In [5]:
```
import pandas as pd
from chembl_webresource_client.new_client import new_client
```

In [10]:
```
target = new_client.target
target_query = target.search('CHEMBL231')
targets = pd.DataFrame.from_dict(target_query)
targets
```

Out[10]:

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Homo sapiens	Histamine H1 receptor	17.0	False	CHEMBL231	[{'accession': 'P35367', 'component_descriptio...	SINGLE PROTEIN	9606

In [13]:
```
selected_target = targets.target_chembl_id[0]
selected_target
```

Out[13]:
```
'CHEMBL231'
```

In [16]:
```
activity = new_client.activity
res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
```

In [19]:
```
df = pd.DataFrame.from_dict(res)
```

In [22]:
```
print(len(res))
```

```
1300
```

In [44]:
```
df.head()
```

Out[44]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_
0	None	None	146647	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
1	None	None	149038	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
2	None	None	149041	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
3	None	None	150189	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
4	None	None	152698	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	

5 rows × 46 columns

In [46]:
```
df.tail()
```

Out[46]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
1295	{'action_type': 'INHIBITOR', 'description': 'N...	None	25523852	[]	CHEMBL5338203	Inhibition of human H1 receptor	B	None	
1296	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658311	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
1297	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658312	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
1298	None	None	25778417	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465586	Selectivity interaction (BET inhibitor selecti...	B	None	
1299	None	None	25787560	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474422	Selectivity interaction (Histamine H1 receptor...	B	None	

5 rows × 46 columns

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1300 entries, 0 to 1299
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   action_type                 9 non-null      object
 1   activity_comment            848 non-null    object
 2   activity_id                 1300 non-null   int64
 3   activity_properties         1300 non-null   object
 4   assay_chembl_id             1300 non-null   object
 5   assay_description           1300 non-null   object
 6   assay_type                  1300 non-null   object
 7   assay_variant_accession     1 non-null      object
 8   assay_variant_mutation      1 non-null      object
 9   bao_endpoint                1300 non-null   object
 10  bao_format                  1300 non-null   object
 11  bao_label                   1300 non-null   object
 12  canonical_smiles            1267 non-null   object
 13  data_validity_comment       6 non-null      object
 14  data_validity_description   6 non-null      object
 15  document_chembl_id          1300 non-null   object
 16  document_journal            407 non-null    object
 17  document_year               429 non-null    float64
 18  ligand_efficiency           242 non-null    object
 19  molecule_chembl_id          1300 non-null   object
 20  molecule_pref_name          912 non-null    object
 21  parent_molecule_chembl_id   1300 non-null   object
 22  pchembl_value               394 non-null    object
 23  potential_duplicate         1300 non-null   int64
 24  qudt_units                  466 non-null    object
 25  record_id                   1300 non-null   int64
 26  relation                    466 non-null    object
 27  src_id                      1300 non-null   int64
 28  standard_flag               1300 non-null   int64
 29  standard_relation           466 non-null    object
 30  standard_text_value         0 non-null      object
 31  standard_type               1300 non-null   object
 32  standard_units              466 non-null    object
 33  standard_upper_value        0 non-null      object
 34  standard_value              466 non-null    object
 35  target_chembl_id            1300 non-null   object
 36  target_organism             1300 non-null   object
```

```
 37  target_pref_name           1300 non-null   object
 38  target_tax_id              1300 non-null   object
 39  text_value                 0 non-null      object
 40  toid                       0 non-null      object
 41  type                       1300 non-null   object
 42  units                      381 non-null    object
 43  uo_units                   466 non-null    object
 44  upper_value                0 non-null      object
 45  value                      466 non-null    object
dtypes: float64(1), int64(5), object(40)
memory usage: 467.3+ KB
```

In [50]:
```
df.columns
```

Out[50]:
```
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 [53]:
```
df2 = df.dropna(subset=["standard_value", "canonical_smiles"])
```

In [55]:
```
df2
```

Out[55]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
0	None	None	146647	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
1	None	None	149038	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
2	None	None	149041	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
3	None	None	150189	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
4	None	None	152698	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
...	...	...	...	...	...	...	...	...	
1295	{'action_type': 'INHIBITOR', 'description': 'N...	None	25523852	[]	CHEMBL5338203	Inhibition of human H1 receptor	B	None	
1296	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658311	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
1297	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658312	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
1298	None	None	25778417	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465586	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	as
1299	None	None	25787560	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474422	Selectivity interaction (Histamine H1 receptor...	B	None	

466 rows × 46 columns

In [59]:
```
len(df2.canonical_smiles.unique())
```

Out[59]: 377

In [64]:
```
df2_nr = df2.drop_duplicates(subset="canonical_smiles", keep="first").reset_index(drop=True)
```

In [67]:
```
df2_nr
```

Out[67]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	ass
0	None	None	146647	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
1	None	None	149038	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
2	None	None	149041	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
3	None	None	150189	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
4	None	None	152698	[]	CHEMBL692983	Inhibitory activity against human Histamine H1...	B	None	
...	...	...	...	...	...	...	...	...	
372	{'action_type': 'INHIBITOR', 'description': 'N...	None	25523852	[]	CHEMBL5338203	Inhibition of human H1 receptor	B	None	
373	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658311	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
374	{'action_type': 'INHIBITOR', 'description': 'N...	None	25658312	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5372886	Displacement of [3H] pyrilamine from human rec...	B	None	
375	None	None	25778417	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5465586	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	ass
376	None	None	25787560	[{'comments': None, 'relation': None, 'result_...	CHEMBL5474422	Selectivity interaction (Histamine H1 receptor...	B	None	

377 rows × 46 columns

In [70]:
```
df2_nr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 377 entries, 0 to 376
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   action_type                 9 non-null      object 
 1   activity_comment            5 non-null      object 
 2   activity_id                 377 non-null    int64  
 3   activity_properties         377 non-null    object 
 4   assay_chembl_id             377 non-null    object 
 5   assay_description           377 non-null    object 
 6   assay_type                  377 non-null    object 
 7   assay_variant_accession     0 non-null      object 
 8   assay_variant_mutation      0 non-null      object 
 9   bao_endpoint                377 non-null    object 
 10  bao_format                  377 non-null    object 
 11  bao_label                   377 non-null    object 
 12  canonical_smiles            377 non-null    object 
 13  data_validity_comment       4 non-null      object 
 14  data_validity_description   4 non-null      object 
 15  document_chembl_id          377 non-null    object 
 16  document_journal            321 non-null    object 
 17  document_year               333 non-null    float64
 18  ligand_efficiency           203 non-null    object 
 19  molecule_chembl_id          377 non-null    object 
 20  molecule_pref_name          74 non-null     object 
 21  parent_molecule_chembl_id   377 non-null    object 
 22  pchembl_value               310 non-null    object 
 23  potential_duplicate         377 non-null    int64  
 24  qudt_units                  377 non-null    object 
 25  record_id                   377 non-null    int64  
 26  relation                    377 non-null    object 
 27  src_id                      377 non-null    int64  
 28  standard_flag               377 non-null    int64  
 29  standard_relation           377 non-null    object 
 30  standard_text_value         0 non-null      object 
 31  standard_type               377 non-null    object 
 32  standard_units              377 non-null    object 
 33  standard_upper_value        0 non-null      object 
 34  standard_value              377 non-null    object 
 35  target_chembl_id            377 non-null    object 
 36  target_organism             377 non-null    object 
```

```
 37  target_pref_name           377 non-null    object
 38  target_tax_id              377 non-null    object
 39  text_value                 0 non-null      object
 40  toid                       0 non-null      object
 41  type                       377 non-null    object
 42  units                      303 non-null    object
 43  uo_units                   377 non-null    object
 44  upper_value                0 non-null      object
 45  value                      377 non-null    object
dtypes: float64(1), int64(5), object(40)
memory usage: 135.6+ KB
```

Data pre-processing of the bioactivity data

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

In [75]:
```
selection = ['molecule_chembl_id','canonical_smiles','standard_value']
df3 = df2_nr[selection]
df3
```

Out[75]:

	molecule_chembl_id	canonical_smiles	standard_value
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.0
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	2511.89
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	501.19
...	...	...	...
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	8000.0
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	340.0
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	520.0
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	31600.0
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0

377 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 [80]:
```
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 [83]:
```
bioactivity_class = pd.Series(bioactivity_threshold, name='class')
df5 = pd.concat([df3, bioactivity_class], axis=1)
df5
```

Out[83]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93	intermediate
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93	intermediate
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.0	active
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	2511.89	intermediate
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	501.19	active
...	...	...	...	...
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	8000.0	intermediate
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	340.0	active
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	520.0	active
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	31600.0	inactive
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.0	inactive

377 rows × 4 columns

In [88]:
```
df5.to_csv('Histamine H1 receptor_03_bioactivity_data_curated.csv', index=False)
```

In [94]:
```
df = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\Histamine H1 receptor\Histamine H1 receptor_03_bioactivity_data_curated.csv")
df
```

Out[94]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93	intermediate
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93	intermediate
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.00	active
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	2511.89	intermediate
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	501.19	active
...	...	...	...	...
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	8000.00	intermediate
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	340.00	active
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	520.00	active
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	31600.00	inactive
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.00	inactive

377 rows × 4 columns

In [97]:
```
df_no_smiles = df.drop(columns='canonical_smiles')
```

In [100…
```
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 [103…
```
df_clean_smiles = pd.concat([df_no_smiles,smiles], axis=1)
df_clean_smiles
```

Out[103...

	molecule_chembl_id	standard_value	class	canonical_smiles
0	CHEMBL156071	1258.93	intermediate	NCCCCCCCCCCCCCc1c[nH]cn1
1	CHEMBL152122	1258.93	intermediate	NCCCCCCCCCCCCCc1cccnc1
2	CHEMBL24665	1000.00	active	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1
3	CHEMBL157842	2511.89	intermediate	NCCCCCCCCCCCc1cccnc1
4	CHEMBL94249	501.19	active	NCCCCCCCCCCCc1c[nH]cn1
...	...	...	...	...
372	CHEMBL5421982	8000.00	intermediate	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...
373	CHEMBL5398630	340.00	active	[2H]C([2H])(Cc1c[nH]c2ccccc12)N(C)C
374	CHEMBL3183055	520.00	active	CN(C)CCc1c[nH]c2ccccc12
375	CHEMBL2017291	31600.00	inactive	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...
376	CHEMBL3643413	30000.00	inactive	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...

377 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 [108…

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

Calculate descriptors

In [115…

```
# 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 [119…

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

Out[119…

	MW	LogP	NumHDonors	NumHAcceptors
0	265.445000	4.20200	2.0	2.0
1	262.441000	4.48380	1.0	2.0
2	255.243000	2.59670	2.0	2.0
3	234.387000	3.70360	1.0	2.0
4	223.364000	3.03170	2.0	2.0
...	...	...	...	...
372	530.626000	2.97850	1.0	8.0
373	190.286204	2.27200	1.0	1.0
374	188.274000	2.27200	1.0	1.0
375	415.453000	4.16254	1.0	7.0
376	450.465000	2.88450	1.0	7.0

377 rows × 4 columns

Combine DataFrames

In [124…

```
df_lipinski
```

Out[124...

	MW	LogP	NumHDonors	NumHAcceptors
0	265.445000	4.20200	2.0	2.0
1	262.441000	4.48380	1.0	2.0
2	255.243000	2.59670	2.0	2.0
3	234.387000	3.70360	1.0	2.0
4	223.364000	3.03170	2.0	2.0
...	...	...	...	...
372	530.626000	2.97850	1.0	8.0
373	190.286204	2.27200	1.0	1.0
374	188.274000	2.27200	1.0	1.0
375	415.453000	4.16254	1.0	7.0
376	450.465000	2.88450	1.0	7.0

377 rows × 4 columns

In [126...

```
df_lipinski.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 377 entries, 0 to 376
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   MW              377 non-null    float64
 1   LogP            377 non-null    float64
 2   NumHDonors      377 non-null    float64
 3   NumHAcceptors   377 non-null    float64
dtypes: float64(4)
memory usage: 11.9 KB
```

In [128...

```
df_lipinski.describe()
```

Out[128...

	MW	LogP	NumHDonors	NumHAcceptors
count	377.000000	377.000000	377.00000	377.000000
mean	426.848953	4.326096	0.63130	4.883289
std	147.117151	1.459095	0.76796	2.601587
min	111.148000	-0.386300	0.00000	1.000000
25%	312.388000	3.353200	0.00000	3.000000
50%	392.503000	4.057400	0.00000	4.000000
75%	506.002000	5.179800	1.00000	7.000000
max	806.976000	8.715300	4.00000	12.000000

In [131...

```
df
```

Out[131...

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93	intermediate
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93	intermediate
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.00	active
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	2511.89	intermediate
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	501.19	active
...	...	...	...	...
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	8000.00	intermediate
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	340.00	active
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	520.00	active
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	31600.00	inactive
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.00	inactive

377 rows × 4 columns

In [133...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 377 entries, 0 to 376
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  377 non-null    object 
 1   canonical_smiles    377 non-null    object 
 2   standard_value      377 non-null    float64
 3   class               377 non-null    object 
dtypes: float64(1), object(3)
memory usage: 11.9+ KB
```

In [135...

```
df.describe()
```

Out[135…

	standard_value
count	377.000000
mean	5495.012305
std	23852.527737
min	0.421000
25%	30.000000
50%	240.000000
75%	2130.000000
max	316227.770000

combine the 2 DataFrame

In [142…

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

In [145…

```
df_combined
```

Out[145...

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHA
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93	intermediate	265.445000	4.20200	2.0	
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93	intermediate	262.441000	4.48380	1.0	
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.00	active	255.243000	2.59670	2.0	
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	2511.89	intermediate	234.387000	3.70360	1.0	
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	501.19	active	223.364000	3.03170	2.0	
...	...	...	...	...	...	...	...	...
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	8000.00	intermediate	530.626000	2.97850	1.0	
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	340.00	active	190.286204	2.27200	1.0	
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	520.00	active	188.274000	2.27200	1.0	
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	31600.00	inactive	415.453000	4.16254	1.0	
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	30000.00	inactive	450.465000	2.88450	1.0	

377 rows × 8 columns

377 rows × 8 columns

In [147...

```
df_combined.head()
```

Out[147…

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHAcceptors
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	1258.93	intermediate	265.445	4.2020	2.0	2.0
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	1258.93	intermediate	262.441	4.4838	1.0	2.0
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	1000.00	active	255.243	2.5967	2.0	2.0
3	CHEMBL157842	NCCCCCCCCCCc1cccnc1	2511.89	intermediate	234.387	3.7036	1.0	2.0
4	CHEMBL94249	NCCCCCCCCCCc1c[nH]cn1	501.19	active	223.364	3.0317	2.0	2.0

In [149…

```
df_combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 377 entries, 0 to 376
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  377 non-null    object 
 1   canonical_smiles    377 non-null    object 
 2   standard_value      377 non-null    float64
 3   class               377 non-null    object 
 4   MW                  377 non-null    float64
 5   LogP                377 non-null    float64
 6   NumHDonors          377 non-null    float64
 7   NumHAcceptors       377 non-null    float64
dtypes: float64(5), object(3)
memory usage: 23.7+ KB
```

In [151…

```
df_combined.describe()
```

Out[151…

	standard_value	MW	LogP	NumHDonors	NumHAcceptors
count	377.000000	377.000000	377.000000	377.00000	377.000000
mean	5495.012305	426.848953	4.326096	0.63130	4.883289
std	23852.527737	147.117151	1.459095	0.76796	2.601587
min	0.421000	111.148000	-0.386300	0.00000	1.000000
25%	30.000000	312.388000	3.353200	0.00000	3.000000
50%	240.000000	392.503000	4.057400	0.00000	4.000000
75%	2130.000000	506.002000	5.179800	1.00000	7.000000
max	316227.770000	806.976000	8.715300	4.00000	12.000000

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 [159…

```
# 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 [162…
```
df_combined.standard_value.describe()
```

Out[162…
```
count       377.000000
mean       5495.012305
std       23852.527737
min           0.421000
25%          30.000000
50%         240.000000
75%        2130.000000
max      316227.770000
Name: standard_value, dtype: float64
```

In [165…
```
-np.log10( (10**-9)* 100000000 )
```

Out[165…
```
1.0
```

In [167…
```
-np.log10( (10**-9)* 10000000000 )
```

Out[167…
```
-1.0
```

In [175…
```
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 [178…]
```
df_norm = norm_value(df_combined)
df_norm
```

Out[178…]

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	standa
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	intermediate	265.445000	4.20200	2.0	2.0	
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	intermediate	262.441000	4.48380	1.0	2.0	
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	active	255.243000	2.59670	2.0	2.0	
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	intermediate	234.387000	3.70360	1.0	2.0	
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	active	223.364000	3.03170	2.0	2.0	
...	...	...	...	...	...	...	...	
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	intermediate	530.626000	2.97850	1.0	8.0	
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	active	190.286204	2.27200	1.0	1.0	
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	active	188.274000	2.27200	1.0	1.0	
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453000	4.16254	1.0	7.0	
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465000	2.88450	1.0	7.0	

377 rows × 8 columns

In [181…]
```
df_norm.standard_value_norm
```

Out[181…

```
0       1258.93
1       1258.93
2       1000.00
3       2511.89
4        501.19
         ...
372     8000.00
373      340.00
374      520.00
375    31600.00
376    30000.00
Name: standard_value_norm, Length: 377, dtype: float64
```

In [183…

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

Out[183…

```
count       377.000000
mean       5495.012305
std       23852.527737
min           0.421000
25%          30.000000
50%         240.000000
75%        2130.000000
max      316227.770000
Name: standard_value_norm, dtype: float64
```

In [195…

```
df_final = df_norm.drop('standard_value_norm', axis=1)
df_final
```

Out[195...

