor                        0.77       0.79  63.68   
XGBRegressor                                 0.76       0.78  65.73   
KNeighborsRegressor                          0.75       0.77  66.94   
BaggingRegressor                             0.73       0.76  69.17   
TweedieRegressor                             0.71       0.74  72.34   
AdaBoostRegressor                            0.67       0.70  76.75   
MLPRegressor                                 0.53       0.58  91.02   
DecisionTreeRegressor                        0.37       0.43 105.87   
GaussianProcessRegressor                     0.31       0.38 110.89   
OrthogonalMatchingPursuit                    0.26       0.34 114.48   
ExtraTreeRegressor                           0.13       0.21 124.60   
SVR                                         -0.00       0.10 133.54   
NuSVR                                       -0.03       0.08 135.16   
QuantileRegressor                           -0.13      -0.02 141.74   
DummyRegressor                              -0.13      -0.02 141.78   
```

```
                               Time Taken
Model                                    
BayesianRidge                        0.03
LassoLarsCV                          0.06
LinearRegression                     0.02
TransformedTargetRegressor           0.03
LarsCV                               0.07
Lars                                 0.03
RANSACRegressor                      0.03
LassoLarsIC                          0.03
HuberRegressor                       0.05
SGDRegressor                         0.02
RidgeCV                              0.02
PassiveAggressiveRegressor           0.02
LinearSVR                            0.04
LassoCV                              0.19
Ridge                                0.02
KernelRidge                          0.03
Lasso                                0.02
LassoLars                            0.03
ElasticNetCV                         0.22
LGBMRegressor                        0.23
GradientBoostingRegressor            0.61
HistGradientBoostingRegressor        0.76
OrthogonalMatchingPursuitCV          0.03
ElasticNet                           0.02
ExtraTreesRegressor                  0.56
RandomForestRegressor                1.11
XGBRegressor                         0.42
KNeighborsRegressor                  0.03
BaggingRegressor                     0.15
TweedieRegressor                     0.03
AdaBoostRegressor                    0.37
MLPRegressor                         1.26
DecisionTreeRegressor                0.03