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC
0	CHEMBL156071	NCCCCCCCCCCCCCc1c[nH]cn1	intermediate	265.445000	4.20200	2.0	2.0	5.8999
1	CHEMBL152122	NCCCCCCCCCCCCCc1cccnc1	intermediate	262.441000	4.48380	1.0	2.0	5.8999
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	active	255.243000	2.59670	2.0	2.0	6.0000
3	CHEMBL157842	NCCCCCCCCCCCc1cccnc1	intermediate	234.387000	3.70360	1.0	2.0	5.5999
4	CHEMBL94249	NCCCCCCCCCCCc1c[nH]cn1	active	223.364000	3.03170	2.0	2.0	6.2999
...	...	...	...	...	...	...	...	
372	CHEMBL5421982	COC1CCN(c2cc(C(=O)NS(=O)(=O)N(C)C)nc3c2c(C2CCC...	intermediate	530.626000	2.97850	1.0	8.0	5.0969
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	active	190.286204	2.27200	1.0	1.0	6.4685
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	active	188.274000	2.27200	1.0	1.0	6.2839
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453000	4.16254	1.0	7.0	4.5003
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465000	2.88450	1.0	7.0	4.5228

377 rows × 8 columns

In [198...

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

Out[198…

```
count    377.000000
mean       6.590691
std        1.232397
min        3.500000
25%        5.671620
50%        6.619789
75%        7.522879
max        9.375718
Name: pIC50, dtype: float64
```

Removing the 'intermediate' bioactivity class

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

In [204…

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

Out[204...

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC50
2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	active	255.243000	2.59670	2.0	2.0	6.000000
4	CHEMBL94249	NCCCCCCCCCCc1c[nH]cn1	active	223.364000	3.03170	2.0	2.0	6.299998
5	CHEMBL157217	c1nc(CCCCCCCCCCN2CCCC2)c[nH]1	active	277.456000	4.16880	1.0	2.0	6.399997
9	CHEMBL155188	NCCCCCCCCCCCCc1c[nH]cn1	active	251.418000	3.81190	2.0	2.0	6.000000
12	CHEMBL1241	CN(C)CCN(Cc1ccccc1)c1ccccn1	active	255.365000	2.64980	0.0	3.0	7.400008
...	...	...	...	...	...	...	...	...
371	CHEMBL4519018	CN1C2CCCC1CC(Nc1ccccc1Br)C2	active	309.251000	3.87630	1.0	2.0	6.696804
373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	active	190.286204	2.27200	1.0	1.0	6.468521
374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	active	188.274000	2.27200	1.0	1.0	6.283997
375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453000	4.16254	1.0	7.0	4.500313
376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465000	2.88450	1.0	7.0	4.522879

326 rows × 8 columns

In [338...

```
df_2class.to_csv('Histamine H1 receptor_04_bioactivity_data_3class_pIC50.csv')
```

Exploratory Data Analysis (Chemical Space Analysis) via Lipinski descriptors

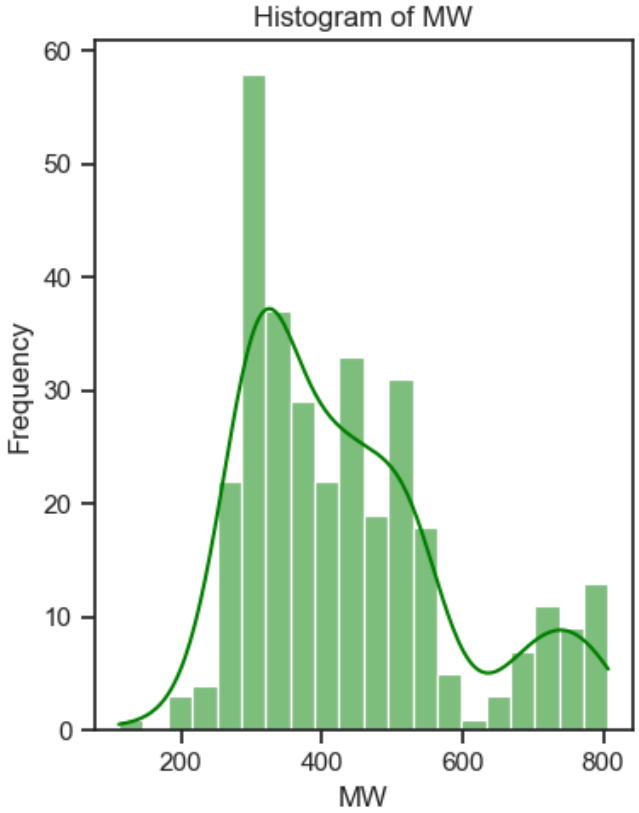
In [213...

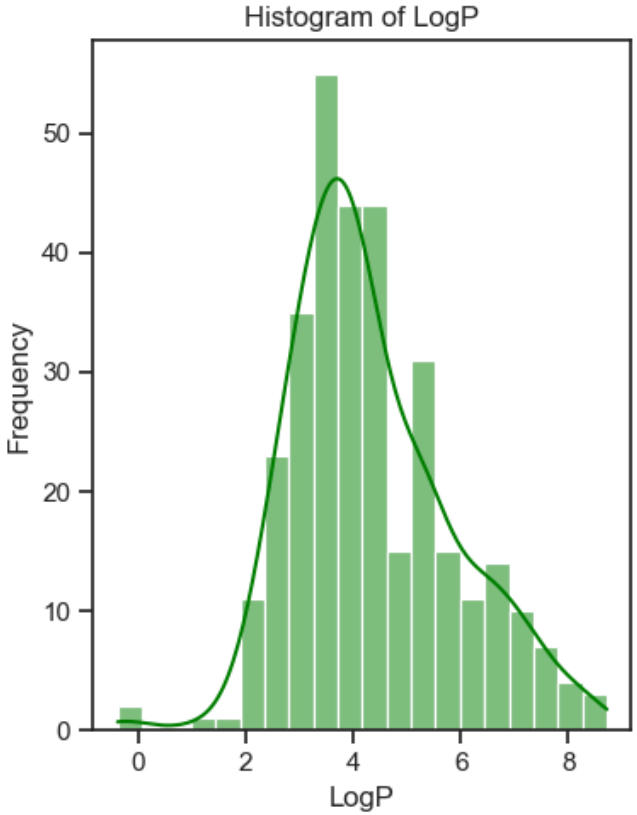
```
import seaborn as sns
sns.set(style='ticks')
import matplotlib.pyplot as plt
```

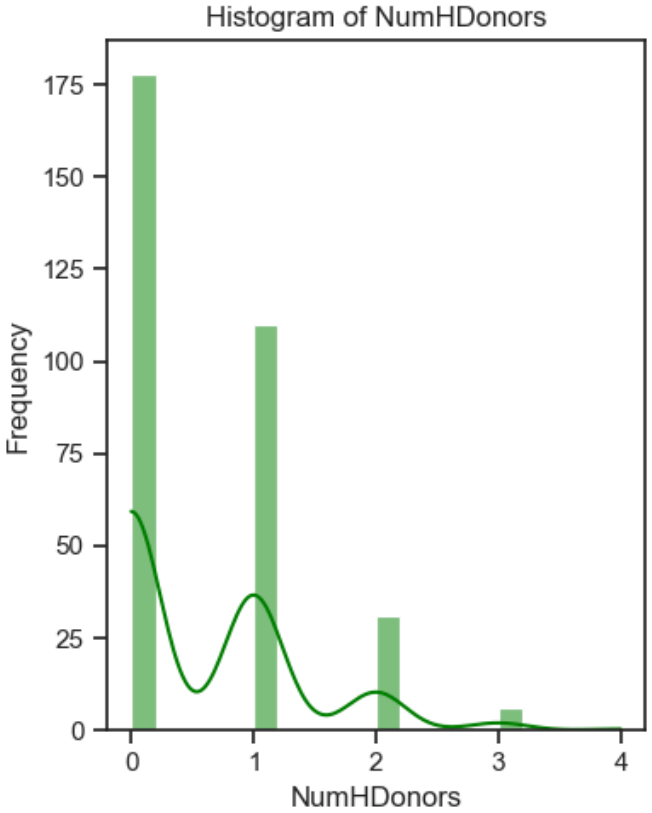
In [223...

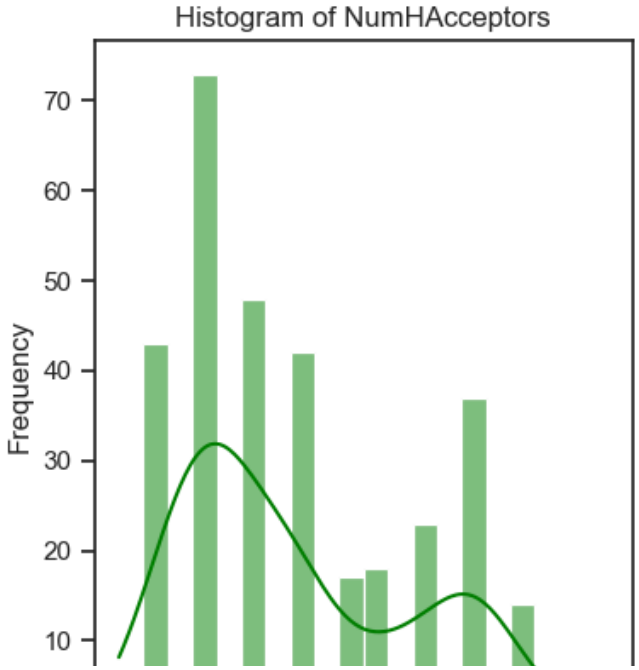
```
plt.figure(figsize=(12, 10))
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors', 'pIC50']
for i, column in enumerate(numeric_columns, 1):
```

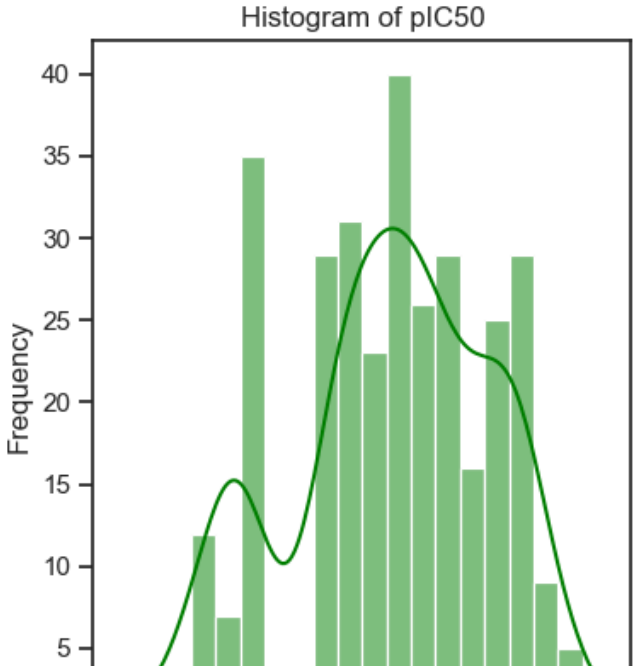
```
 plt.subplot(2, 3, i) # 2 rows and 3 columns for better layout
 sns.histplot(df_2class[column], kde=True, bins=20,color='green')# kde=True for kernel density estimate
 plt.title(f'Histogram of {column}')
 plt.xlabel(column)
 plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```


Histogram of MW
Frequency
MW
0
10
20
30
40
50
60
200
400
600
800


Histogram of LogP
Frequency
LogP
0
10
20
30
40
50
0
2
4
6
8


Histogram of NumHDonors
Frequency
NumHDonors
0
25
50
75
100
125
150
175
0
1
2
3
4

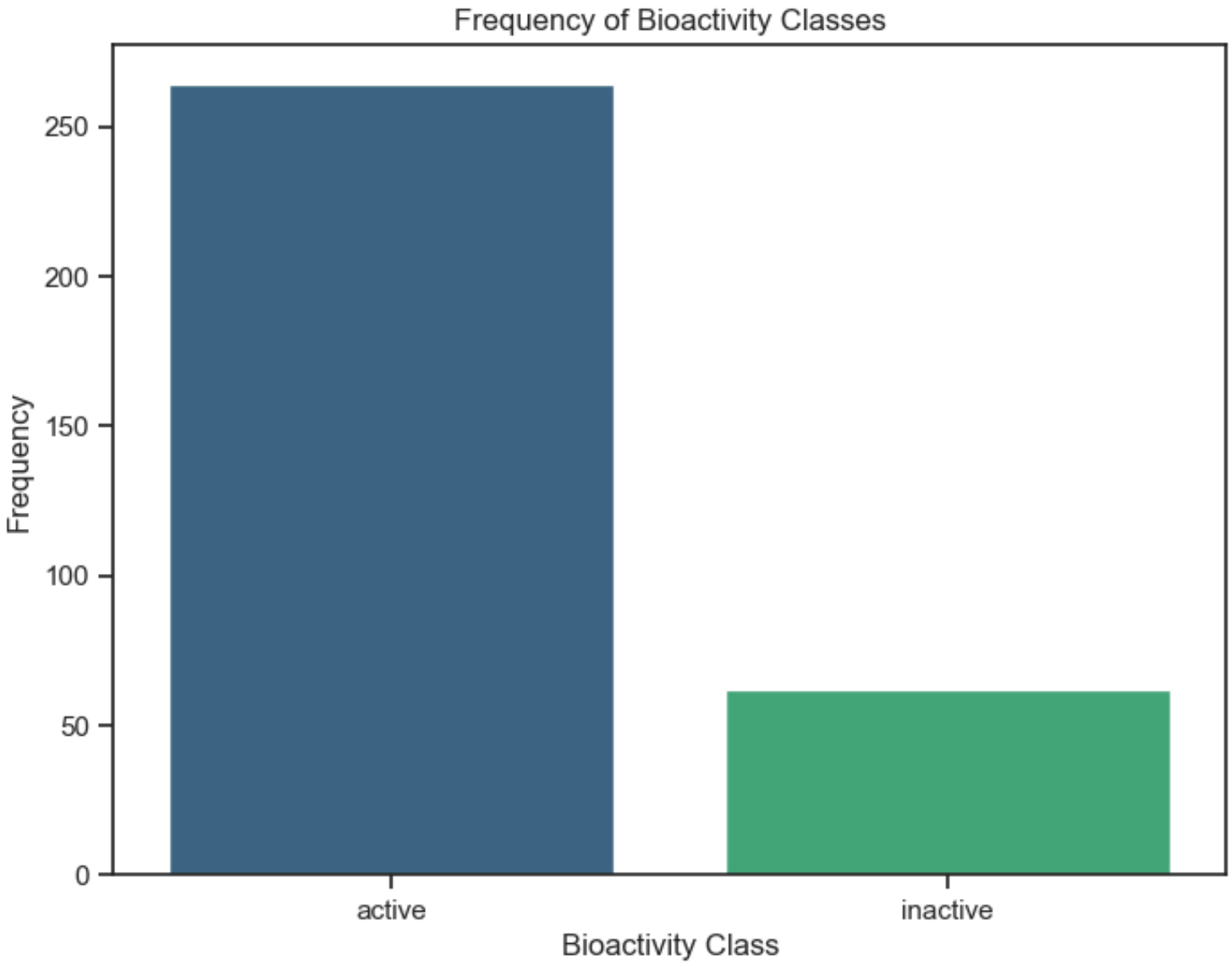

Histogram of NumHAcceptors
Frequency
10
20
30
40
50
60
70


Histogram of pIC50
Frequency
5
10
15
20
25
30
35
40

Frequency of Bioactivity Classes

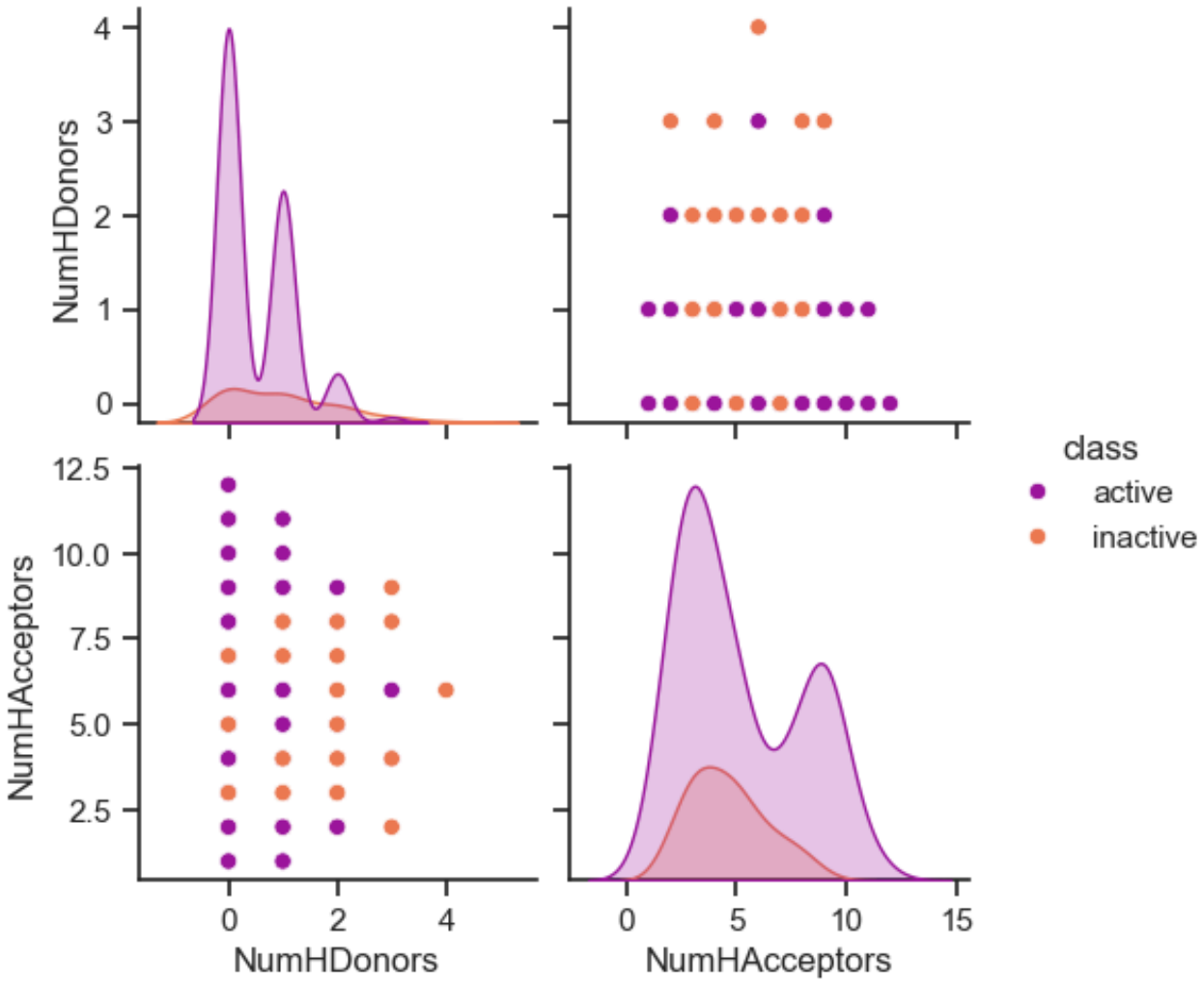
In [230…

```
# 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()
```



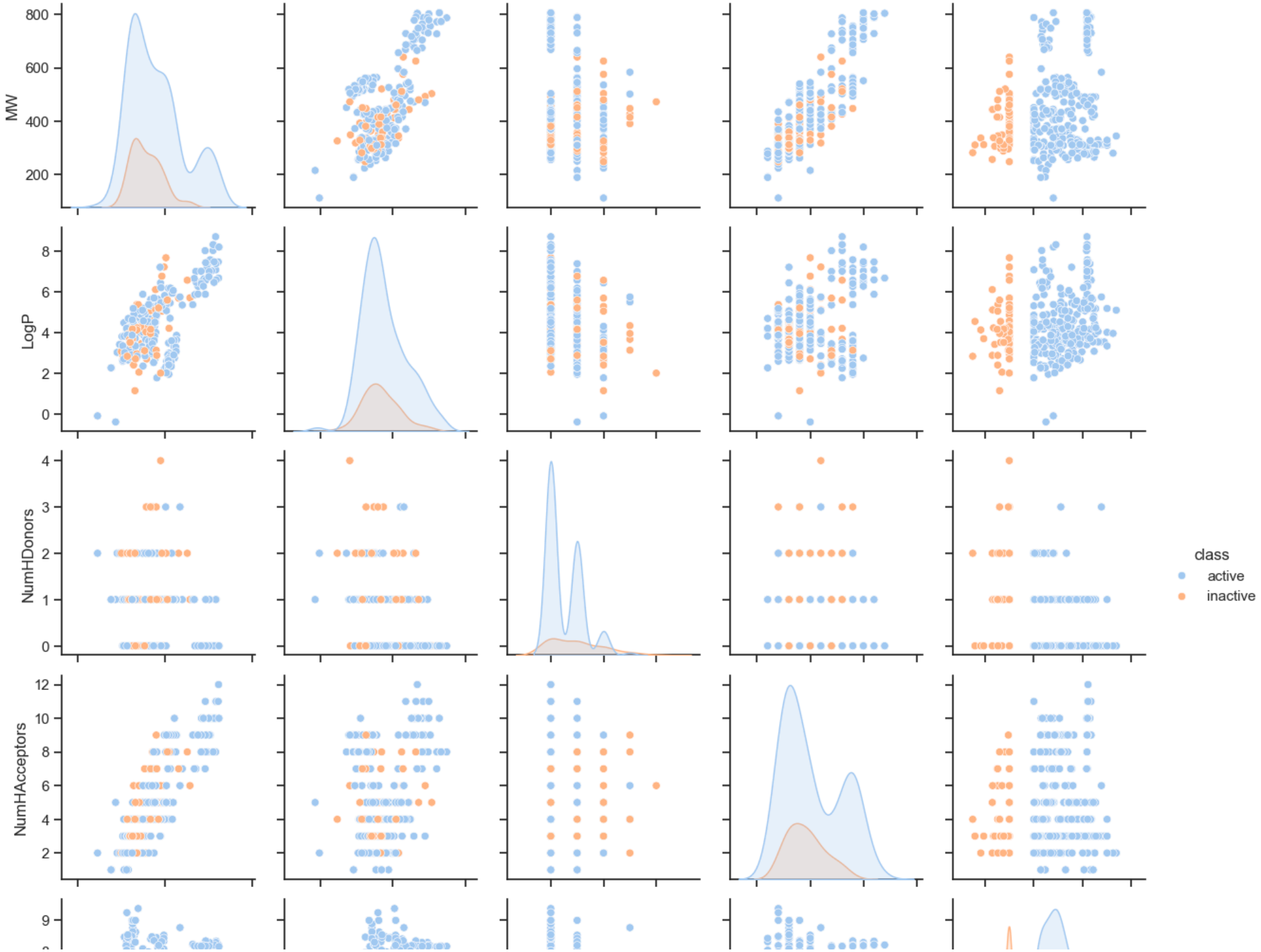
In [233…

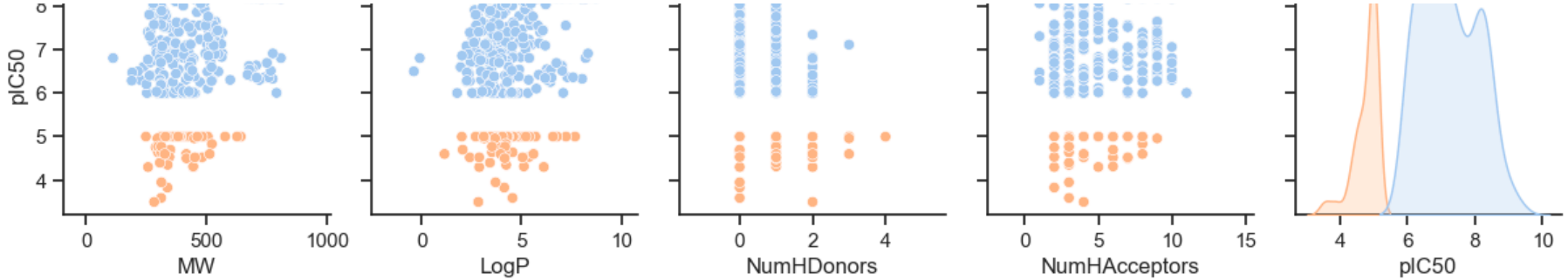
```
sns.pairplot(df_2class, vars=['NumHDonors', 'NumHAcceptors'], hue='class', palette='plasma')
# Show plot
plt.show()
```



In [237…

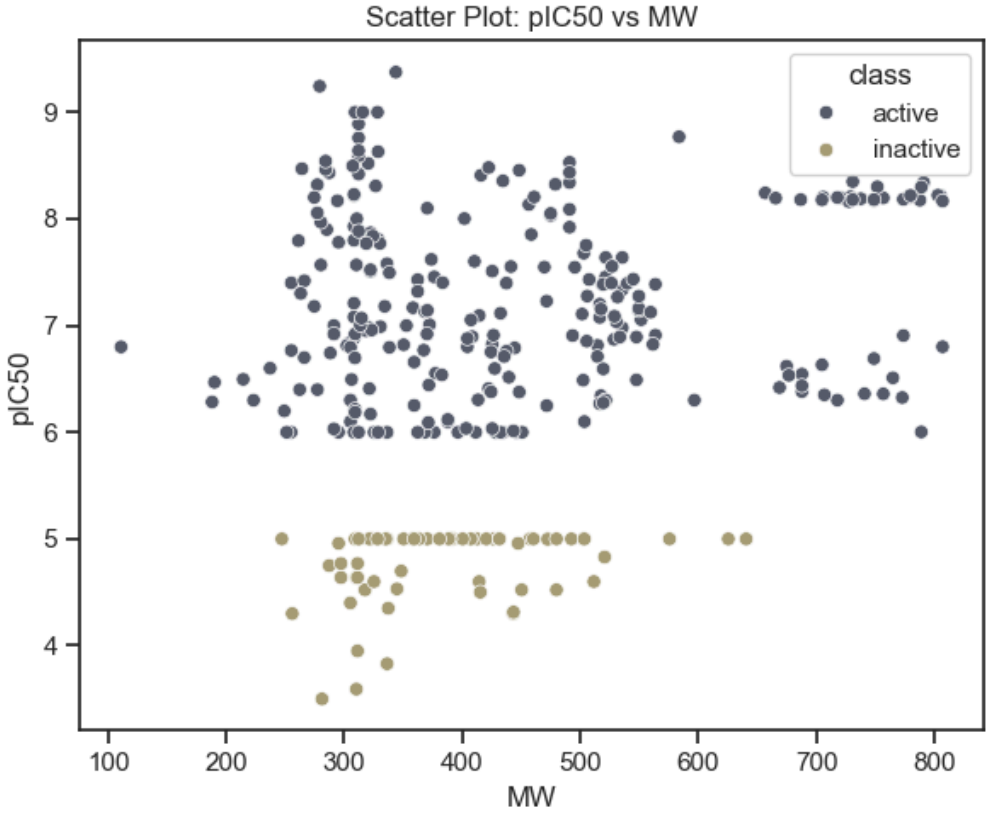
```
sns.pairplot(df_2class, hue='class', palette='pastel')
# Show plot
plt.show()
```

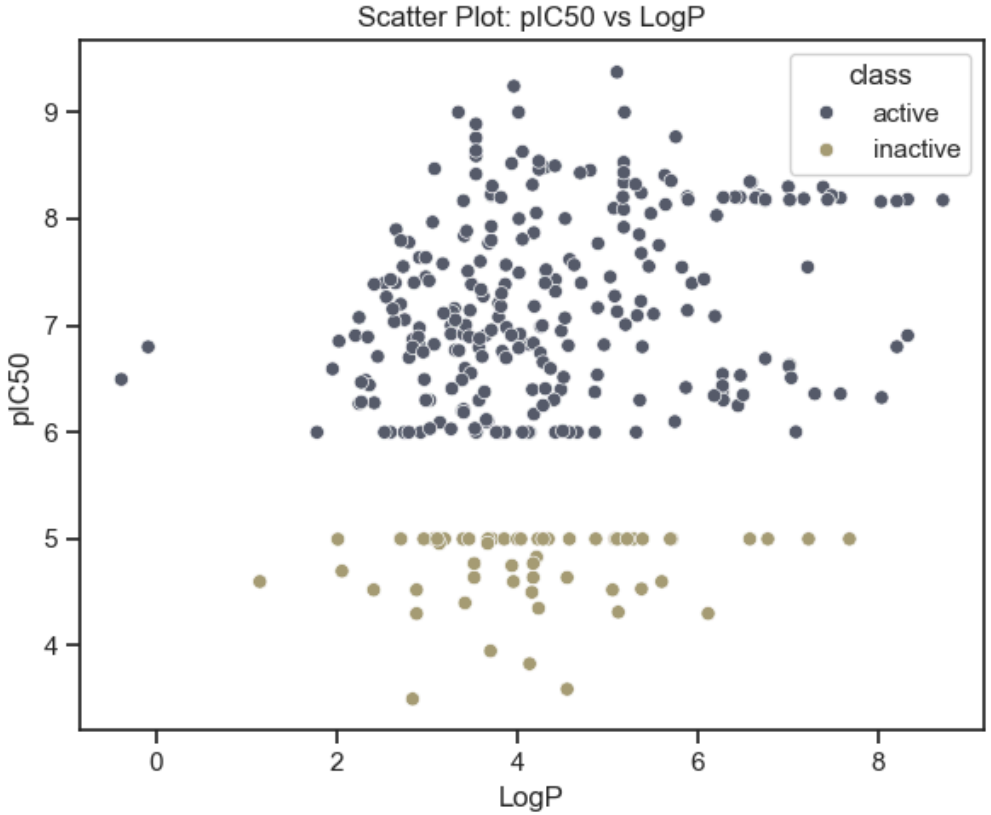

MW
LogP
NumHDonors
NumHAcceptors
800
600
400
200
8
6
4
2
0
4
3
2
1
0
12
10
8
6
4
2
9
class
active
inactive

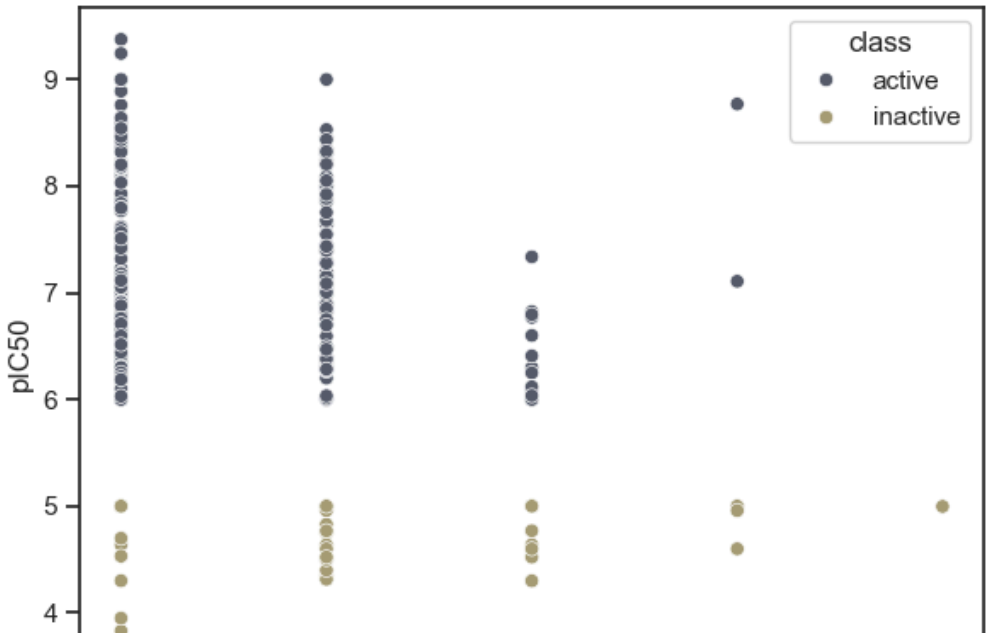


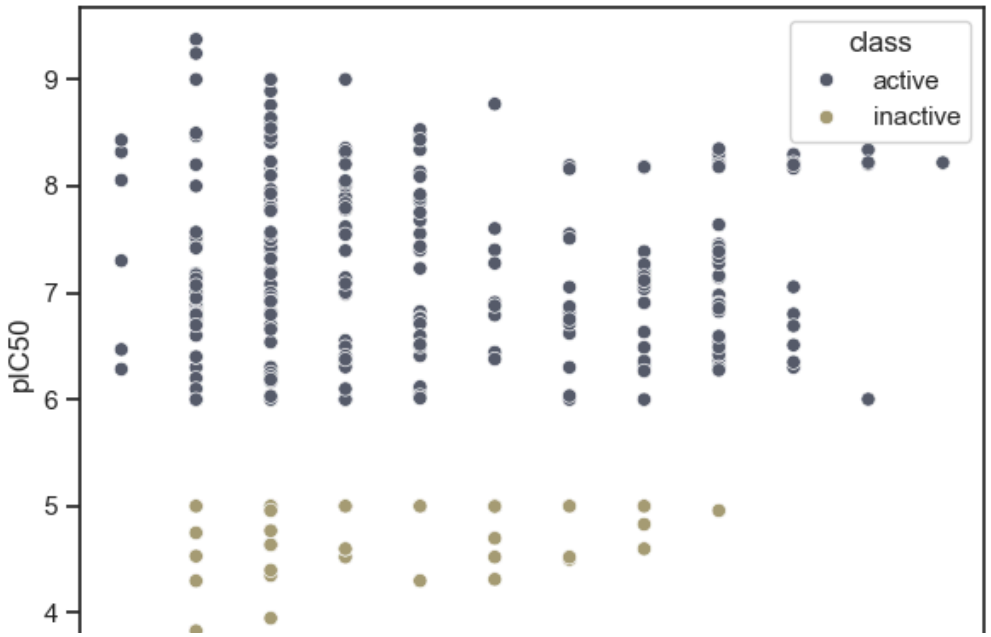
In [246…

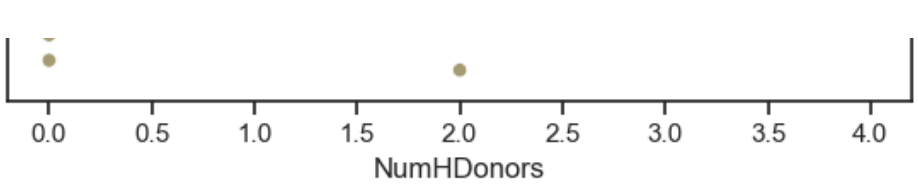
```
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors']
# Set figure size
plt.figure(figsize=(12, 10))
# Loop through numeric columns and create scatter plots
for i, column in enumerate(numeric_columns, 1):
 plt.subplot(2, 2, i) # 2 rows, 2 columns layout
 sns.scatterplot(x=df_2class[column],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