GaussianProcessRegressor             0.05
OrthogonalMatchingPursuit            0.02
ExtraTreeRegressor                   0.03
SVR                                  0.05
NuSVR                                0.05
```

```
QuantileRegressor                        0.06
DummyRegressor                           0.02
```

In [615…

```
predictions_test
```

Out[615…

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
BayesianRidge	1.00	1.00	0.10	0.03
LassoLarsCV	1.00	1.00	0.10	0.06
LinearRegression	1.00	1.00	0.10	0.02
TransformedTargetRegressor	1.00	1.00	0.10	0.03
LarsCV	1.00	1.00	0.10	0.07
Lars	1.00	1.00	0.10	0.03
RANSACRegressor	1.00	1.00	0.10	0.03
LassoLarsIC	1.00	1.00	0.10	0.03
HuberRegressor	1.00	1.00	0.10	0.05
SGDRegressor	1.00	1.00	0.10	0.02
RidgeCV	1.00	1.00	0.11	0.02
PassiveAggressiveRegressor	1.00	1.00	0.11	0.02
LinearSVR	1.00	1.00	0.11	0.04
LassoCV	1.00	1.00	0.29	0.19
Ridge	1.00	1.00	0.38	0.02
KernelRidge	1.00	1.00	2.39	0.03
Lasso	1.00	1.00	3.12	0.02
LassoLars	1.00	1.00	3.12	0.03
ElasticNetCV	0.99	0.99	12.09	0.22
LGBMRegressor	0.89	0.90	44.70	0.23
GradientBoostingRegressor	0.87	0.89	47.40	0.61

	Adjusted R-Squared	R-Squared	RMSE	Time Taken
Model				
HistGradientBoostingRegressor	0.87	0.89	47.48	0.76
OrthogonalMatchingPursuitCV	0.87	0.88	48.06	0.03