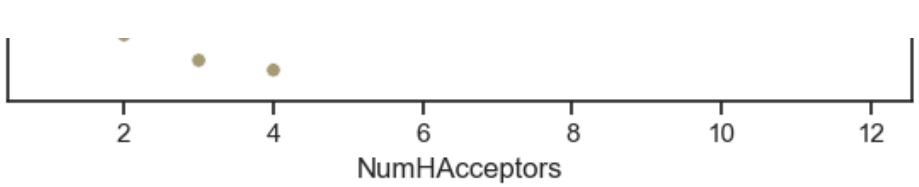

Scatter Plot: pIC50 vs MW
class
active
inactive
pIC50
MW
9
8
7
6
5
4
100
200
300
400
500
600
700
800


Scatter Plot: pIC50 vs LogP
class
active
inactive
pIC50
LogP
9
8
7
6
5
4
0
2
4
6
8


Scatter Plot: pIC50 vs NumHDonors
class
active
inactive
pIC50
9
8
7
6
5
4

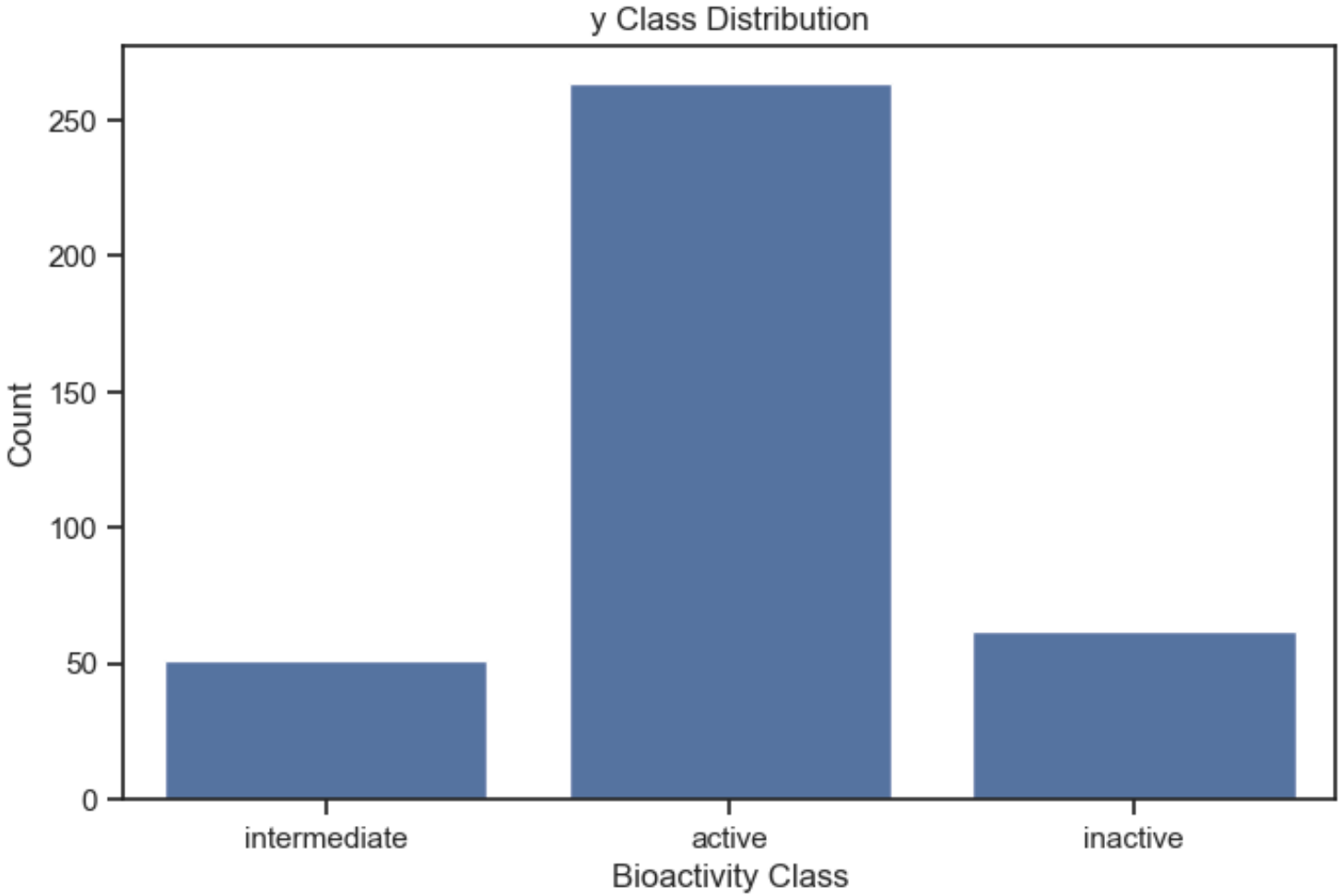

Scatter Plot: pIC50 vs NumHAcceptors
class
active
inactive
pIC50
9
8
7
6
5
4





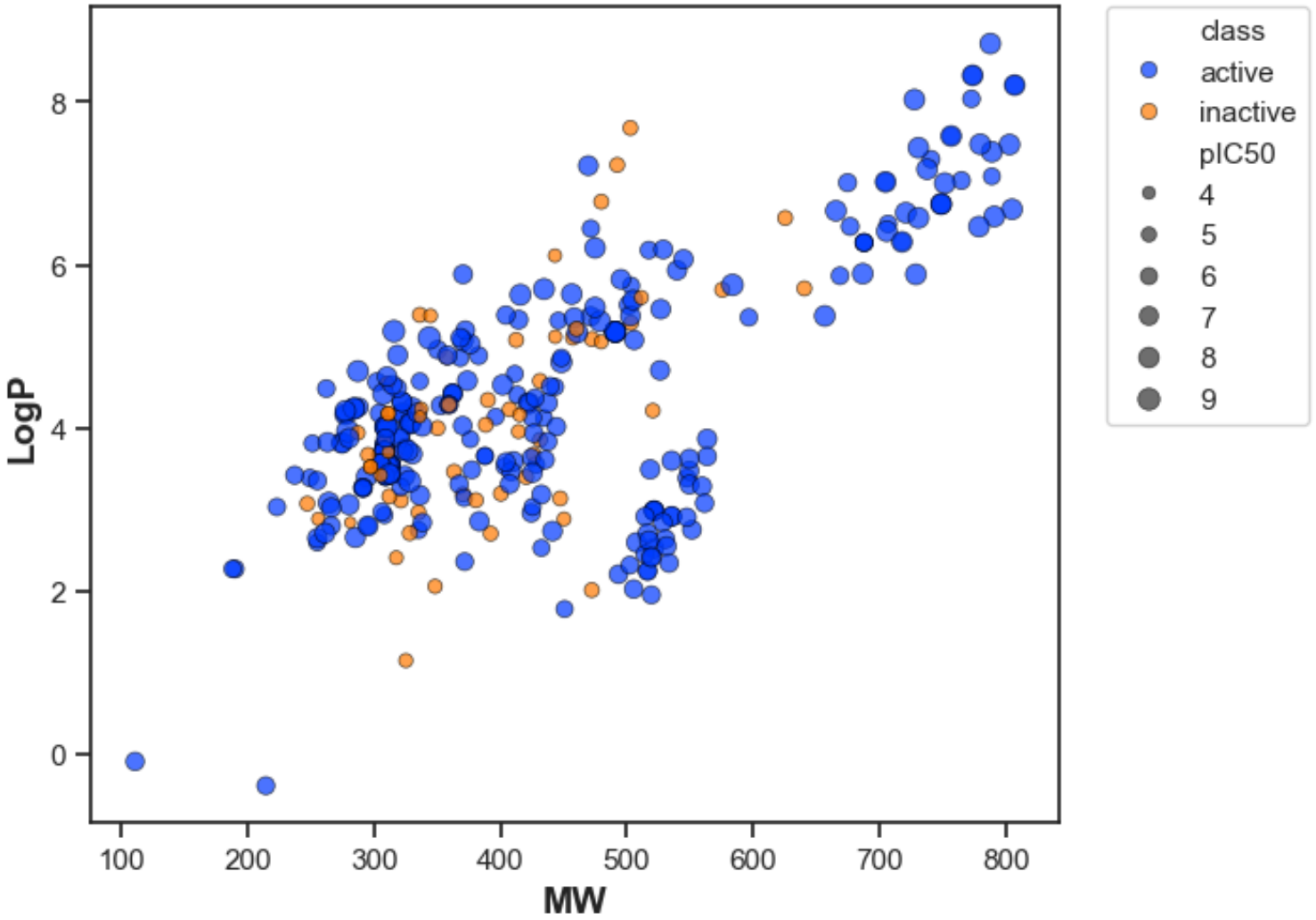
In [249…

```
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 [252…

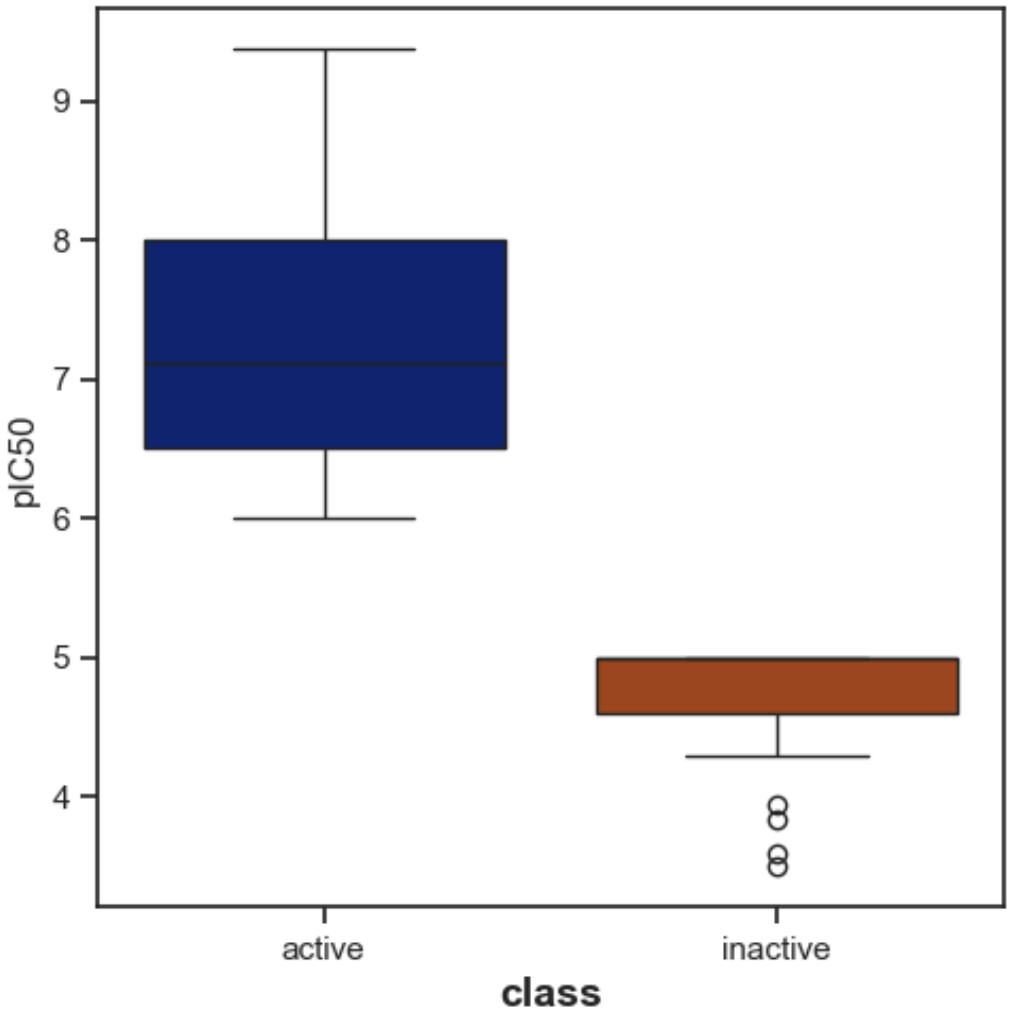
```
plt.figure(figsize=(6.6, 5.6))
sns.scatterplot(x='MW', y='LogP', data=df_2class, hue='class', size='pIC50',palette='bright', edgecolor='black', alpha=0.7)
plt.xlabel('MW', fontsize=14, fontweight='bold')
plt.ylabel('LogP', fontsize=14, fontweight='bold')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,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