ElasticNet	0.86	0.88	49.50	0.02
ExtraTreesRegressor	0.81	0.83	57.80	0.56
RandomForestRegressor	0.77	0.79	63.68	1.11
XGBRegressor	0.76	0.78	65.73	0.42
KNeighborsRegressor	0.75	0.77	66.94	0.03
BaggingRegressor	0.73	0.76	69.17	0.15
TweedieRegressor	0.71	0.74	72.34	0.03
AdaBoostRegressor	0.67	0.70	76.75	0.37
MLPRegressor	0.53	0.58	91.02	1.26
DecisionTreeRegressor	0.37	0.43	105.87	0.03
GaussianProcessRegressor	0.31	0.38	110.89	0.05
OrthogonalMatchingPursuit	0.26	0.34	114.48	0.02
ExtraTreeRegressor	0.13	0.21	124.60	0.03
SVR	-0.00	0.10	133.54	0.05
NuSVR	-0.03	0.08	135.16	0.05
QuantileRegressor	-0.13	-0.02	141.74	0.06
DummyRegressor	-0.13	-0.02	141.78	0.02

In [618…

```
predictions_test
```

Out[618…

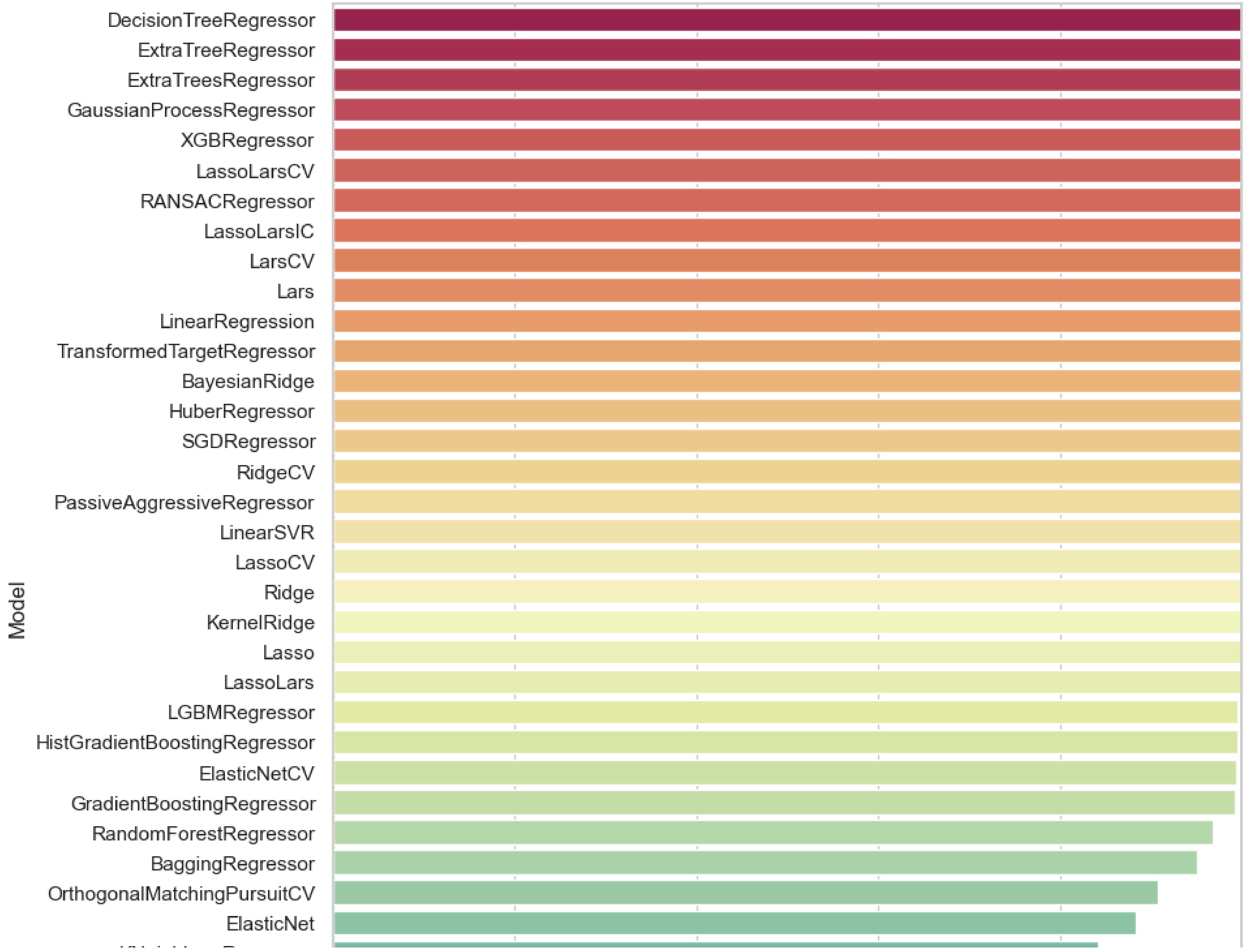
Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
BayesianRidge	1.00	1.00	0.10	0.03
LassoLarsCV	1.00	1.00	0.10	0.06
LinearRegression	1.00	1.00	0.10	0.02
TransformedTargetRegressor	1.00	1.00	0.10	0.03
LarsCV	1.00	1.00	0.10	0.07
Lars	1.00	1.00	0.10	0.03
RANSACRegressor	1.00	1.00	0.10	0.03
LassoLarsIC	1.00	1.00	0.10	0.03
HuberRegressor	1.00	1.00	0.10	0.05
SGDRegressor	1.00	1.00	0.10	0.02
RidgeCV	1.00	1.00	0.11	0.02
PassiveAggressiveRegressor	1.00	1.00	0.11	0.02
LinearSVR	1.00	1.00	0.11	0.04
LassoCV	1.00	1.00	0.29	0.19
Ridge	1.00	1.00	0.38	0.02
KernelRidge	1.00	1.00	2.39	0.03
Lasso	1.00	1.00	3.12	0.02
LassoLars	1.00	1.00	3.12	0.03
ElasticNetCV	0.99	0.99	12.09	0.22
LGBMRegressor	0.89	0.90	44.70	0.23
GradientBoostingRegressor	0.87	0.89	47.40	0.61

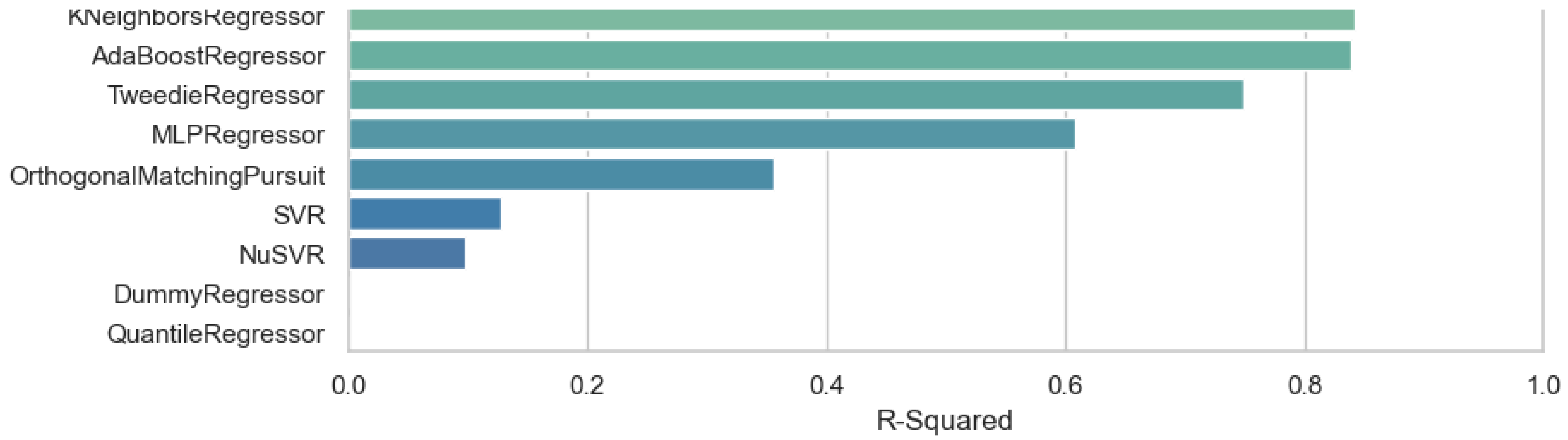
	Adjusted R-Squared	R-Squared	RMSE	Time Taken
Model				
HistGradientBoostingRegressor	0.87	0.89	47.48	0.76
OrthogonalMatchingPursuitCV	0.87	0.88	48.06	0.03
ElasticNet	0.86	0.88	49.50	0.02
ExtraTreesRegressor	0.81	0.83	57.80	0.56
RandomForestRegressor	0.77	0.79	63.68	1.11
XGBRegressor	0.76	0.78	65.73	0.42
KNeighborsRegressor	0.75	0.77	66.94	0.03
BaggingRegressor	0.73	0.76	69.17	0.15
TweedieRegressor	0.71	0.74	72.34	0.03
AdaBoostRegressor	0.67	0.70	76.75	0.37
MLPRegressor	0.53	0.58	91.02	1.26
DecisionTreeRegressor	0.37	0.43	105.87	0.03
GaussianProcessRegressor	0.31	0.38	110.89	0.05
OrthogonalMatchingPursuit	0.26	0.34	114.48	0.02
ExtraTreeRegressor	0.13	0.21	124.60	0.03
SVR	-0.00	0.10	133.54	0.05
NuSVR	-0.03	0.08	135.16	0.05
QuantileRegressor	-0.13	-0.02	141.74	0.06
DummyRegressor	-0.13	-0.02	141.78	0.02

Bar plot of R-squared values

In [621…

```
# Set figure size
plt.figure(figsize=(9, 12))
# Set Seaborn theme
sns.set_theme(style="whitegrid")