In [257…

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

Out[257… Text(0.5, 0, 'class')



Statistical analysis | Mann-Whitney U Test

In [265…

```
 # 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 [268...
```
mannwhitney('pIC50')
```

Out[268...

	Descriptor	Statistics	p	alpha	Interpretation
0	pIC50	16368.0	1.454226e-34	0.05	Different distribution (reject H0)

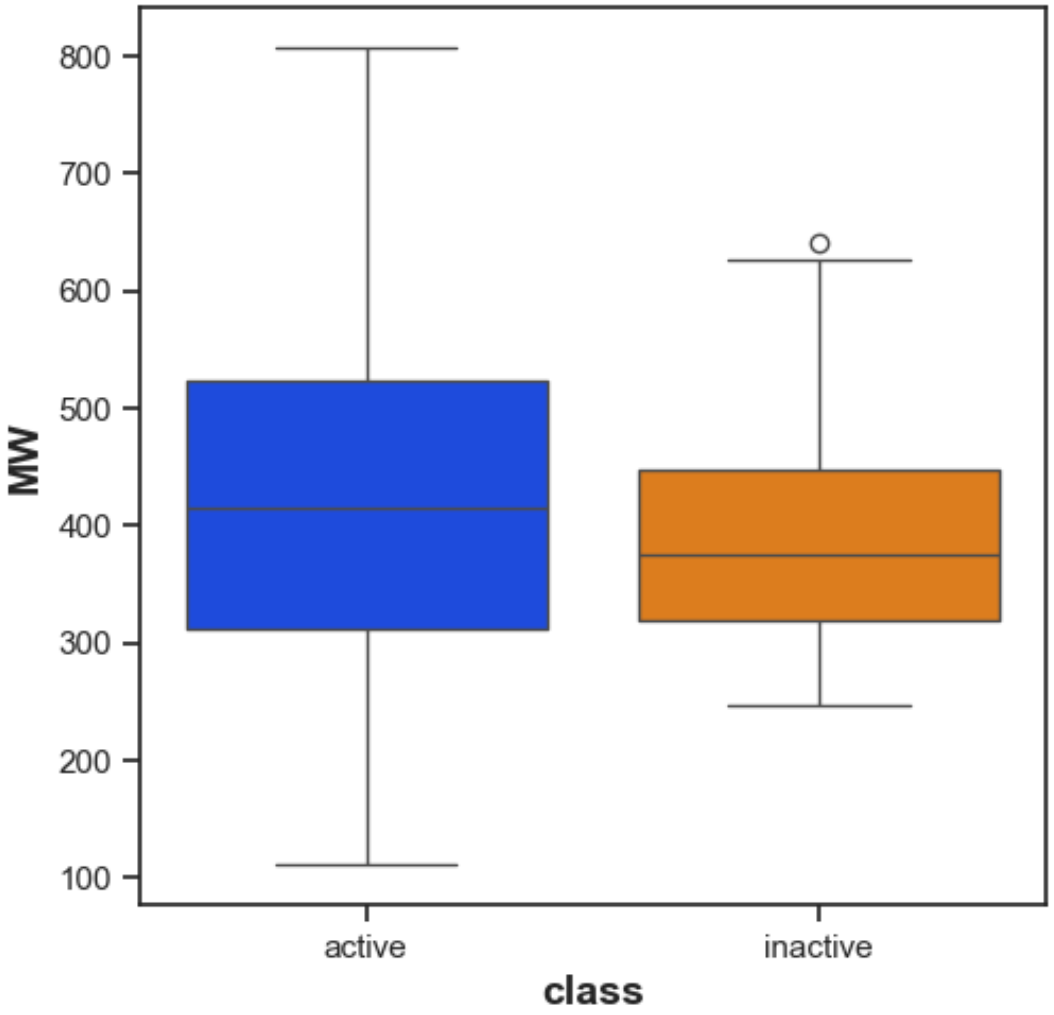
MW

In [280…

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

Out[280…

```
Text(0, 0.5, 'MW')
```



In [283…

```
mannwhitney('MW')
```

Out[283…

	Descriptor	Statistics	p	alpha	Interpretation
0	MW	9443.5	0.059407	0.05	Same distribution (fail to reject H0)

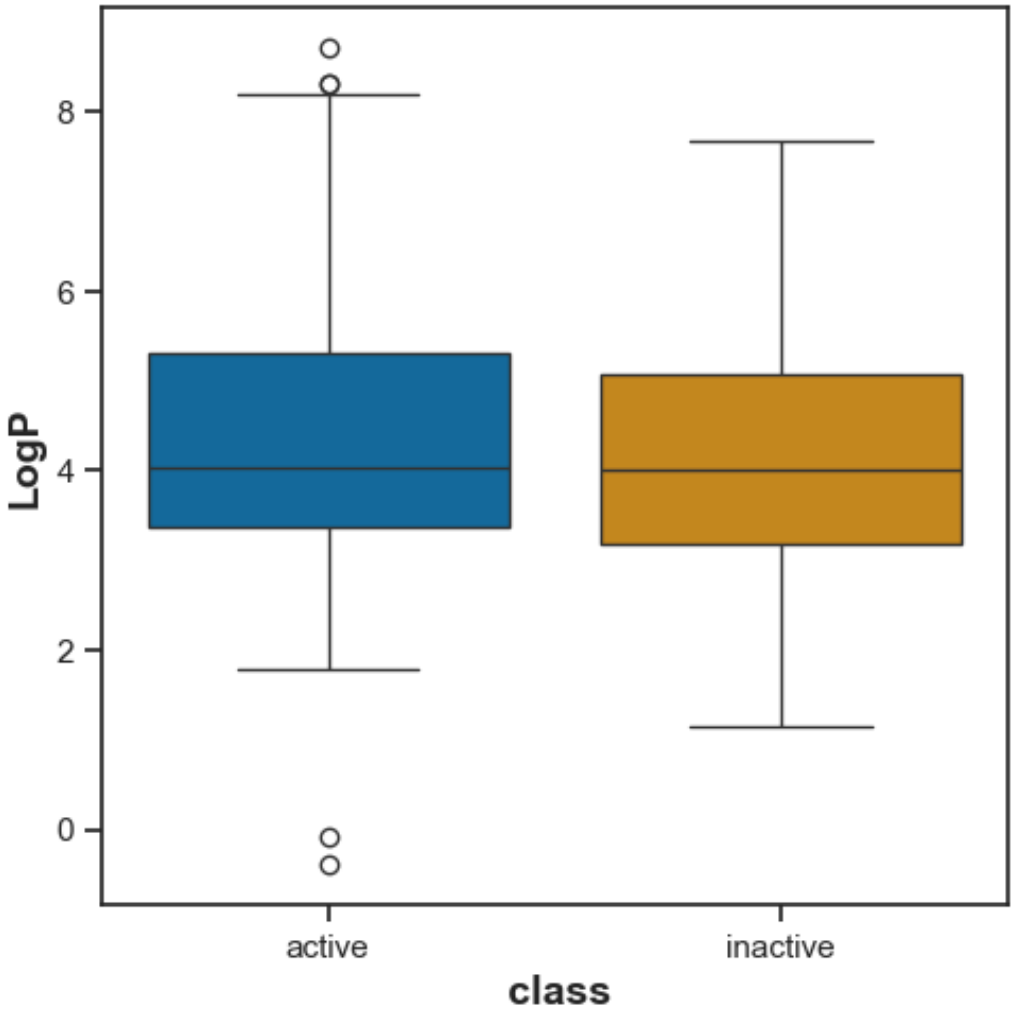
LogP

In [310…

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

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

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

Out[310…

```
Text(0, 0.5, 'LogP')
```



Statistical analysis | Mann-Whitney U Test

In [301…
```
mannwhitney('LogP')
```

Out[301…

	Descriptor	Statistics	p	alpha	Interpretation
0	LogP	8681.5	0.456767	0.05	Same distribution (fail to reject H0)

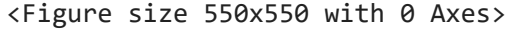
NumHDonors

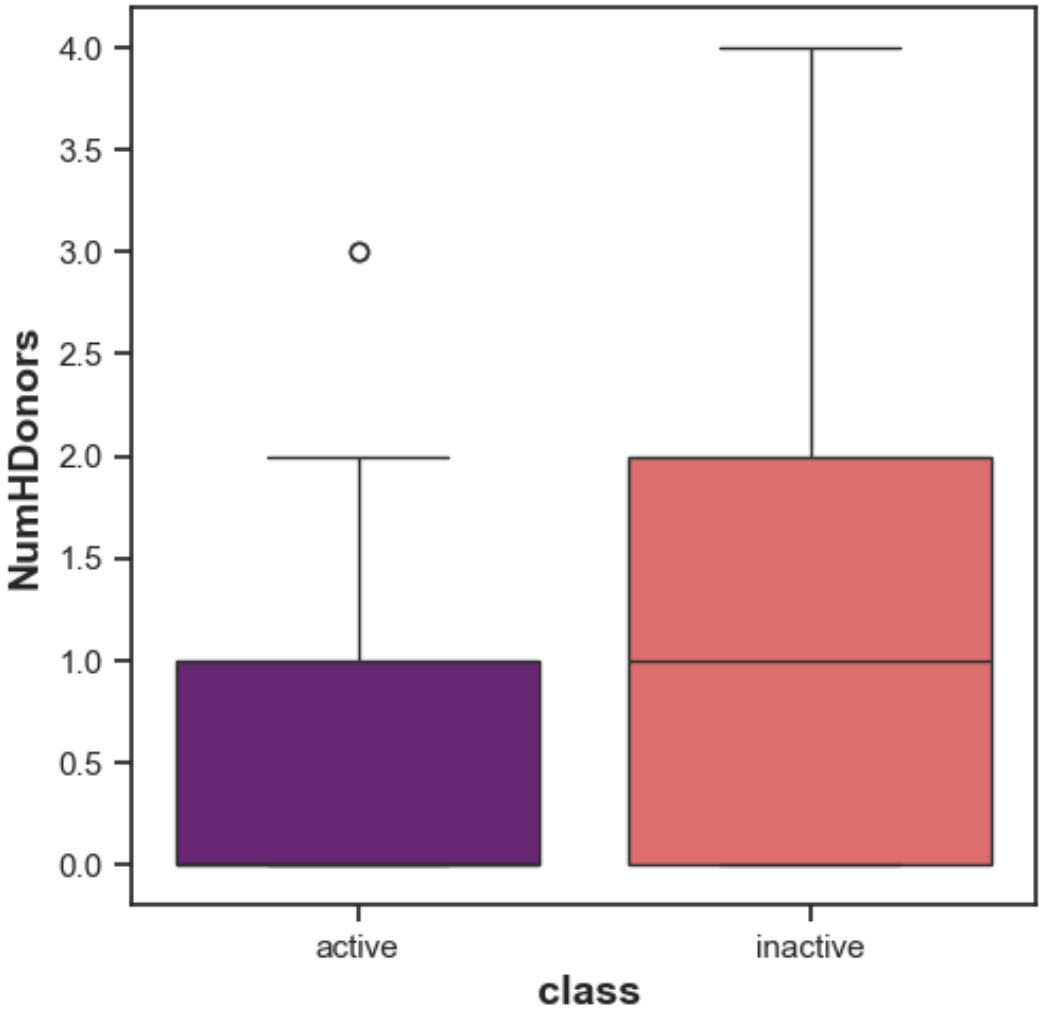
In [306…

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

Out[306…

```
Text(0, 0.5, 'NumHDonors')
```





Statistical analysis | Mann-Whitney U Test

In [319...
```
mannwhitney('NumHDonors')
```

Out[319...

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHDonors	6149.5	0.000651	0.05	Different distribution (reject H0)

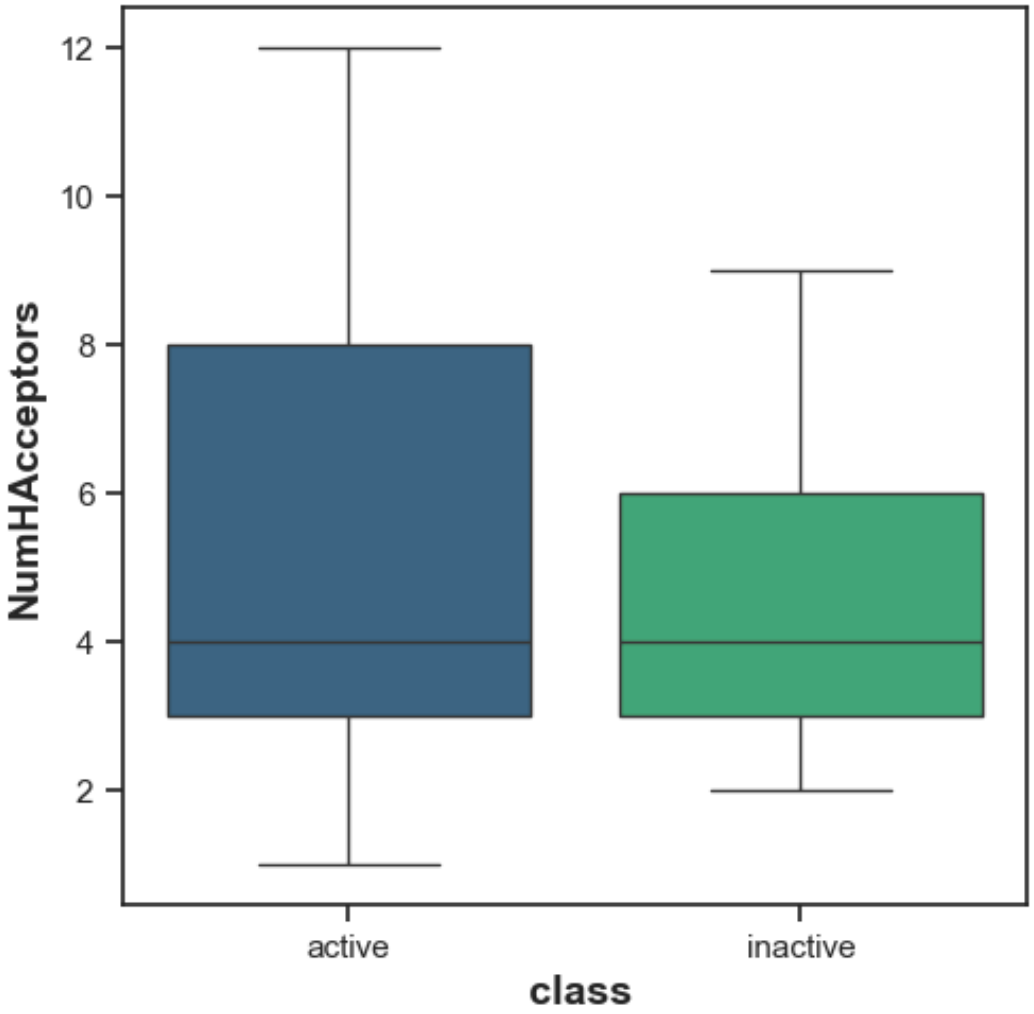
NumHAcceptors

In [327…

```
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[327…

```
Text(0, 0.5, 'NumHAcceptors')
```



Statistical analysis | Mann-Whitney U Test

In [321…
```
mannwhitney('NumHAcceptors')
```

Out[321…

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHAcceptors	8682.0	0.451515	0.05	Same distribution (fail to 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 exhibited difference between the actives and inactives while the other descriptors (only (MW, LogP, and NumHAcceptors shows statistically significant same difference between actives and inactives.

In [341…
```
df3 = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\Histamine H1 receptor\Histamine H1 receptor_04_bioactivity_data_3class_pIC50.cs
```

In [343…
```
df3
```

Out[343…

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors
0	2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	active	255.243000	2.59670	2.0	2.0
1	4	CHEMBL94249	NCCCCCCCCCCc1c[nH]cn1	active	223.364000	3.03170	2.0	2.0
2	5	CHEMBL157217	c1nc(CCCCCCCCCCN2CCCC2)c[nH]1	active	277.456000	4.16880	1.0	2.0
3	9	CHEMBL155188	NCCCCCCCCCCCCc1c[nH]cn1	active	251.418000	3.81190	2.0	2.0
4	12	CHEMBL1241	CN(C)CCN(Cc1ccccc1)c1ccccn1	active	255.365000	2.64980	0.0	3.0
...	...	...	...	...	...	...	...	...
321	371	CHEMBL4519018	CN1C2CCCC1CC(Nc1ccccc1Br)C2	active	309.251000	3.87630	1.0	2.0
322	373	CHEMBL5398630	O=C(O)/C=C/C(=O)O.[2H]C([2H])(Cc1c[nH]c2ccccc1...	active	190.286204	2.27200	1.0	1.0
323	374	CHEMBL3183055	CN(C)CCc1c[nH]c2ccccc12.O=C(O)/C=C/C(=O)O	active	188.274000	2.27200	1.0	1.0
324	375	CHEMBL2017291	COc1cc2c(cc1-c1c(C)noc1C)ncc1[nH]c(=O)n([C@H](...	inactive	415.453000	4.16254	1.0	7.0
325	376	CHEMBL3643413	CCC(=O)N1CC[C@H](Nc2ncnc3c2CN(c2cnc(OC)c(C(F)(...	inactive	450.465000	2.88450	1.0	7.0

326 rows × 9 columns

In [346...]
```
df3.head()
```

Out[346...]

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC50
0	2	CHEMBL24665	NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1	active	255.243	2.5967	2.0	2.0	6.000000
1	4	CHEMBL94249	NCCCCCCCCCCc1c[nH]cn1	active	223.364	3.0317	2.0	2.0	6.299998
2	5	CHEMBL157217	c1nc(CCCCCCCCCCN2CCCC2)c[nH]1	active	277.456	4.1688	1.0	2.0	6.399997
3	9	CHEMBL155188	NCCCCCCCCCCCCc1c[nH]cn1	active	251.418	3.8119	2.0	2.0	6.000000
4	12	CHEMBL1241	CN(C)CCN(Cc1ccccc1)c1ccccn1	active	255.365	2.6498	0.0	3.0	7.400008

In [348...]
```
df3.describe()
```

Out[348...]

	Unnamed: 0	MW	LogP	NumHDonors	NumHAcceptors	pIC50
count	326.000000	326.000000	326.000000	326.000000	326.000000	326.000000
mean	190.895706	434.956041	4.331838	0.595092	5.073620	6.760478
std	107.148013	147.782879	1.490037	0.761683	2.632479	1.237425
min	2.000000	111.148000	-0.386300	0.000000	1.000000	3.500000
25%	100.250000	312.778500	3.349175	0.000000	3.000000	6.005044
50%	189.500000	407.516000	4.034560	0.000000	4.000000	6.861769
75%	283.750000	516.598000	5.185950	1.000000	7.000000	7.734490
max	376.000000	806.976000	8.715300	4.000000	12.000000	9.375718

In [362...]
```
selection = ['canonical_smiles','molecule_chembl_id']
df3_selection = df3[selection]
df3_selection.to_csv('molecule.smi', sep='\t', index=False, header=False)
```

In [364...]
```
with open('molecule.smi', 'r') as file:
 for _ in range(5):
```

```
    print(file.readline().strip())
```

```
NCCc1c[nH]c(-c2cccc(C(F)(F)F)c2)n1      CHEMBL24665
NCCCCCCCCCCc1c[nH]cn1   CHEMBL94249
c1nc(CCCCCCCCCCN2CCCC2)c[nH]1   CHEMBL157217
NCCCCCCCCCCCCc1c[nH]cn1 CHEMBL155188
CN(C)CCN(Cc1ccccc1)c1ccccn1     CHEMBL1241
```

In [366…
```
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: 326
```

Preparing the X and Y Data Matrices

In [416…
```
X = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\New folder\archive\descriptors_output.csv")
```

In [418…
```
X
```

Out[418…

	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 [420…

```
X.head(5)
```

Out[420…

	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 [421…

```
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 [424…

```
X.dtypes
```

Out[424…

```
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 [425…

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

Out[425...

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	Pubchem
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 [429...

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

In [430...

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

Split dataset

In [435...

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

```
Shape of X: (6946, 881)
Shape of y: (326,)
```

In [439…
```
X = X.iloc[:y.shape[0], :]  # Trim X to match y
```

In [442…
```
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (326, 881)
Shape of y: (326,)
```

In [445…
```
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 [448…
```
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 [469…
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Logistic Regression Model

In [471…
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
lr = LogisticRegression(max_iter=500)
```

In [472…
```
lr.fit(X_train, y_train)
```

Out[472…

▾ LogisticRegression

```
LogisticRegression(max_iter=500)
```

In [475…
```
y_pred_lr = lr.predict(X_test)
y_pred_lr
```

Out[475…
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 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, 1, 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, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [478…
```
print(classification_report(y_test, y_pred_lr))
```

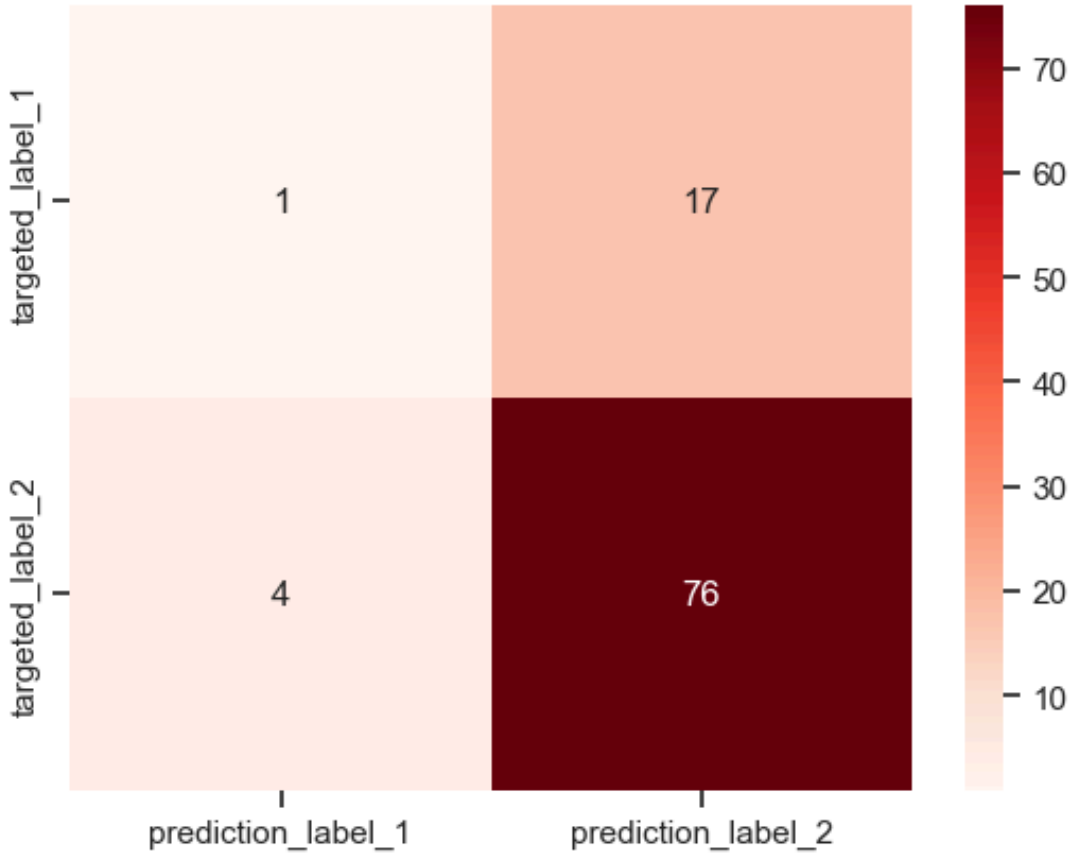
```
              precision    recall  f1-score   support

           0       0.20      0.06      0.09        18
           1       0.82      0.95      0.88        80

    accuracy                           0.79        98
   macro avg       0.51      0.50      0.48        98
weighted avg       0.70      0.79      0.73        98
```

In [481…
```
accuracy = accuracy_score(y_test, y_pred_lr)
print(f"Logistic Regression Model Accuracy: {accuracy * 100:.2f}%")
```

```
Logistic Regression Model Accuracy: 78.57%
```

In [484…
```
cm = confusion_matrix(y_test, y_pred_lr)
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[484…
```
[Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



Logistic Regression Model Accuracy: 78.57%

In [491…

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(X_train,y_train)
```

Out[491…

DecisionTreeClassifier

```
DecisionTreeClassifier()
```

In [494…

```
y_pred_DT = DT.predict(X_test)
y_pred_DT
```

Out[494…
```
array([1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [497…
```
print(classification_report(y_test,y_pred_DT))
```

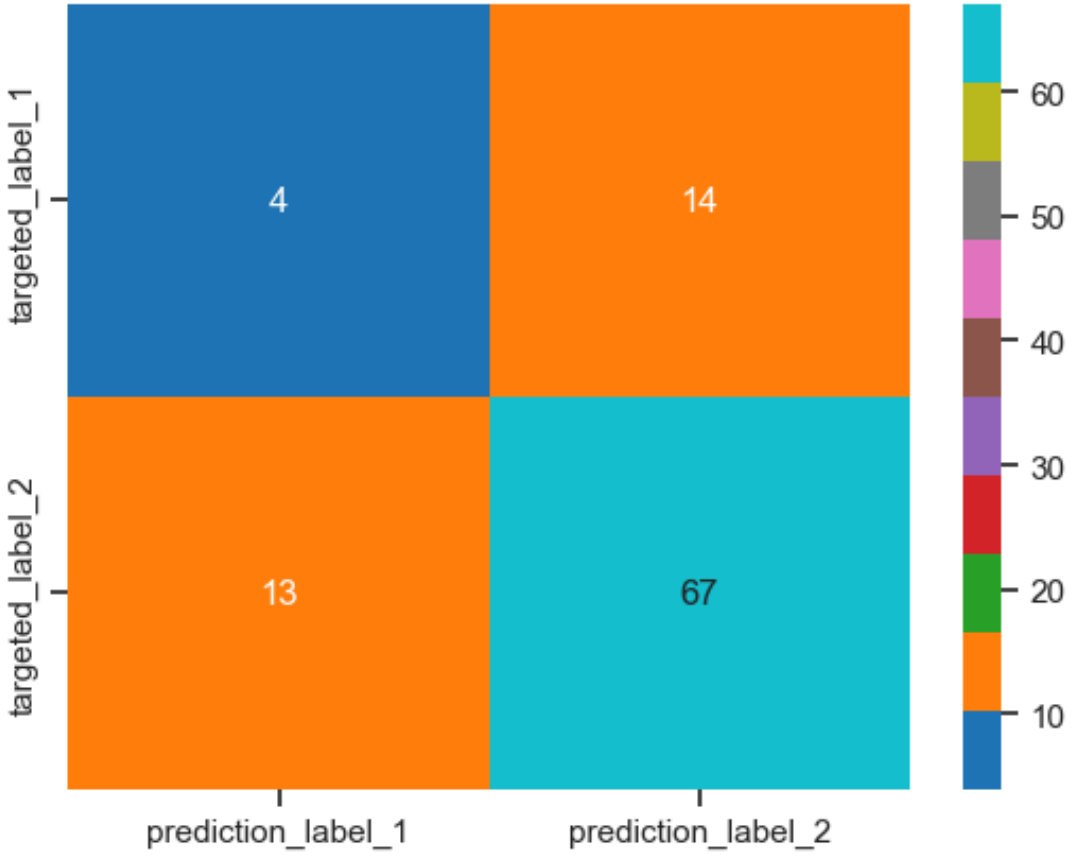
```
              precision    recall  f1-score   support

           0       0.24      0.22      0.23        18
           1       0.83      0.84      0.83        80

    accuracy                           0.72        98
   macro avg       0.53      0.53      0.53        98
weighted avg       0.72      0.72      0.72        98
```

In [500…
```
accuracy = accuracy_score(y_test, y_pred_DT)
print(f"DecisionTreeClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
DecisionTreeClassifier Model Accuracy: 72.45%
```

In [508…
```
cm = confusion_matrix(y_test, y_pred_DT)
ax = sns.heatmap(cm, annot=True, fmt='d', cmap="tab10")
ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

Out[508…
```
[Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



DecisionTreeClassifier Model Accuracy: 72.45%

RandomForestClassifier

In [519…

```
rnf = RandomForestClassifier()
rnf.fit(X_train,y_train)
```

Out[519…

RandomForestClassifier

RandomForestClassifier()

In [521…
```
y_pred_rnf = rnf.predict(X_test)
y_pred_rnf
```

Out[521…
```
array([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, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 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], dtype=int64)
```

In [523…
```
print(classification_report(y_test,y_pred_rnf))
```

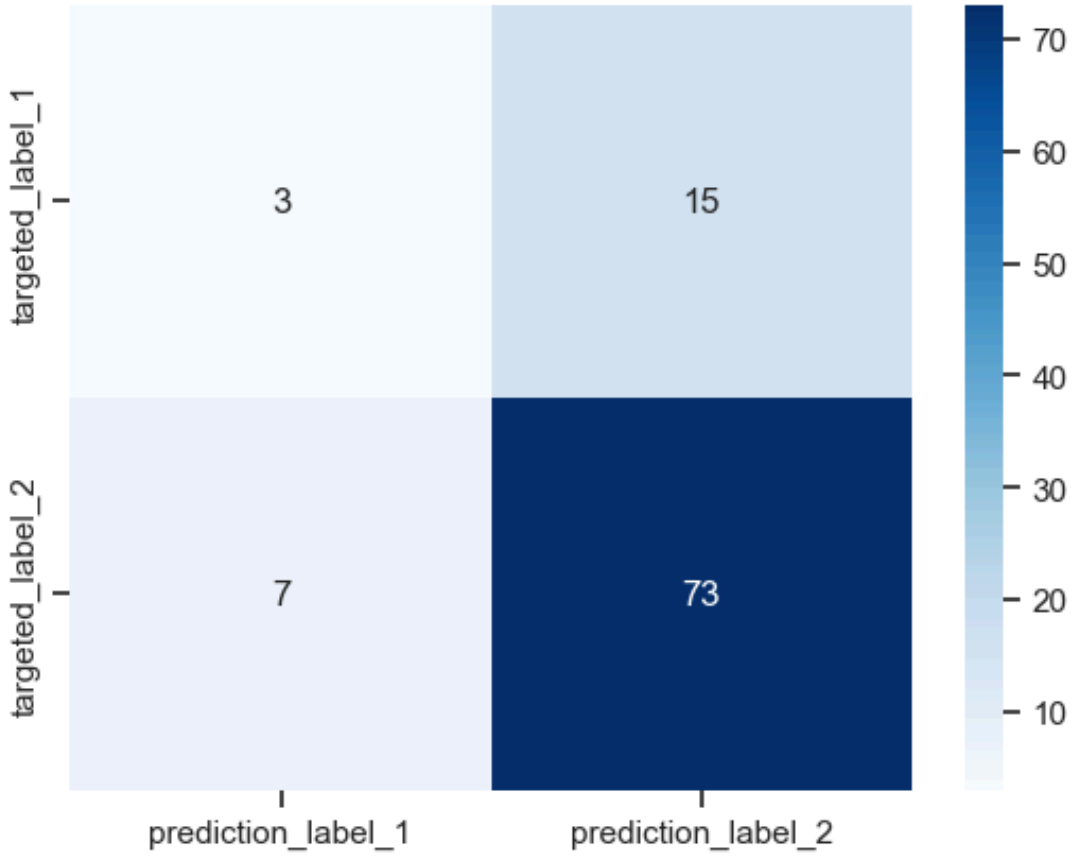
```
              precision    recall  f1-score   support