# Use a gradient color palette (like "viridis", "coolwarm", or "Spectral")
ax = sns.barplot(
 y=predictions_train.index,
 x="R-Squared",
 data=predictions_train,
 palette=sns.color_palette("Spectral", len(predictions_train))
)
# Set X-axis limits
ax.set(xlim=(0, 1))
# Show plot
plt.show()
```


Model
DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
LassoLarsCV
RANSACRegressor
LassoLarsIC
LarsCV
Lars
LinearRegression
TransformedTargetRegressor
BayesianRidge
HuberRegressor
SGDRegressor
RidgeCV
PassiveAggressiveRegressor
LinearSVR
LassoCV
Ridge
KernelRidge
Lasso
LassoLars
LGBMRegressor
HistGradientBoostingRegressor
ElasticNetCV
GradientBoostingRegressor
RandomForestRegressor
BaggingRegressor
OrthogonalMatchingPursuitCV
ElasticNet



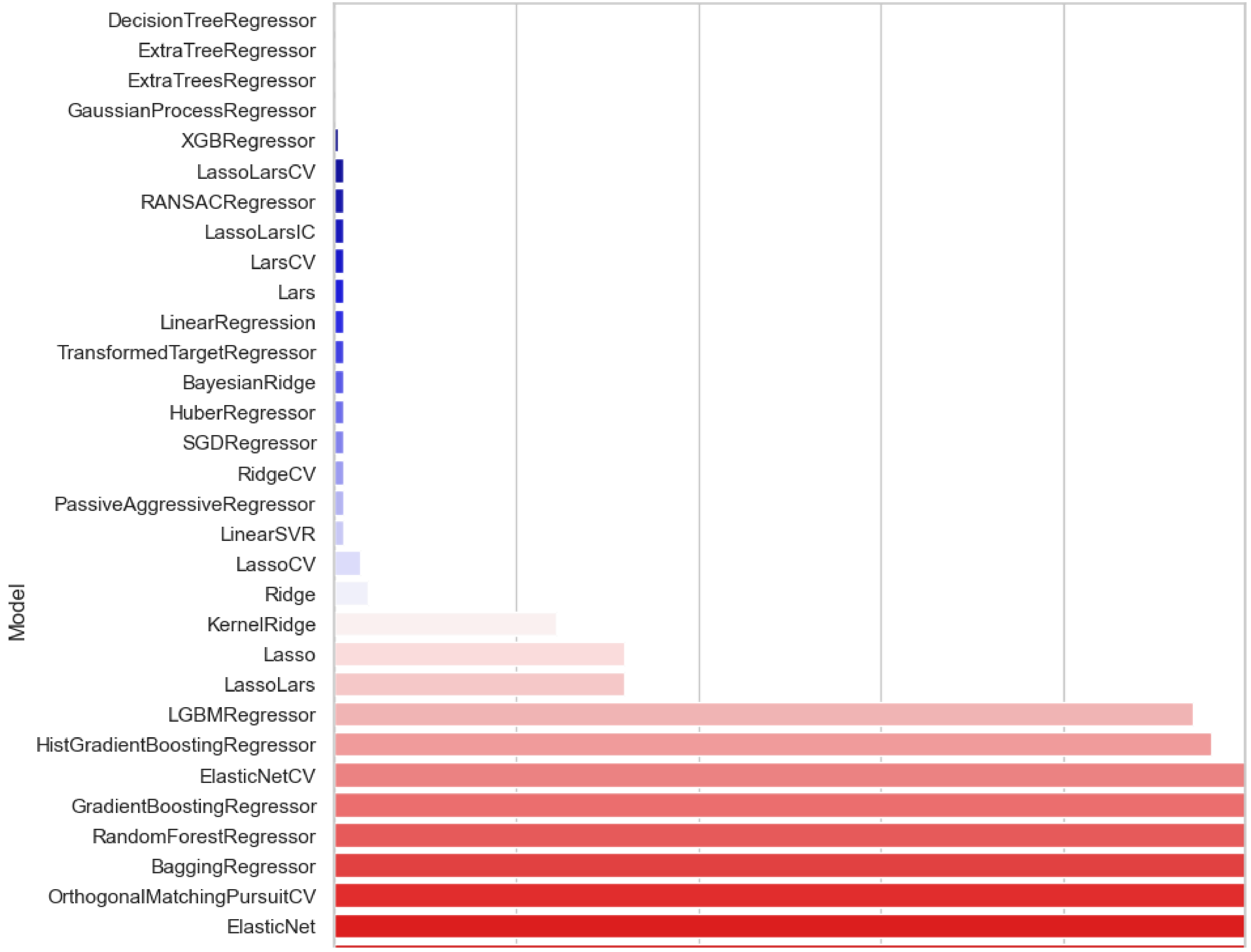
Bar plot of RMSE values

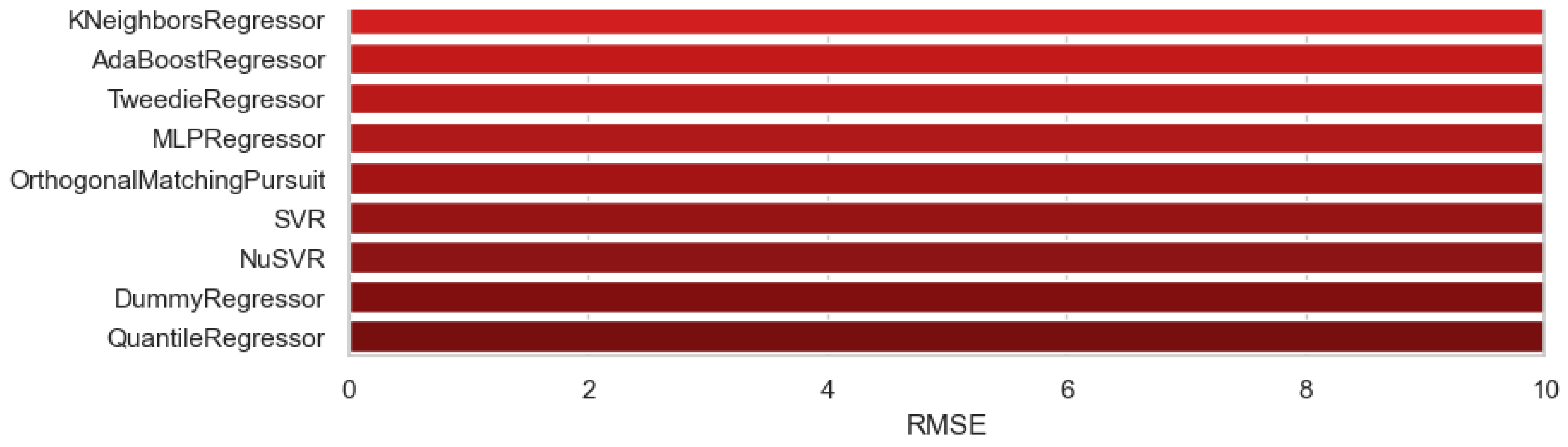
In [631…

```
plt.figure(figsize=(9, 12))
sns.set_theme(style="whitegrid")
ax = sns.barplot(y=predictions_train.index, x="RMSE", data=predictions_train,palette=sns.color_palette("seismic", len(predictions_t
ax.set(xlim=(0, 10))
```

Out[631…

```
[(0.0, 10.0)]
```

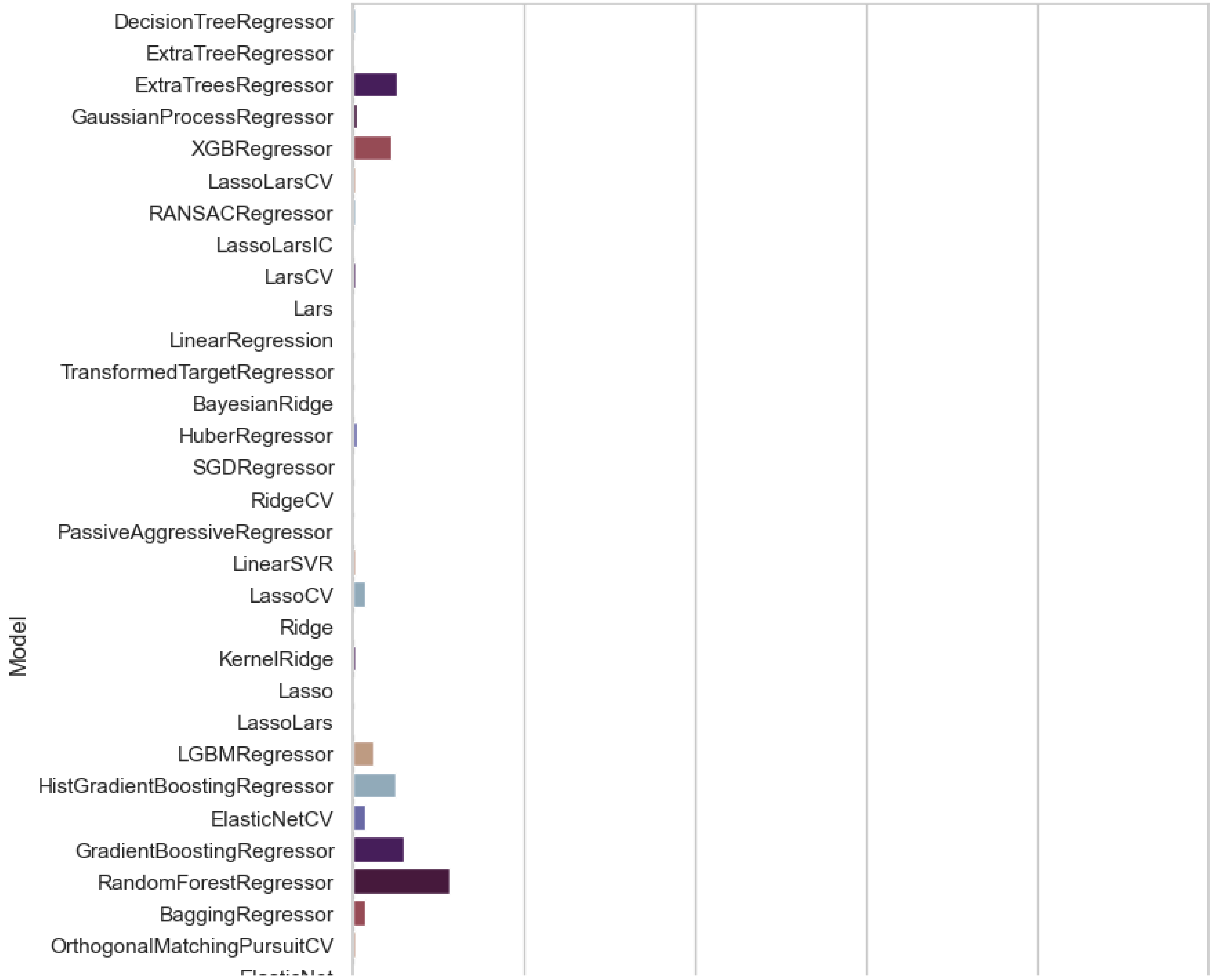

Model
DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
LassoLarsCV
RANSACRegressor
LassoLarsIC
LarsCV
Lars
LinearRegression
TransformedTargetRegressor
BayesianRidge
HuberRegressor
SGDRegressor
RidgeCV
PassiveAggressiveRegressor
LinearSVR
LassoCV
Ridge
KernelRidge
Lasso
LassoLars
LGBMRegressor
HistGradientBoostingRegressor
ElasticNetCV
GradientBoostingRegressor
RandomForestRegressor
BaggingRegressor
OrthogonalMatchingPursuitCV
ElasticNet

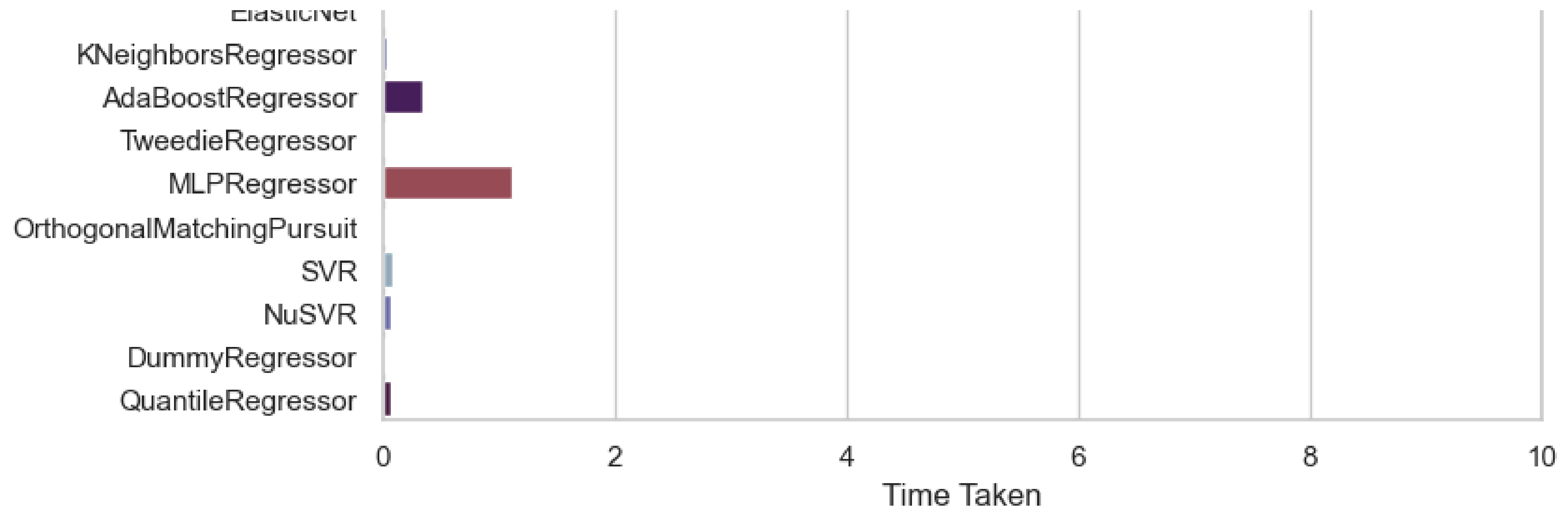


Bar plot of calculation time

```
In [635…
```

```python
plt.figure(figsize=(8, 12))
sns.set_theme(style="whitegrid")

ax = sns.barplot(y=predictions_train.index, x="Time Taken", data=predictions_train,
                 palette=sns.color_palette("twilight"))

ax.set(xlim=(0, 10))
plt.show()
```


Model
DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
LassoLarsCV
RANSACRegressor
LassoLarsIC
LarsCV
Lars
LinearRegression
TransformedTargetRegressor
BayesianRidge
HuberRegressor
SGDRegressor
RidgeCV
PassiveAggressiveRegressor
LinearSVR
LassoCV
Ridge
KernelRidge
Lasso
LassoLars
LGBMRegressor
HistGradientBoostingRegressor
ElasticNetCV
GradientBoostingRegressor
RandomForestRegressor
BaggingRegressor
OrthogonalMatchingPursuitCV


ElasticNet
KNeighborsRegressor
AdaBoostRegressor
TweedieRegressor
MLPRegressor
OrthogonalMatchingPursuit
SVR
NuSVR
DummyRegressor
QuantileRegressor
0
2
4
6
8
10
Time Taken

```
In [ ]:
```