           0       0.30      0.17      0.21        18
           1       0.83      0.91      0.87        80

    accuracy                           0.78        98
   macro avg       0.56      0.54      0.54        98
weighted avg       0.73      0.78      0.75        98
```

In [539…
```
accuracy = accuracy_score(y_test, y_pred_rnf)
print(f"RandomForestClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
RandomForestClassifier Model Accuracy: 77.55%
```

In [530…
```
cm = confusion_matrix(y_test, y_pred_rnf)
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[530…
```
[Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



RandomForestClassifier Model Accuracy: 77.55%

K-Nearest Neighbors (KNN)

In [547…

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

Out[547…

KNeighborsClassifier

KNeighborsClassifier()

In [549…
```
y_pred_knn = knn.predict(X_test)
y_pred_knn
```

Out[549…
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 1, 1, 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, 1], dtype=int64)
```

In [551…
```
print(classification_report(y_test,y_pred_knn))
```

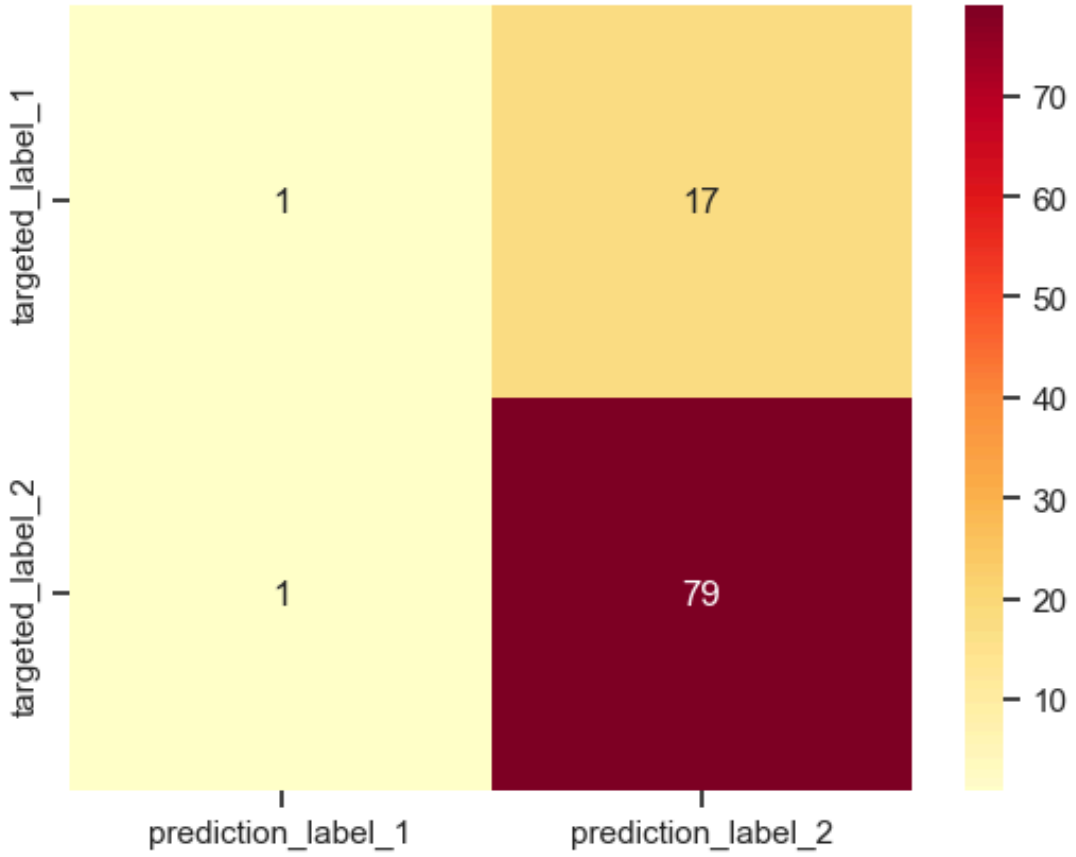
```
              precision    recall  f1-score   support

           0       0.50      0.06      0.10        18
           1       0.82      0.99      0.90        80

    accuracy                           0.82        98
   macro avg       0.66      0.52      0.50        98
weighted avg       0.76      0.82      0.75        98
```

In [554…
```
accuracy = accuracy_score(y_test, y_pred_knn)
print(f"KNeighborsClassifier Model Accuracy: {accuracy * 100:.2f}%")
```

```
KNeighborsClassifier Model Accuracy: 81.63%
```

In [564…
```
cm = confusion_matrix(y_test, y_pred_knn)
ax = sns.heatmap(cm, annot=True, fmt='d', cmap="YlOrRd")
ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

Out[564…
```
[Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]
```



KNeighborsClassifier Model Accuracy: 81.63%

In [571…
```
df3_X = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\New folder\archive\descriptors_output.csv")
```

In [574…
```
df3_X
```

Out[574...

	Name	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	Pubch
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 [577...

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

Out[577...

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	PubchemI
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 [596...

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

Out[596…

```
0     6.00
1     6.30
2     6.40
3     6.00
4     7.40
      ...
321   6.70
322   6.47
323   6.28
324   4.50
325   4.52
Name: pIC50, Length: 326, dtype: float64
```

In [598…

```
dataset3 = pd.concat([df3_X,df3_Y], axis=1)
dataset3
```

Out[598…

	PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	PubchemI
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 [600…
```
from sklearn.model_selection import train_test_split
import lazypredict
from lazypredict.Supervised import LazyRegressor
```

In [601…
```
X.shape
```

Out[601… (326, 129)

In [604…
```
# Remove low variance features
from sklearn.feature_selection import VarianceThreshold
selection = VarianceThreshold(threshold=(.8 * (1 - .8)))
X = selection.fit_transform(X)
X.shape
```

Out[604… (326, 129)

In [605…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [607…
```
clf = LazyRegressor(verbose=0,ignore_warnings=True, custom_metric=None)
models_train,predictions_train = clf.fit(X_train, X_train, Y_train, Y_train)
models_test,predictions_test = clf.fit(X_train, X_test, Y_train, Y_test)
```

```
100%|██████████| 42/42 [00:28<00:00,  1.47it/s]
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001114 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 387
[LightGBM] [Info] Number of data points in the train set: 260, number of used features: 129
[LightGBM] [Info] Start training from score 6.709252
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

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

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
'tuple' object has no attribute '__name__'
Invalid Regressor(s)
```

```
100%|██████████| 42/42 [00:25<00:00,  1.63it/s]
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001046 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 387
[LightGBM] [Info] Number of data points in the train set: 260, number of used features: 129
[LightGBM] [Info] Start training from score 6.709252
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

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

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

In [609…

```
predictions_train
```

Out[609...

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
DecisionTreeRegressor	0.23	0.61	0.77	0.04
ExtraTreeRegressor	0.23	0.61	0.77	0.05
ExtraTreesRegressor	0.23	0.61	0.77	1.20
GaussianProcessRegressor	0.23	0.61	0.77	0.07
XGBRegressor	0.23	0.61	0.77	1.39
RandomForestRegressor	0.03	0.51	0.86	1.11
BaggingRegressor	-0.04	0.48	0.90	0.14
MLPRegressor	-0.22	0.39	0.97	1.68
GradientBoostingRegressor	-0.23	0.38	0.97	0.45
HistGradientBoostingRegressor	-0.34	0.33	1.02	0.68
LGBMRegressor	-0.35	0.32	1.02	0.16
Ridge	-0.43	0.28	1.05	0.04
RidgeCV	-0.53	0.23	1.09	0.06
HuberRegressor	-0.58	0.21	1.10	0.21
SGDRegressor	-0.63	0.18	1.12	0.04
PoissonRegressor	-0.64	0.18	1.12	0.07
NuSVR	-0.67	0.16	1.13	0.06
LinearSVR	-0.71	0.14	1.15	0.13
SVR	-0.76	0.11	1.17	0.07
AdaBoostRegressor	-0.78	0.11	1.17	0.10
OrthogonalMatchingPursuit	-0.78	0.11	1.17	0.04

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
KNeighborsRegressor	-0.79	0.10	1.17	0.06
TweedieRegressor	-0.81	0.09	1.18	0.06
GammaRegressor	-0.81	0.09	1.18	0.23
LarsCV	-0.94	0.02	1.22	0.51
OrthogonalMatchingPursuitCV	-0.95	0.02	1.23	0.08
BayesianRidge	-0.99	0.00	1.24	0.09
LassoLars	-0.99	0.00	1.24	0.03
DummyRegressor	-0.99	0.00	1.24	0.03
ElasticNet	-0.99	0.00	1.24	0.04
ElasticNetCV	-0.99	0.00	1.24	10.28
LassoLarsCV	-0.99	0.00	1.24	0.20
LassoCV	-0.99	0.00	1.24	7.27
Lasso	-0.99	0.00	1.24	0.03
LassoLarsIC	-0.99	0.00	1.24	0.10
QuantileRegressor	-1.01	-0.01	1.25	0.13
PassiveAggressiveRegressor	-1.81	-0.41	1.47	0.04
TransformedTargetRegressor	-1.84	-0.43	1.48	0.04
LinearRegression	-1.84	-0.43	1.48	0.16
KernelRidge	-58.83	-29.03	6.79	0.04
RANSACRegressor	-346209153905188605316300 8.00	-173772934392565703324467 2.00	1633586300395.05	1.13
Lars	-449463117505541425445470208 0.00	-225599248168804581784944640 0.00	58859943278549.79	0.12

In [613…

```
predictions_test
```

Out[613...

Model	Adjusted R-Squared	R-Squared	
Lars	34935877234292879230979793899081844945700065273...	-34398402199919143777719716499119321959445404 89...	2
LinearRegression	3022246608953207286661 12.00	-2975750507277004025364 48.00	
TransformedTargetRegressor	3022246608953207286661 12.00	-2975750507277004025364 48.00	
RANSACRegressor	1200295131362980507156 48.00	-1181829052418934712565 76.00	
KernelRidge	36.32	-33.78	
GaussianProcessRegressor	15.73	-13.50	
PassiveAggressiveRegressor	3.01	-0.98	
DecisionTreeRegressor	2.87	-0.84	
ExtraTreesRegressor	2.77	-0.75	
XGBRegressor	2.71	-0.68	
ExtraTreeRegressor	2.69	-0.66	
HuberRegressor	2.67	-0.64	
LinearSVR	2.62	-0.60	
MLPRegressor	2.59	-0.56	
Ridge	2.58	-0.55	
BaggingRegressor	2.47	-0.44	
RandomForestRegressor	2.35	-0.33	
RidgeCV	2.32	-0.30	
SGDRegressor	2.29	-0.27	
KNeighborsRegressor	2.24	-0.22	
GradientBoostingRegressor	2.22	-0.21	

Model	Adjusted R-Squared	R-Squared
PoissonRegressor	2.19	-0.17
LGBMRegressor	2.17	-0.15
HistGradientBoostingRegressor	2.15	-0.13
AdaBoostRegressor	2.12	-0.11
OrthogonalMatchingPursuit	2.09	-0.08
GammaRegressor	2.07	-0.06
TweedieRegressor	2.07	-0.05
LassoLarsIC	2.06	-0.04
DummyRegressor	2.06	-0.04
ElasticNet	2.06	-0.04
ElasticNetCV	2.06	-0.04
LassoLarsCV	2.06	-0.04
Lasso	2.06	-0.04
LassoCV	2.06	-0.04
LassoLars	2.06	-0.04
BayesianRidge	2.06	-0.04
SVR	2.04	-0.03
OrthogonalMatchingPursuitCV	2.04	-0.02
LarsCV	2.04	-0.02
QuantileRegressor	2.03	-0.01
NuSVR	2.01	0.00

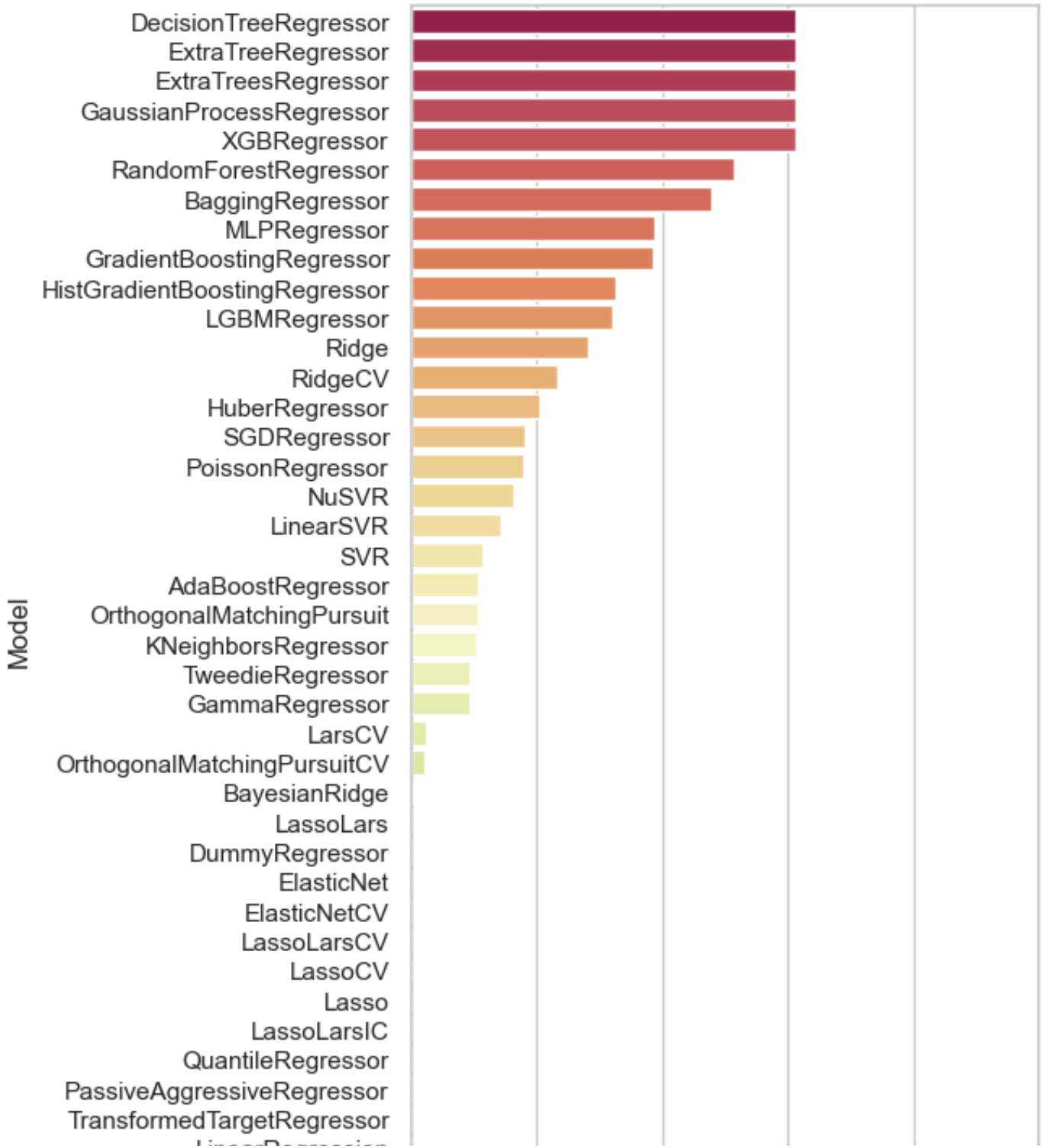
Bar plot of R-squared values

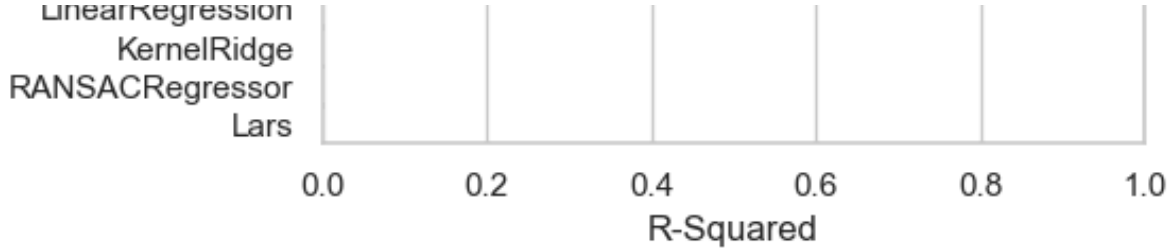
In [618…

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

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

Out[618…

```
[(0.0, 1.0)]
```


Model
DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
RandomForestRegressor
BaggingRegressor
MLPRegressor
GradientBoostingRegressor
HistGradientBoostingRegressor
LGBMRegressor
Ridge
RidgeCV
HuberRegressor
SGDRegressor
PoissonRegressor
NuSVR
LinearSVR
SVR
AdaBoostRegressor
OrthogonalMatchingPursuit
KNeighborsRegressor
TweedieRegressor
GammaRegressor
LarsCV
OrthogonalMatchingPursuitCV
BayesianRidge
LassoLars
DummyRegressor
ElasticNet
ElasticNetCV
LassoLarsCV
LassoCV
Lasso
LassoLarsIC
QuantileRegressor
PassiveAggressiveRegressor
TransformedTargetRegressor



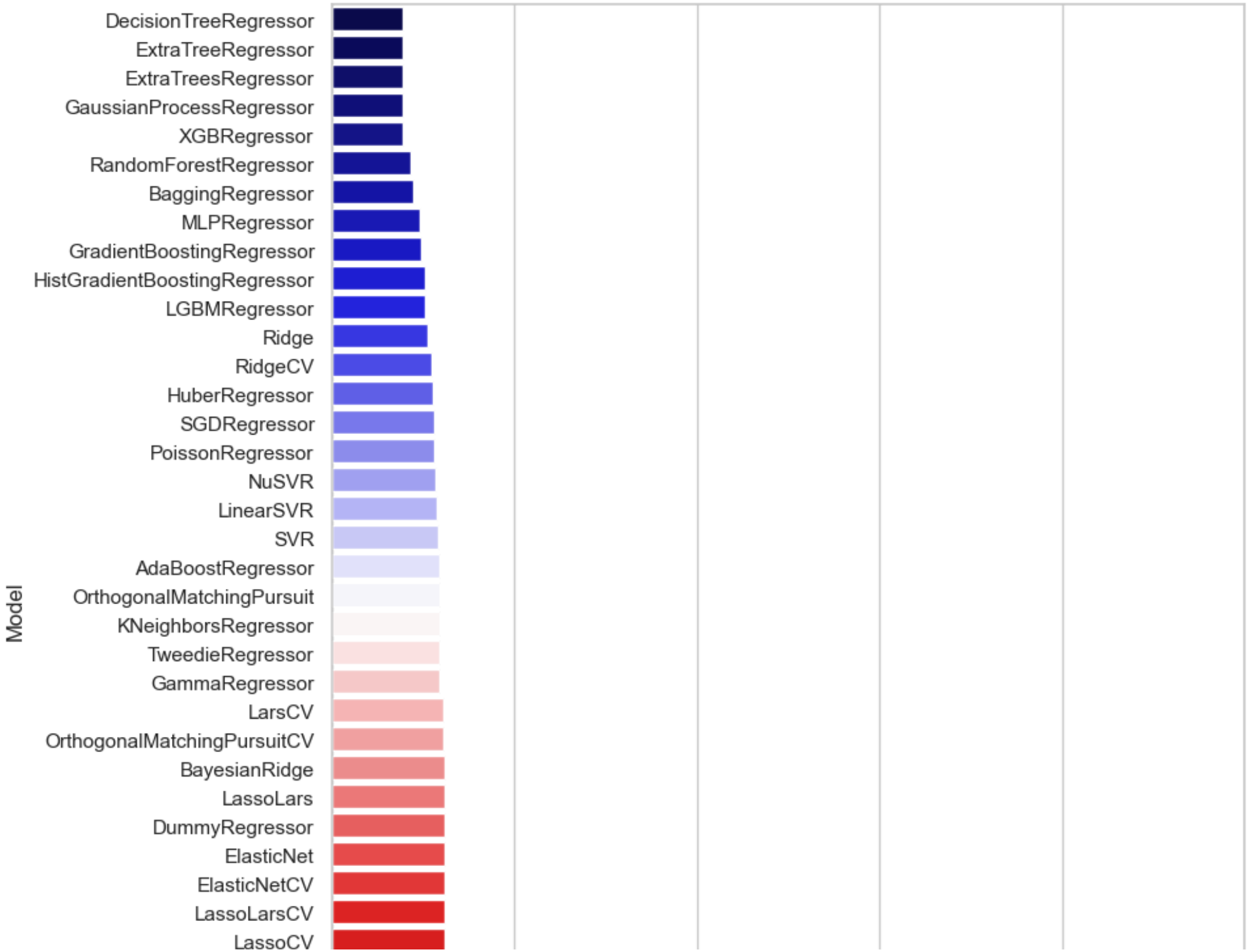
Bar plot of RMSE values

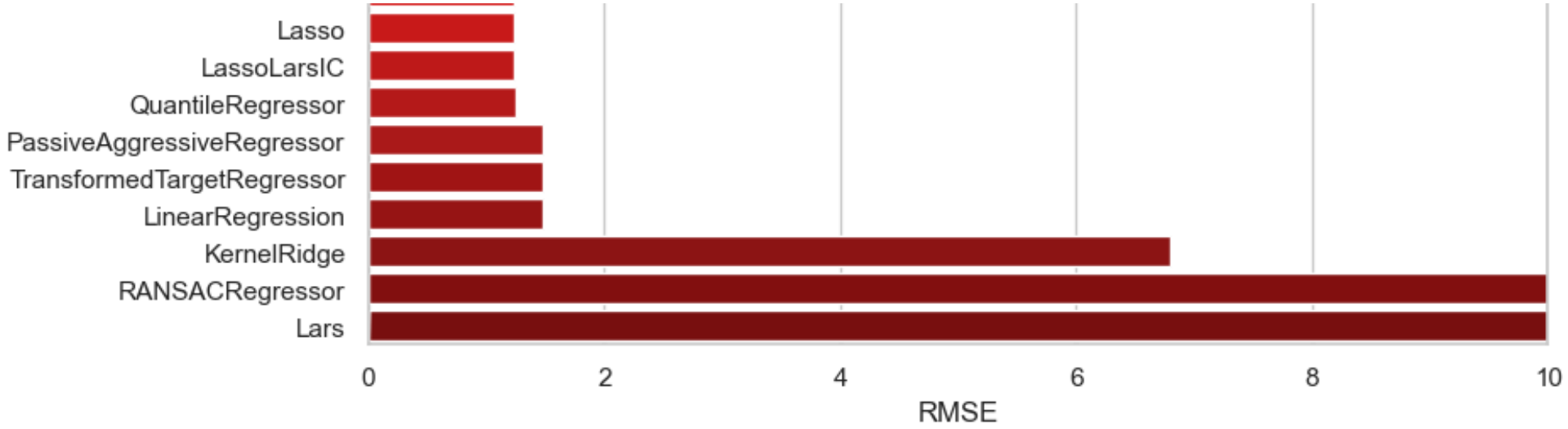
In [624…

```
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[624…

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

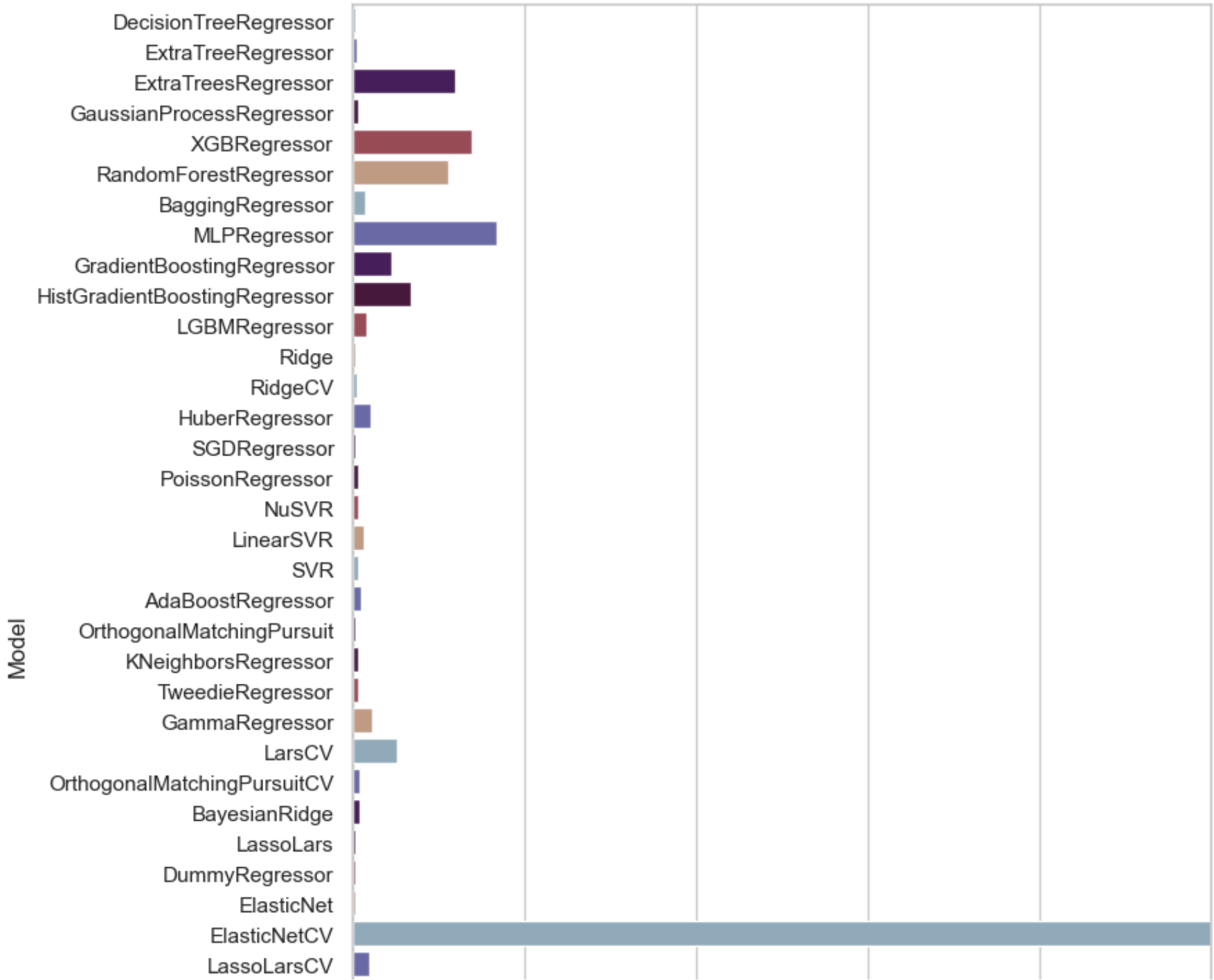

DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
RandomForestRegressor
BaggingRegressor
MLPRegressor
GradientBoostingRegressor
HistGradientBoostingRegressor
LGBMRegressor
Ridge
RidgeCV
HuberRegressor
SGDRegressor
PoissonRegressor
NuSVR
LinearSVR
SVR
AdaBoostRegressor
OrthogonalMatchingPursuit
KNeighborsRegressor
TweedieRegressor
GammaRegressor
LarsCV
OrthogonalMatchingPursuitCV
BayesianRidge
LassoLars
DummyRegressor
ElasticNet
ElasticNetCV
LassoLarsCV
LassoCV
Model

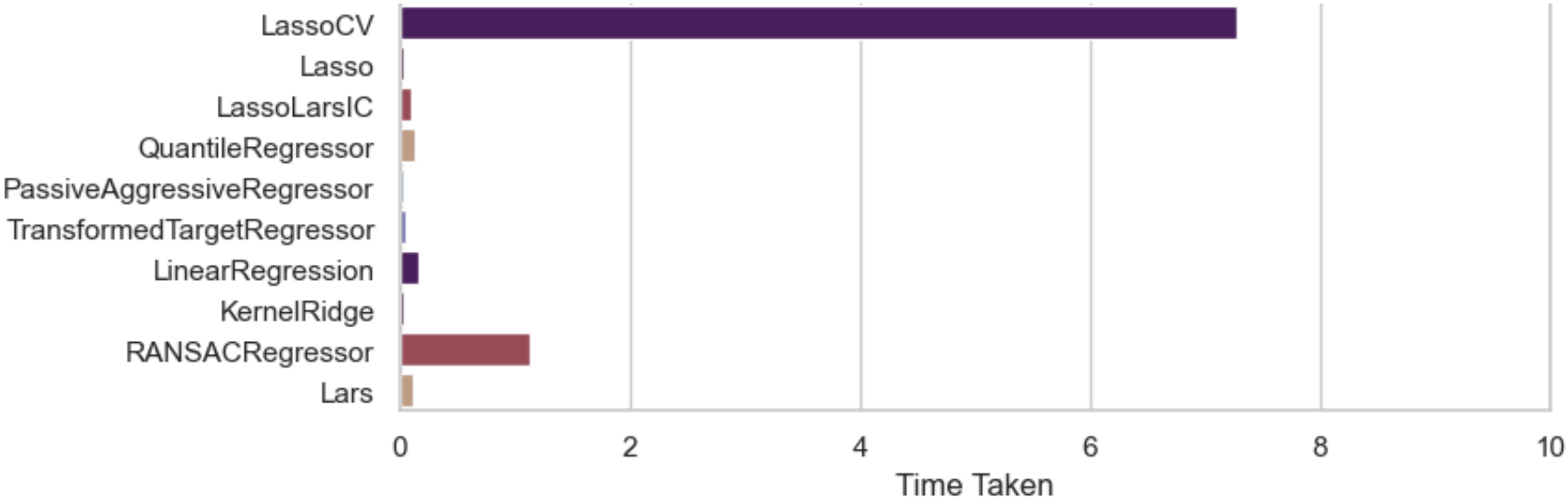


Bar plot of calculation time

In [628…

```
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()
```


DecisionTreeRegressor
ExtraTreeRegressor
ExtraTreesRegressor
GaussianProcessRegressor
XGBRegressor
RandomForestRegressor
BaggingRegressor
MLPRegressor
GradientBoostingRegressor
HistGradientBoostingRegressor
LGBMRegressor
Ridge
RidgeCV
HuberRegressor
SGDRegressor
PoissonRegressor
NuSVR
LinearSVR
SVR
AdaBoostRegressor
OrthogonalMatchingPursuit
KNeighborsRegressor
TweedieRegressor
GammaRegressor
LarsCV
OrthogonalMatchingPursuitCV
BayesianRidge
LassoLars
DummyRegressor
ElasticNet
ElasticNetCV
LassoLarsCV
Model


LassoCV
Lasso
LassoLarsIC
QuantileRegressor
PassiveAggressiveRegressor
TransformedTargetRegressor
LinearRegression
KernelRidge
RANSACRegressor
Lars
0
2
4
6
8
10
Time Taken

In [ ]: