In [2]:
```
import pandas as pd
from chembl_webresource_client.new_client import new_client
```

In [3]:
```
target = new_client.target
target_query = target.search('CHEMBL313')
targets = pd.DataFrame.from_dict(target_query)
targets
```

Out[3]:

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Rattus norvegicus	Serotonin transporter	14.0	False	CHEMBL313	[{'accession': 'P31652', 'component_descriptio...	SINGLE PROTEIN	10116
1	[]	Rattus norvegicus	Monoamine transporters; serotonin & dopamine	11.0	False	CHEMBL2094252	[{'accession': 'P31652', 'component_descriptio...	SELECTIVITY GROUP	10116
2	[]	Rattus norvegicus	Monoamine transporters; Norepinephrine & serot...	11.0	False	CHEMBL2096672	[{'accession': 'P31652', 'component_descriptio...	SELECTIVITY GROUP	10116

In [4]:
```
selected_target = targets.target_chembl_id[0]
selected_target
```

Out[4]:
```
'CHEMBL313'
```

In [5]:
```
activity = new_client.activity
res = activity.filter(target_chembl_id=selected_target).filter(standard_type="IC50")
```

In [6]:
```
df = pd.DataFrame.from_dict(res)
```

In [7]:
```
print(len(res))
```

```
1680
```

In [11]:
```
df.head()
```

Out[11]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	assay_
0	None	None	32389	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
1	None	None	32393	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
2	None	None	32520	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	
3	None	None	33547	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
4	None	None	33628	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	

5 rows × 46 columns

In [15]:
```
df.tail()
```

Out[15]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
1675	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966872	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1676	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966873	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1677	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966874	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1678	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966875	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1679	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966876	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	

5 rows × 46 columns

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1680 entries, 0 to 1679
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   action_type                 29 non-null     object
 1   activity_comment            23 non-null     object
 2   activity_id                 1680 non-null   int64
 3   activity_properties         1680 non-null   object
 4   assay_chembl_id             1680 non-null   object
 5   assay_description           1680 non-null   object
 6   assay_type                  1680 non-null   object
 7   assay_variant_accession     0 non-null      object
 8   assay_variant_mutation      0 non-null      object
 9   bao_endpoint                1680 non-null   object
 10  bao_format                  1680 non-null   object
 11  bao_label                   1680 non-null   object
 12  canonical_smiles            1680 non-null   object
 13  data_validity_comment       17 non-null     object
 14  data_validity_description   17 non-null     object
 15  document_chembl_id          1680 non-null   object
 16  document_journal            1677 non-null   object
 17  document_year               1680 non-null   int64
 18  ligand_efficiency           1115 non-null   object
 19  molecule_chembl_id          1680 non-null   object
 20  molecule_pref_name          164 non-null    object
 21  parent_molecule_chembl_id   1680 non-null   object
 22  pchembl_value               1523 non-null   object
 23  potential_duplicate         1680 non-null   int64
 24  qudt_units                  1669 non-null   object
 25  record_id                   1680 non-null   int64
 26  relation                    1657 non-null   object
 27  src_id                      1680 non-null   int64
 28  standard_flag               1680 non-null   int64
 29  standard_relation           1657 non-null   object
 30  standard_text_value         0 non-null      object
 31  standard_type               1680 non-null   object
 32  standard_units              1669 non-null   object
 33  standard_upper_value        0 non-null      object
 34  standard_value              1657 non-null   object
 35  target_chembl_id            1680 non-null   object
 36  target_organism             1680 non-null   object
```

```
 37  target_pref_name           1680 non-null   object
 38  target_tax_id              1680 non-null   object
 39  text_value                 0 non-null      object
 40  toid                       0 non-null      object
 41  type                       1680 non-null   object
 42  units                      1626 non-null   object
 43  uo_units                   1669 non-null   object
 44  upper_value                0 non-null      object
 45  value                      1657 non-null   object
dtypes: int64(6), object(40)
memory usage: 603.9+ KB
```

In [22]:
```
df.columns
```

Out[22]:
```
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 [25]:
```
df2 = df.dropna(subset=["standard_value", "canonical_smiles"])
```

In [27]:
```
df2
```

Out[27]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
0	None	None	32389	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
1	None	None	32393	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
2	None	None	32520	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	
3	None	None	33547	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
4	None	None	33628	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	
...	...	...	...	...	...	...	...	...	
1675	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966872	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1676	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966873	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1677	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966874	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	
1678	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966875	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
1679	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966876	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	

1657 rows × 46 columns

In [30]:
```
len(df2.canonical_smiles.unique())
```

Out[30]: 1370

In [34]:
```
df2_nr = df2.drop_duplicates(subset="canonical_smiles", keep="first").reset_index(drop=True)
```

In [37]:
```
df2_nr
```

Out[37]:

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
0	None	None	32389	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
1	None	None	32393	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
2	None	None	32520	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	
3	None	None	33547	[]	CHEMBL804187	Inhibition of [3H]peroxitine binding to rat co...	B	None	
4	None	None	33628	[]	CHEMBL806971	Binding affinity at serotonin transporter from...	B	None	
...	...	...	...	...	...	...	...	...	
1365	{'action_type': 'INHIBITOR', 'description': 'N...	None	23295742	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL4840616	Inhibition of serotonin transporter expressed ...	B	None	
1366	{'action_type': 'INHIBITOR', 'description': 'N...	None	23295743	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL4840616	Inhibition of serotonin transporter expressed ...	B	None	
1367	{'action_type': 'INHIBITOR', 'description': 'N...	None	24708348	[{'comments': None, 'relation': '=', 'result_f...	CHEMBL5109316	Displacement of [3H]-5HT from rat SERT express...	B	None	
1368	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966872	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	

	action_type	activity_comment	activity_id	activity_properties	assay_chembl_id	assay_description	assay_type	assay_variant_accession	as
1369	{'action_type': 'INHIBITOR', 'description': 'N...	None	24966873	[]	CHEMBL5217326	Inhibition of serotonin transporter in rat bra...	B	None	

1370 rows × 46 columns

In [41]:
```
df2_nr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1370 entries, 0 to 1369
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   action_type                 25 non-null     object
 1   activity_comment            0 non-null      object
 2   activity_id                 1370 non-null   int64
 3   activity_properties         1370 non-null   object
 4   assay_chembl_id             1370 non-null   object
 5   assay_description           1370 non-null   object
 6   assay_type                  1370 non-null   object
 7   assay_variant_accession     0 non-null      object
 8   assay_variant_mutation      0 non-null      object
 9   bao_endpoint                1370 non-null   object
 10  bao_format                  1370 non-null   object
 11  bao_label                   1370 non-null   object
 12  canonical_smiles            1370 non-null   object
 13  data_validity_comment       12 non-null     object
 14  data_validity_description   12 non-null     object
 15  document_chembl_id          1370 non-null   object
 16  document_journal            1367 non-null   object
 17  document_year               1370 non-null   int64
 18  ligand_efficiency           919 non-null    object
 19  molecule_chembl_id          1370 non-null   object
 20  molecule_pref_name          68 non-null     object
 21  parent_molecule_chembl_id   1370 non-null   object
 22  pchembl_value               1244 non-null   object
 23  potential_duplicate         1370 non-null   int64
 24  qudt_units                  1370 non-null   object
 25  record_id                   1370 non-null   int64
 26  relation                    1370 non-null   object
 27  src_id                      1370 non-null   int64
 28  standard_flag               1370 non-null   int64
 29  standard_relation           1370 non-null   object
 30  standard_text_value         0 non-null      object
 31  standard_type               1370 non-null   object
 32  standard_units              1370 non-null   object
 33  standard_upper_value        0 non-null      object
 34  standard_value              1370 non-null   object
 35  target_chembl_id            1370 non-null   object
 36  target_organism             1370 non-null   object
```

```
 37  target_pref_name           1370 non-null   object
 38  target_tax_id              1370 non-null   object
 39  text_value                 0 non-null      object
 40  toid                       0 non-null      object
 41  type                       1370 non-null   object
 42  units                      1328 non-null   object
 43  uo_units                   1370 non-null   object
 44  upper_value                0 non-null      object
 45  value                      1370 non-null   object
dtypes: int64(6), object(40)
memory usage: 492.5+ KB
```

In [45]:
```
df2_nr .columns
```

Out[45]:
```
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 a DataFrame

In [49]:
```
selection = ['molecule_chembl_id','canonical_smiles','standard_value']
df3 = df2_nr[selection]
df3
```

Out[49]:

	molecule_chembl_id	canonical_smiles	standard_value
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.4
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.0
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.0
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.8
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.0
...	...	...	...
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	4.2
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	0.53
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	740.0
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	8.2
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	17.0

1370 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 [53]:
```
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 [57]:
```
bioactivity_class = pd.Series(bioactivity_threshold, name='class')
df5 = pd.concat([df3, bioactivity_class], axis=1)
df5
```

Out[57]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.4	active
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.0	active
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.0	active
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.8	active
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.0	active
...	...	...	...	...
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	4.2	active
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	0.53	active
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	740.0	active
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	8.2	active
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	17.0	active

1370 rows × 4 columns

In [65]:
```
df5.to_csv('Serotonin_03_bioactivity_data_curated.csv', index=False)
```

In [67]:
```
df = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\Serotonin\Serotonin_03_bioactivity_data_curated.csv")
df
```

Out[67]:

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.40	active
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.00	active
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.00	active
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.80	active
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.00	active
...	...	...	...	...
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	4.20	active
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	0.53	active
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	740.00	active
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	8.20	active
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	17.00	active

1370 rows × 4 columns

In [70]:
```
df_no_smiles = df.drop(columns='canonical_smiles')
```

In [73]:
```
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 [75]:
```
df_clean_smiles = pd.concat([df_no_smiles,smiles], axis=1)
df_clean_smiles
```

Out[75]:

	molecule_chembl_id	standard_value	class	canonical_smiles
0	CHEMBL18501	1.40	active	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...
1	CHEMBL19226	35.00	active	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...
2	CHEMBL162006	761.00	active	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1
3	CHEMBL19203	0.80	active	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...
4	CHEMBL351215	102.00	active	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1
...	...	...	...	...
1365	CHEMBL4851065	4.20	active	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...
1366	CHEMBL4864918	0.53	active	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...
1367	CHEMBL5175117	740.00	active	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...
1368	CHEMBL5220371	8.20	active	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...
1369	CHEMBL5220872	17.00	active	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...

1370 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 [79]:
```
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors, Lipinski
```

Calculate descriptors

In [84]:
```
# 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 [87]:
```
df_lipinski = lipinski(df_clean_smiles.canonical_smiles)
df_lipinski
```

Out[87]:

	MW	LogP	NumHDonors	NumHAcceptors
0	477.605	5.31370	1.0	3.0
1	489.616	5.43740	1.0	3.0
2	407.504	5.98300	0.0	2.0
3	457.599	5.39930	1.0	3.0
4	399.578	6.13740	0.0	2.0
...	...	...	...	...
1365	477.568	4.69148	1.0	6.0
1366	491.595	5.08158	1.0	6.0
1367	583.685	4.47470	5.0	6.0
1368	445.941	5.17410	2.0	2.0
1369	462.396	5.68840	2.0	2.0

1370 rows × 4 columns

Combine DataFrames

In [95]:
```
df_lipinski
```

Out[95]:

	MW	LogP	NumHDonors	NumHAcceptors
0	477.605	5.31370	1.0	3.0
1	489.616	5.43740	1.0	3.0
2	407.504	5.98300	0.0	2.0
3	457.599	5.39930	1.0	3.0
4	399.578	6.13740	0.0	2.0
...	...	...	...	...
1365	477.568	4.69148	1.0	6.0
1366	491.595	5.08158	1.0	6.0
1367	583.685	4.47470	5.0	6.0
1368	445.941	5.17410	2.0	2.0
1369	462.396	5.68840	2.0	2.0

1370 rows × 4 columns

In [97]:
```
df_lipinski.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1370 entries, 0 to 1369
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   MW              1370 non-null   float64
 1   LogP            1370 non-null   float64
 2   NumHDonors      1370 non-null   float64
 3   NumHAcceptors   1370 non-null   float64
dtypes: float64(4)
memory usage: 42.9 KB
```

In [99]:
```
df_lipinski.describe()
```

Out[99]:

	MW	LogP	NumHDonors	NumHAcceptors
count	1370.000000	1370.000000	1370.000000	1370.000000
mean	358.400109	4.277567	0.734307	2.997810
std	92.309515	1.352354	0.921018	1.281148
min	159.232000	-0.968500	0.000000	1.000000
25%	291.438000	3.371600	0.000000	2.000000
50%	359.147500	4.233850	1.000000	3.000000
75%	424.972000	5.146900	1.000000	4.000000
max	952.075000	10.314500	12.000000	13.000000

In [102…

```
df
```

Out[102...

	molecule_chembl_id	canonical_smiles	standard_value	class
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.40	active
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.00	active
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.00	active
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.80	active
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.00	active
...	...	...	...	...
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	4.20	active
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	0.53	active
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	740.00	active
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	8.20	active
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	17.00	active

1370 rows × 4 columns

In [104...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1370 entries, 0 to 1369
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  1370 non-null   object 
 1   canonical_smiles    1370 non-null   object 
 2   standard_value      1370 non-null   float64
 3   class               1370 non-null   object 
dtypes: float64(1), object(3)
memory usage: 42.9+ KB
```

In [106...

```
df.describe()
```

Out[106…

	standard_value
count	1.370000e+03
mean	9.394356e+03
std	1.033348e+05
min	3.000000e-02
25%	4.713500e+01
50%	4.970000e+02
75%	2.237500e+03
max	3.600000e+06

combine the 2 DataFrame

In [205…

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

In [207…

```
df_combined
```

Out[207...

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHA
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.40	active	477.605	5.31370	1.0	
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.00	active	489.616	5.43740	1.0	
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.00	active	407.504	5.98300	0.0	
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.80	active	457.599	5.39930	1.0	
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.00	active	399.578	6.13740	0.0	
...	...	...	...	...	...	...	...	
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	4.20	active	477.568	4.69148	1.0	
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	0.53	active	491.595	5.08158	1.0	
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	740.00	active	583.685	4.47470	5.0	
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	8.20	active	445.941	5.17410	2.0	
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	17.00	active	462.396	5.68840	2.0	

1370 rows × 8 columns

1370 rows × 8 columns

In [210...

```
df_combined.head()
```

Out[210...

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHAccepto
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	1.4	active	477.605	5.3137	1.0	
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	35.0	active	489.616	5.4374	1.0	
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	761.0	active	407.504	5.9830	0.0	
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	0.8	active	457.599	5.3993	1.0	
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	102.0	active	399.578	6.1374	0.0	

In [212...

```
df_combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1370 entries, 0 to 1369
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   molecule_chembl_id  1370 non-null   object 
 1   canonical_smiles    1370 non-null   object 
 2   standard_value      1370 non-null   float64
 3   class               1370 non-null   object 
 4   MW                  1370 non-null   float64
 5   LogP                1370 non-null   float64
 6   NumHDonors          1370 non-null   float64
 7   NumHAcceptors       1370 non-null   float64
dtypes: float64(5), object(3)
memory usage: 85.8+ KB
```

In [214...

```
df_combined.describe()
```

Out[214...

	standard_value	MW	LogP	NumHDonors	NumHAcceptors
count	1.370000e+03	1370.000000	1370.000000	1370.000000	1370.000000
mean	9.394356e+03	358.400109	4.277567	0.734307	2.997810
std	1.033348e+05	92.309515	1.352354	0.921018	1.281148
min	3.000000e-02	159.232000	-0.968500	0.000000	1.000000
25%	4.713500e+01	291.438000	3.371600	0.000000	2.000000
50%	4.970000e+02	359.147500	4.233850	1.000000	3.000000
75%	2.237500e+03	424.972000	5.146900	1.000000	4.000000
max	3.600000e+06	952.075000	10.314500	12.000000	13.000000

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 [217...

```
# 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 [219…
```
df_combined.standard_value.describe()
```

Out[219…
```
count    1.370000e+03
mean     9.394356e+03
std      1.033348e+05
min      3.000000e-02
25%      4.713500e+01
50%      4.970000e+02
75%      2.237500e+03
max      3.600000e+06
Name: standard_value, dtype: float64
```

In [221…
```
-np.log10( (10**-9)* 100000000 )
```

Out[221…
```
1.0
```

In [223…
```
-np.log10( (10**-9)* 10000000000 )
```

Out[223…
```
-1.0
```

In [225…
```
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 [227…
```
df_norm = norm_value(df_combined)
df_norm
```

Out[227…

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	standa
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	active	477.605	5.31370	1.0	3.0	
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	active	489.616	5.43740	1.0	3.0	
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	active	407.504	5.98300	0.0	2.0	
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	active	457.599	5.39930	1.0	3.0	
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	active	399.578	6.13740	0.0	2.0	
...	...	...	...	...	...	...	...	
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	active	477.568	4.69148	1.0	6.0	
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	active	491.595	5.08158	1.0	6.0	
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	active	583.685	4.47470	5.0	6.0	
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	active	445.941	5.17410	2.0	2.0	
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	active	462.396	5.68840	2.0	2.0	

1370 rows × 8 columns

In [229…
```
df_norm.standard_value_norm
```

Out[229…
```
0         1.40
1        35.00
2       761.00
3         0.80
4       102.00
         ...
1365      4.20
1366      0.53
1367    740.00
1368      8.20
1369     17.00
Name: standard_value_norm, Length: 1370, dtype: float64
```

In [236…
```
df_norm.standard_value_norm.describe()
```

Out[236…
```
count    1.370000e+03
mean     9.394356e+03
std      1.033348e+05
min      3.000000e-02
25%      4.713500e+01
50%      4.970000e+02
75%      2.237500e+03
max      3.600000e+06
Name: standard_value_norm, dtype: float64
```

In [238…
```
print(df_norm.columns)
```

```
Index(['molecule_chembl_id', 'canonical_smiles', 'class', 'MW', 'LogP',
       'NumHDonors', 'NumHAcceptors', 'standard_value_norm'],
      dtype='object')
```

In [259…
```
df_final = df_norm.drop('standard_value_norm', axis=1)
df_final
```

Out[259...

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC5
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	active	477.605	5.31370	1.0	3.0	8.8538
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	active	489.616	5.43740	1.0	3.0	7.4559
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	active	407.504	5.98300	0.0	2.0	6.1186
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	active	457.599	5.39930	1.0	3.0	9.0969
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	active	399.578	6.13740	0.0	2.0	6.9914
...	...	...	...	...	...	...	...	
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	active	477.568	4.69148	1.0	6.0	8.3767
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	active	491.595	5.08158	1.0	6.0	9.2757
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	active	583.685	4.47470	5.0	6.0	6.1307
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	active	445.941	5.17410	2.0	2.0	8.0861
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	active	462.396	5.68840	2.0	2.0	7.7695

1370 rows × 8 columns

In [264...

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

Out[264...

```
count    1370.000000
mean        6.493037
std         1.267606
min         2.443697
25%         5.650257
50%         6.303647
75%         7.326662
max        10.522879
Name: pIC50, dtype: float64
```

In [271...

```
df_final.columns
```

Out[271…
```
Index(['molecule_chembl_id', 'canonical_smiles', 'class', 'MW', 'LogP',
       'NumHDonors', 'NumHAcceptors', 'pIC50'],
      dtype='object')
```

Removing the 'intermediate' bioactivity class

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

In [275…
```
df_2class = df_final[df_final["class"]  != 'intermediate']
df_2class
```

Out[275…

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC5
0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	active	477.605	5.31370	1.0	3.0	8.8538
1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	active	489.616	5.43740	1.0	3.0	7.4559
2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	active	407.504	5.98300	0.0	2.0	6.1186
3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	active	457.599	5.39930	1.0	3.0	9.0969
4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	active	399.578	6.13740	0.0	2.0	6.9914
...	...	...	...	...	...	...	...	
1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	active	477.568	4.69148	1.0	6.0	8.3767
1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	active	491.595	5.08158	1.0	6.0	9.2757
1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	active	583.685	4.47470	5.0	6.0	6.1307
1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	active	445.941	5.17410	2.0	2.0	8.0861
1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	active	462.396	5.68840	2.0	2.0	7.7695

1009 rows × 8 columns

In [428…
```
df_2class.to_csv('Serotonin_04_bioactivity_data_3class_pIC50.csv')
```

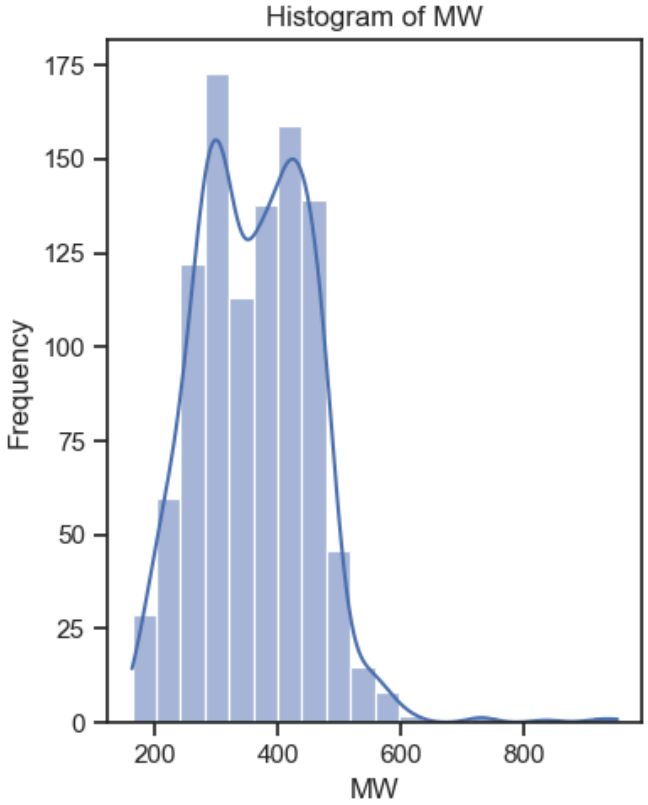
Exploratory Data Analysis (Chemical Space Analysis) via Lipinski descriptors

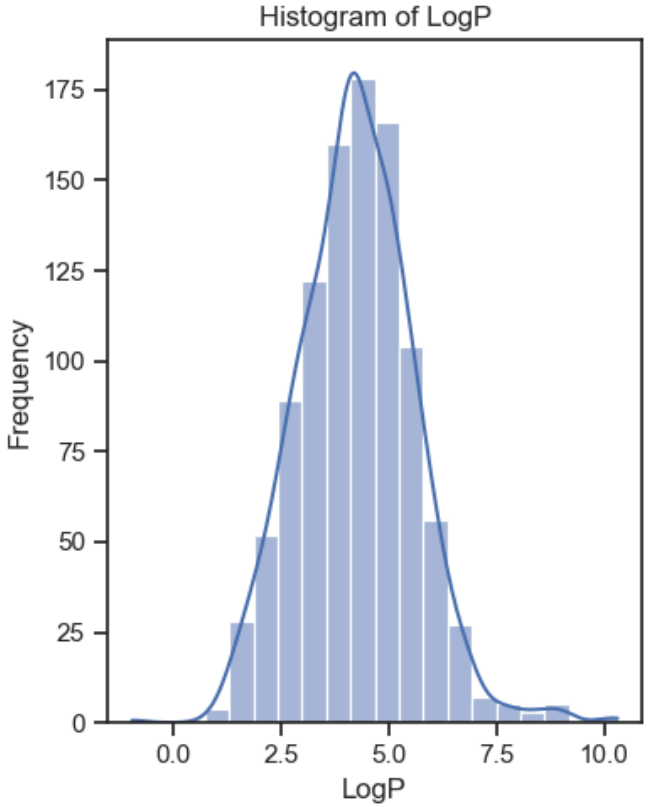
In [284…

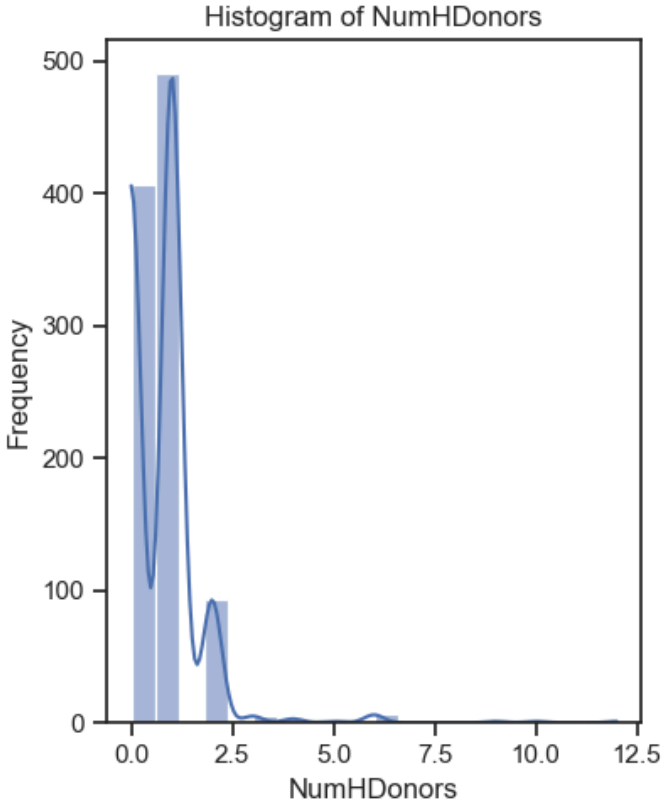
```
import seaborn as sns
sns.set(style='ticks')
import matplotlib.pyplot as plt
```

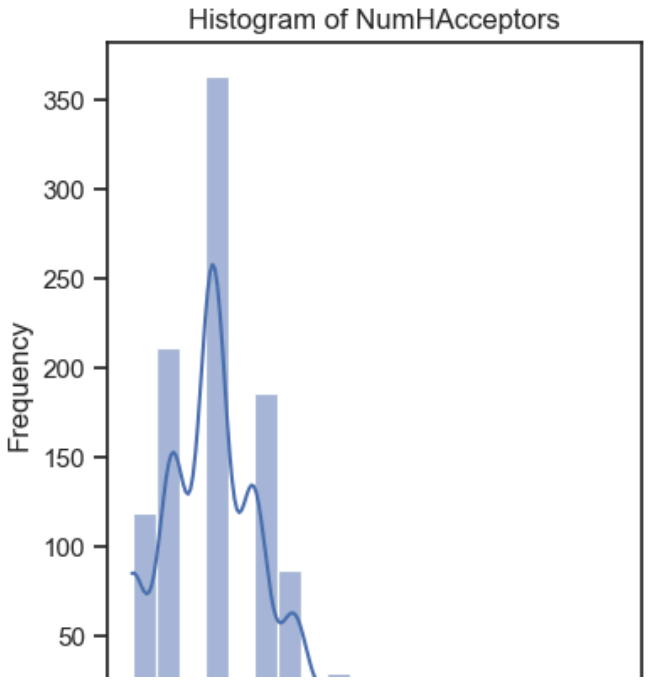
In [286…

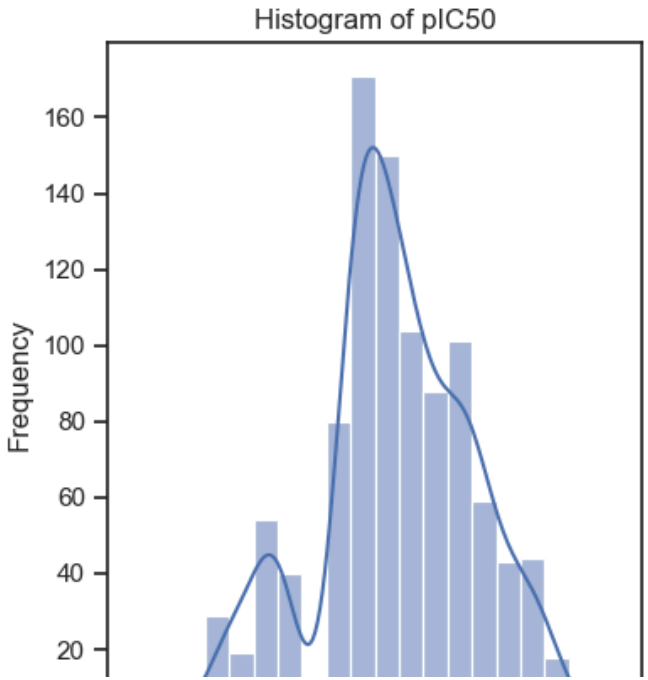
```
plt.figure(figsize=(12, 10))
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors', 'pIC50']
for i, column in enumerate(numeric_columns, 1):
 plt.subplot(2, 3, i) # 2 rows and 3 columns for better layout
 sns.histplot(df_2class[column], kde=True, bins=20) # kde=True for kernel density estimate
 plt.title(f'Histogram of {column}')
 plt.xlabel(column)
 plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

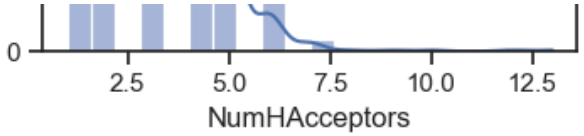

Histogram of MW
Frequency
MW
0
25
50
75
100
125
150
175
200
400
600
800

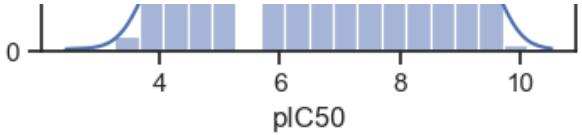

Histogram of LogP
Frequency
LogP
0
25
50
75
100
125
150
175
0.0
2.5
5.0
7.5
10.0


Histogram of NumHDonors
Frequency
NumHDonors
0
100
200
300
400
500
0.0
2.5
5.0
7.5
10.0
12.5


Histogram of NumHAcceptors
Frequency
50
100
150
200
250
300
350


Histogram of pIC50
Frequency
20
40
60
80
100
120
140
160

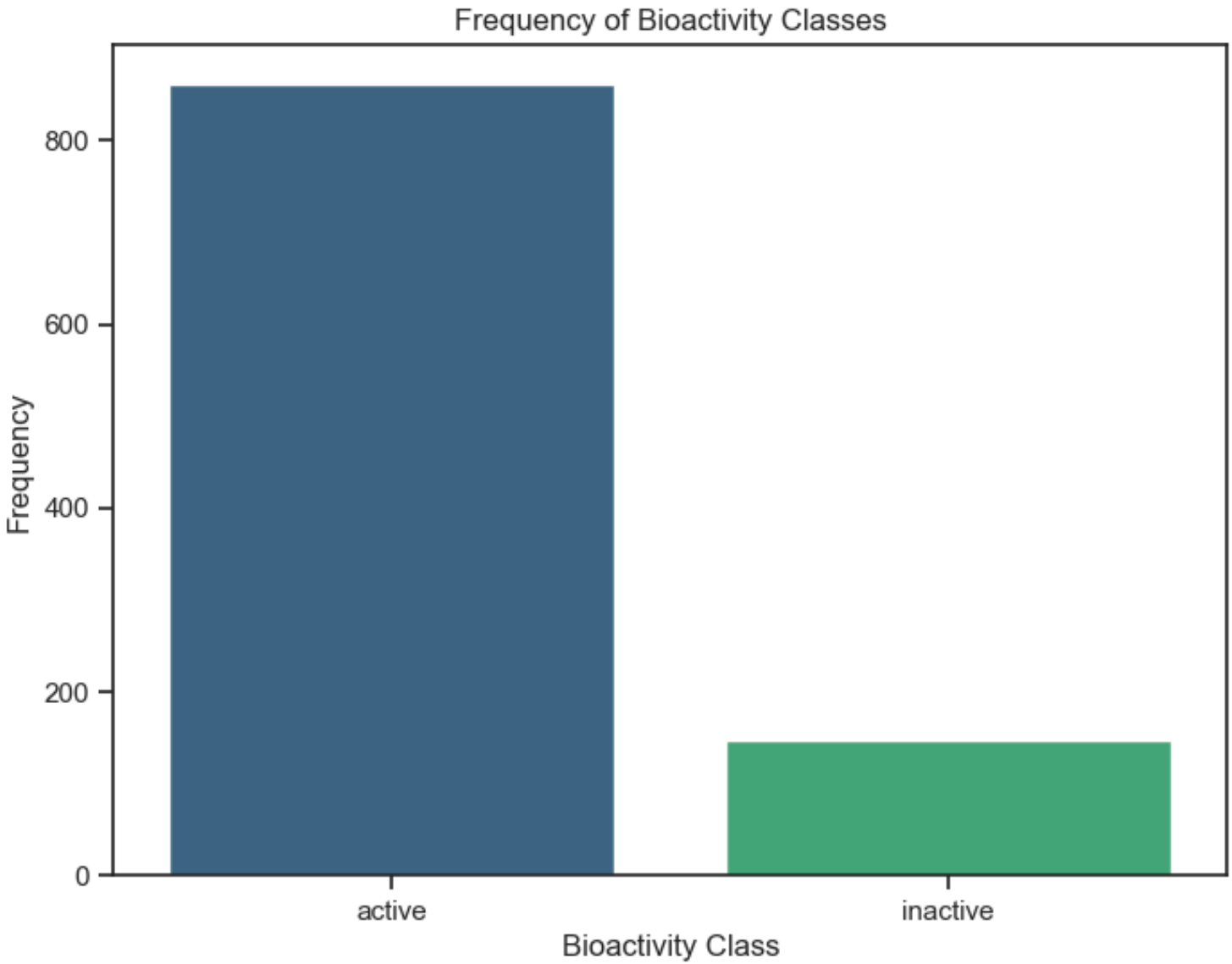




Frequency of Bioactivity Classes

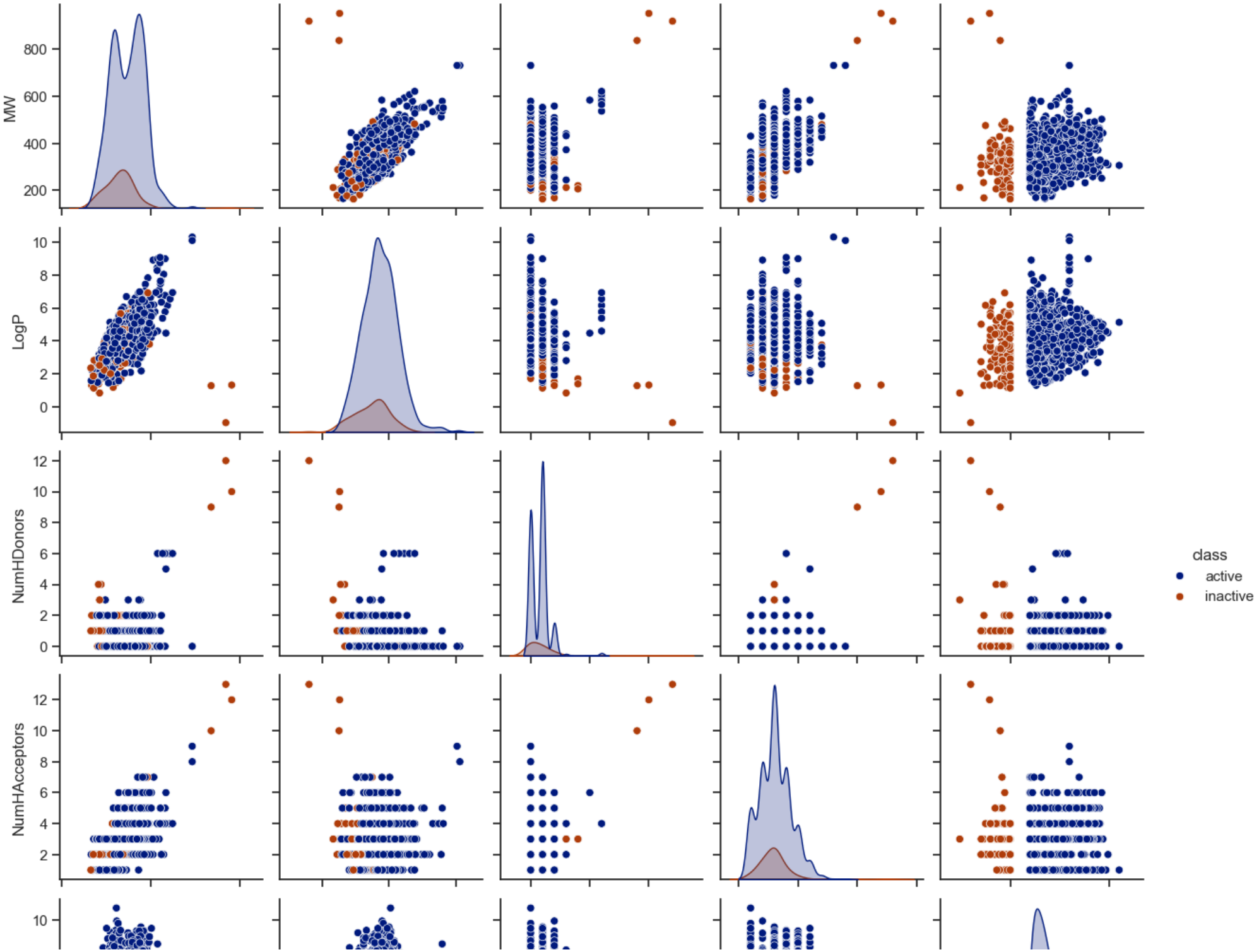
In [292…

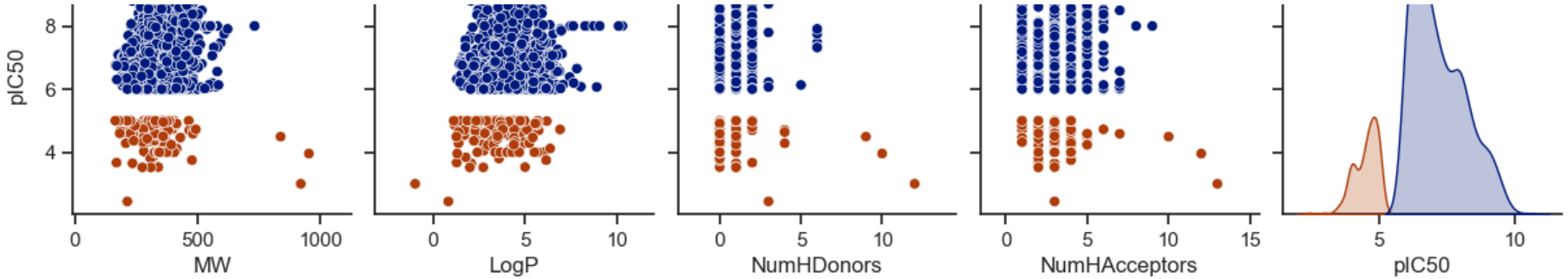
```
# Set figure size
plt.figure(figsize=(8, 6))
# Count plot with hue
sns.countplot(x='class', hue='class', data=df_2class, palette='viridis')
# Add labels and title
plt.xlabel('Bioactivity Class')
plt.ylabel('Frequency')
plt.title('Frequency of Bioactivity Classes')
# Show plot
plt.show()
```



In [297…

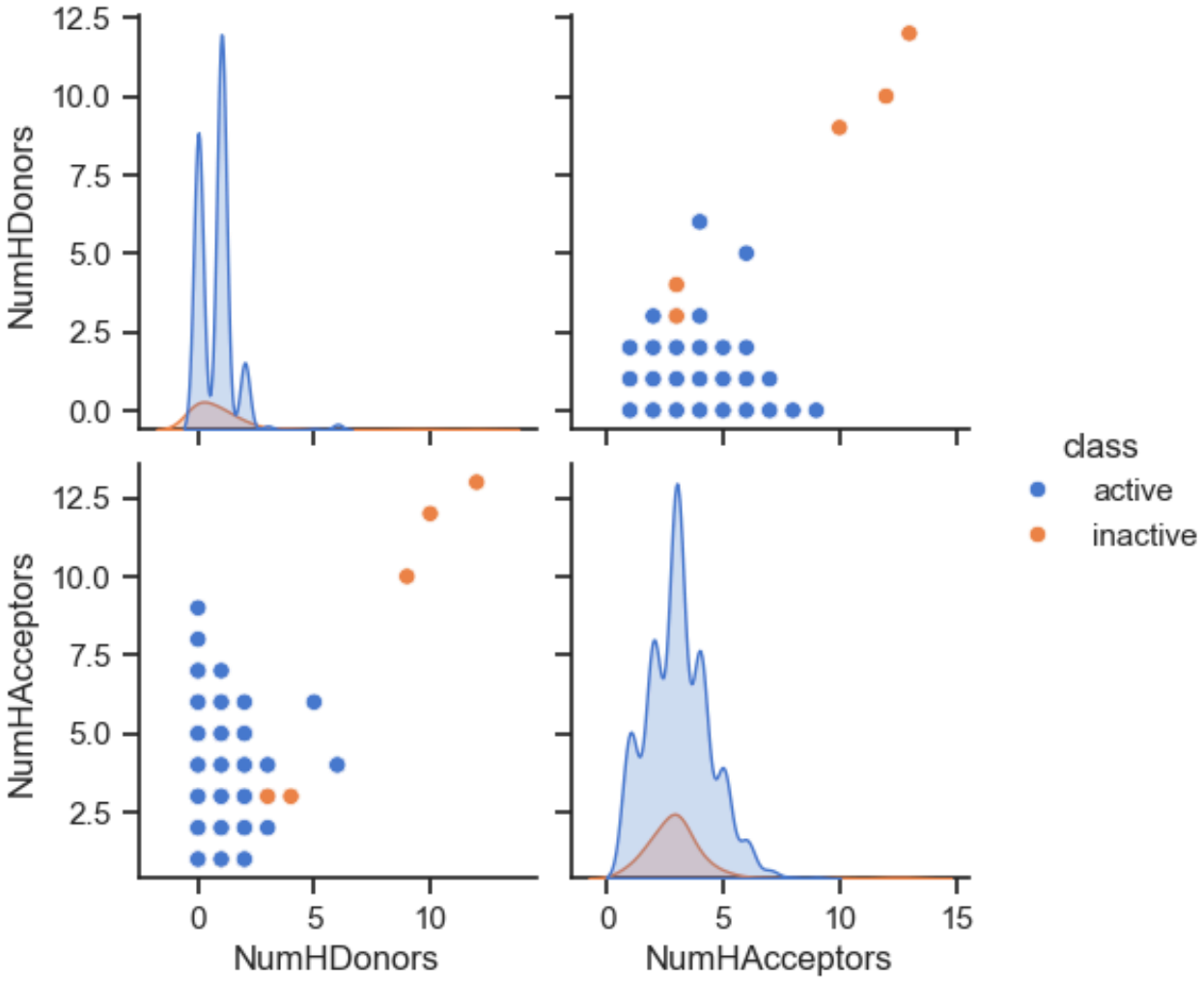
```
sns.pairplot(df_2class, hue='class', palette='dark')
# Show plot
plt.show()
```


MW
LogP
NumHDonors
NumHAcceptors
class
active
inactive



In [299…

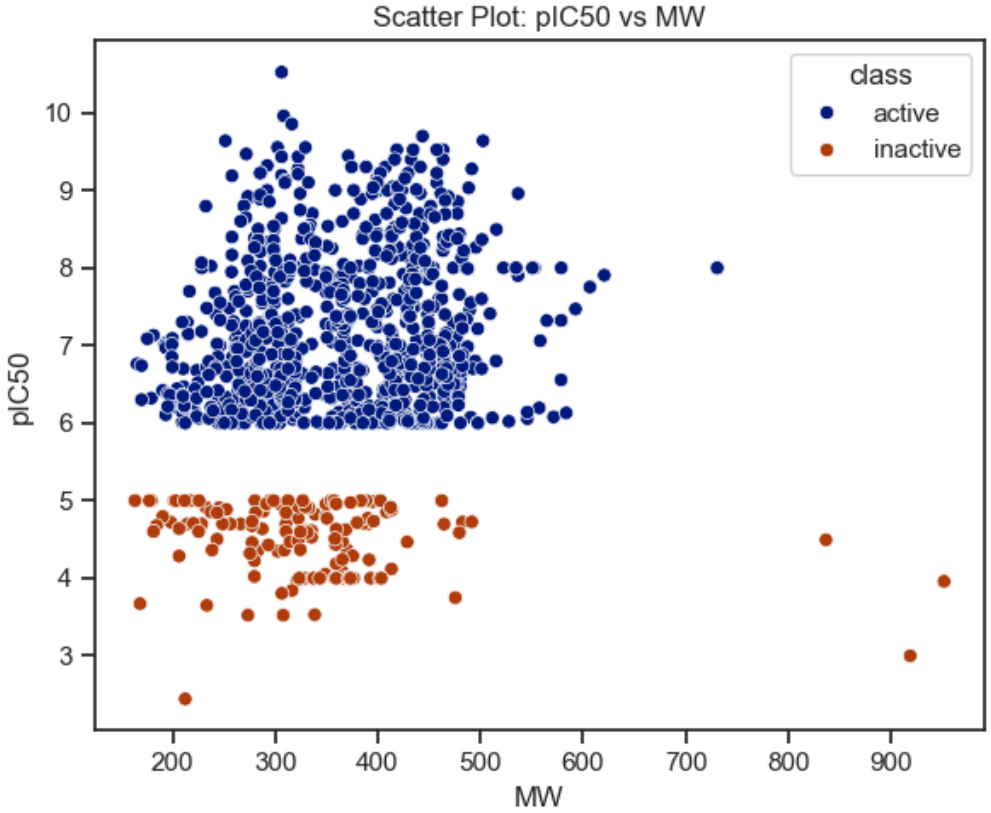
```
sns.pairplot(df_2class, vars=['NumHDonors', 'NumHAcceptors'], hue='class', palette='muted')
# Show plot
plt.show()
```

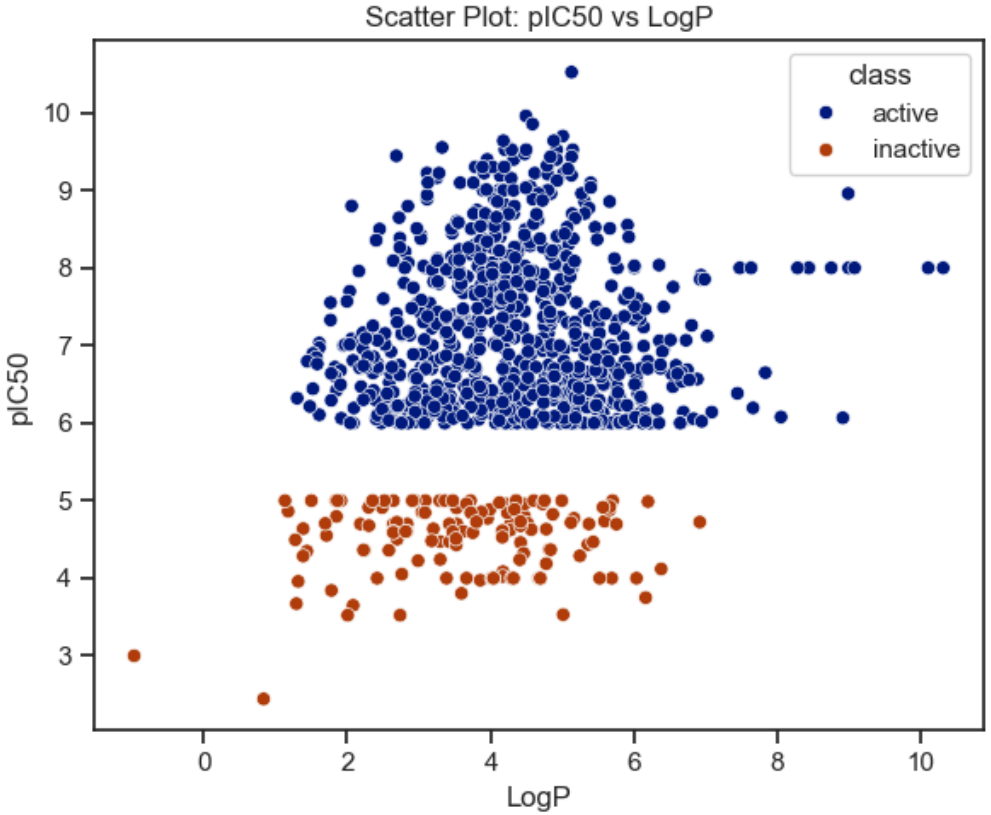


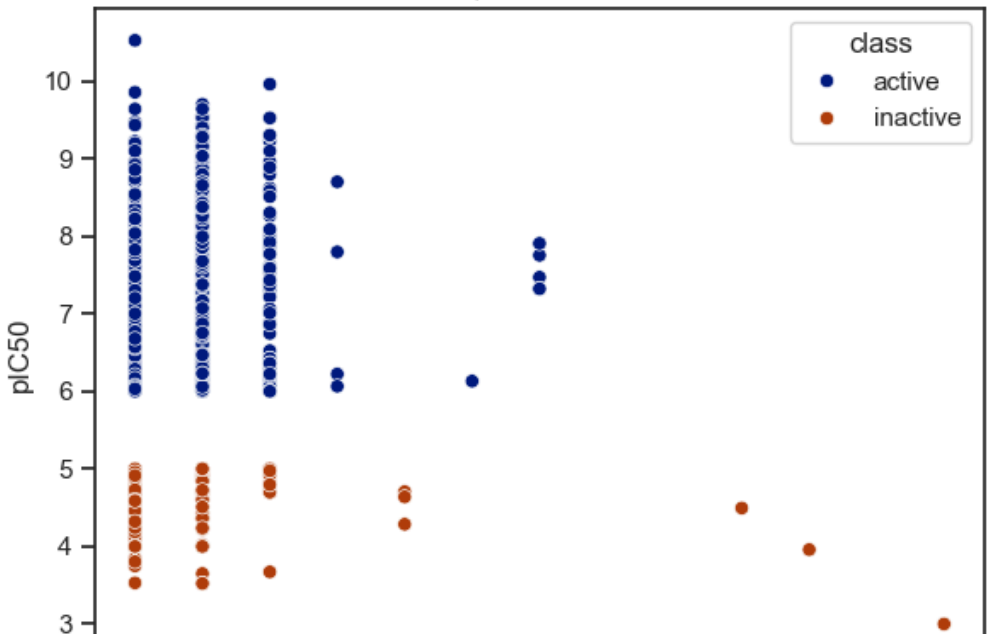
In [303…

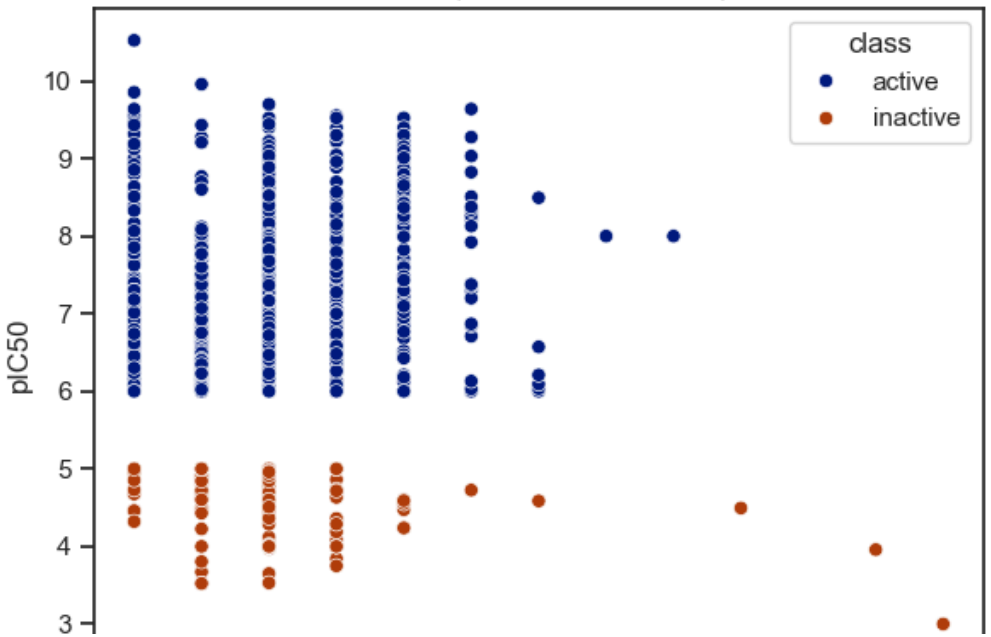
```
numeric_columns = ['MW', 'LogP', 'NumHDonors', 'NumHAcceptors']
# Set figure size
plt.figure(figsize=(12, 10))
# Loop through numeric columns and create scatter plots
for i, column in enumerate(numeric_columns, 1):
 plt.subplot(2, 2, i) # 2 rows, 2 columns layout
 sns.scatterplot(x=df_2class[column],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

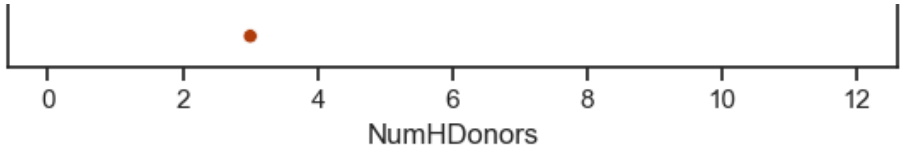
```
plt.tight_layout()
plt.show()
```

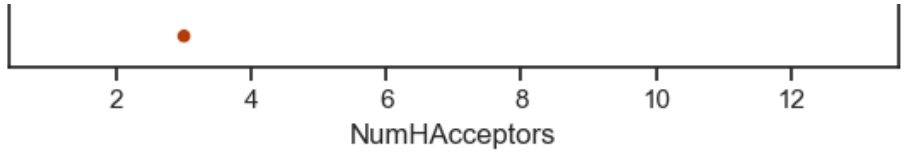

Scatter Plot: pIC50 vs MW
class
active
inactive
pIC50
MW
10
9
8
7
6
5
4
3
200
300
400
500
600
700
800
900


Scatter Plot: pIC50 vs LogP
class
active
inactive
pIC50
LogP
10
9
8
7
6
5
4
3
0
2
4
6
8
10


Scatter Plot: pIC50 vs NumHDonors
class
active
inactive
pIC50
10
9
8
7
6
5
4
3

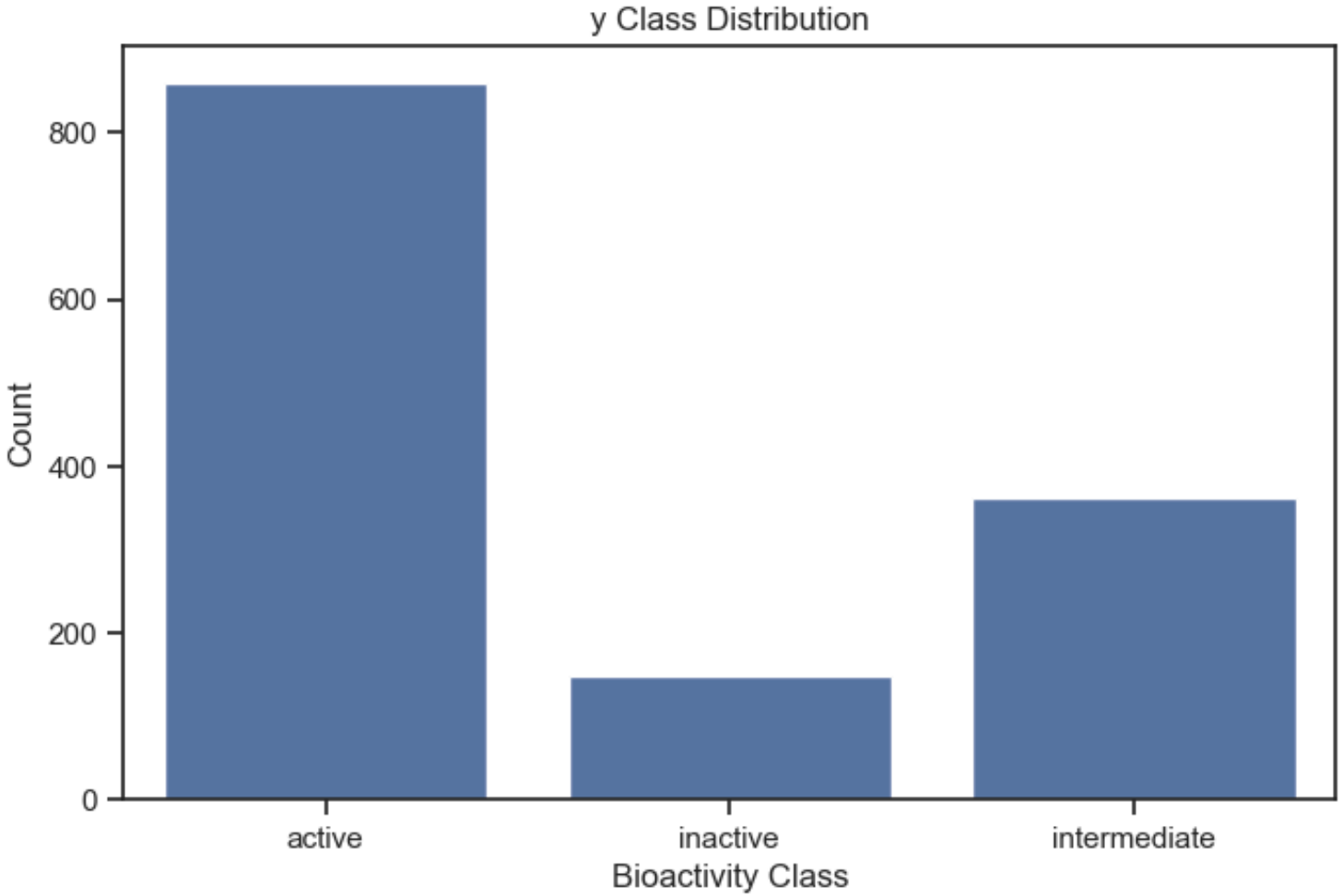

Scatter Plot: pIC50 vs NumHAcceptors
class
active
inactive
pIC50
10
9
8
7
6
5
4
3





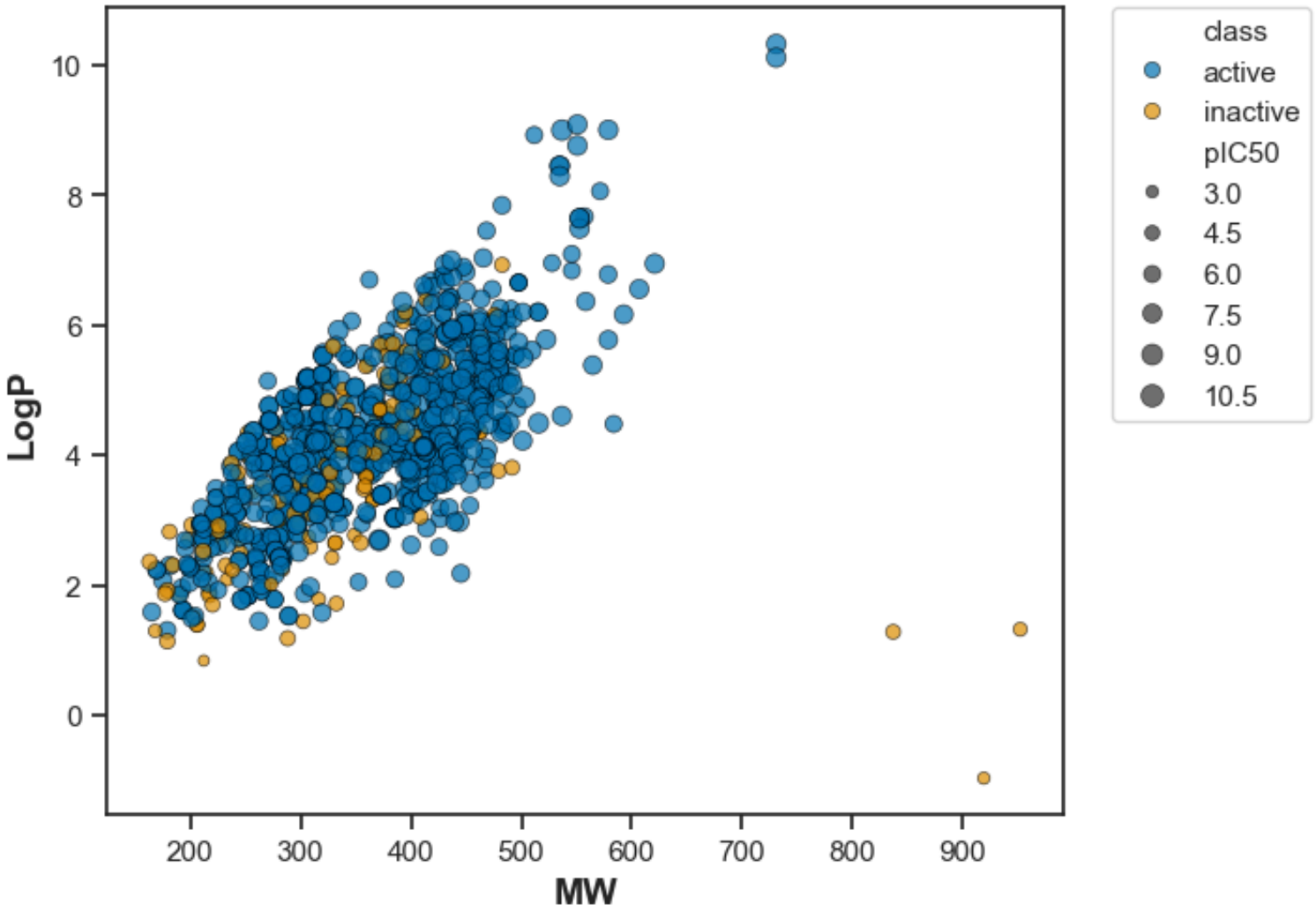
In [307…

```
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 [310…

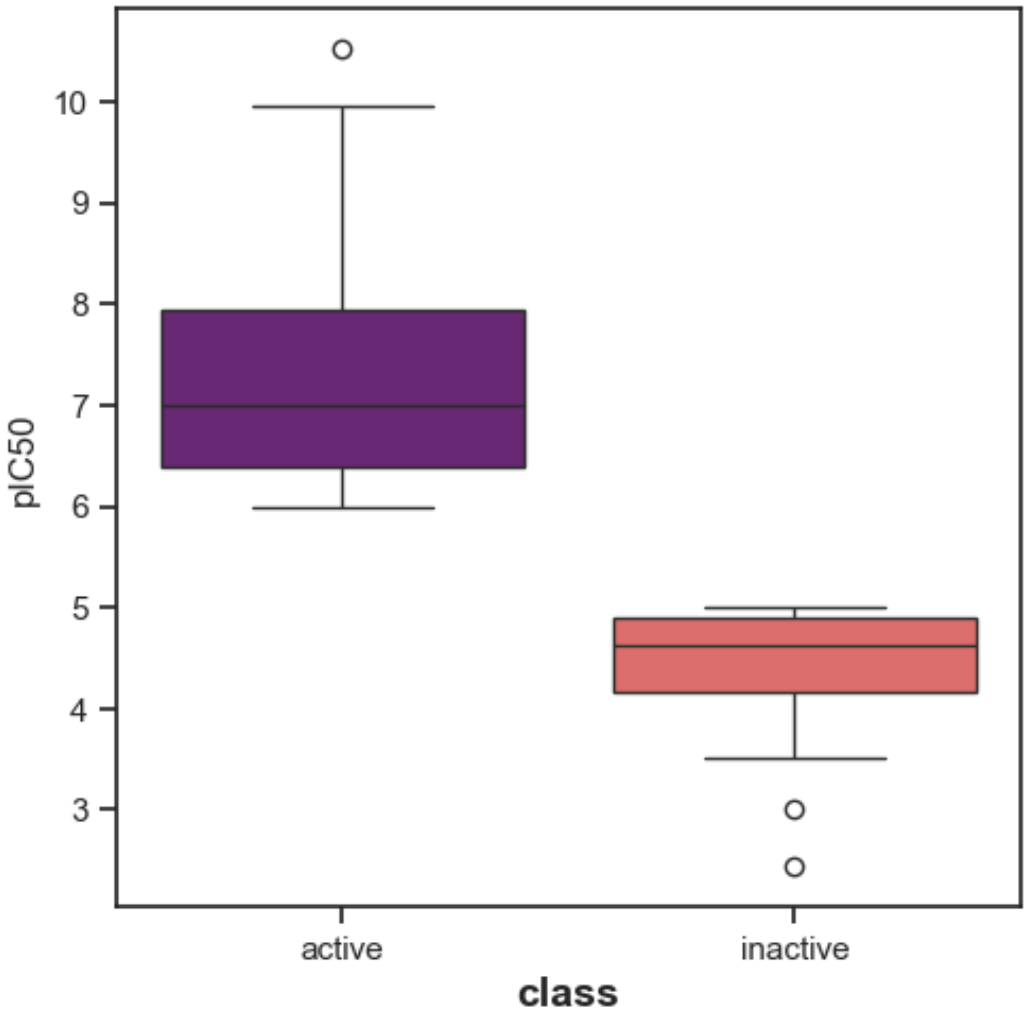
```
plt.figure(figsize=(6.6, 5.6))
sns.scatterplot(x='MW', y='LogP', data=df_2class, hue='class', size='pIC50',palette='colorblind', edgecolor='black', alpha=0.7)
plt.xlabel('MW', fontsize=14, fontweight='bold')
plt.ylabel('LogP', fontsize=14, fontweight='bold')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,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



In [325…

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

Out[325… Text(0.5, 0, 'class')



Statistical analysis | Mann-Whitney U Test

In [352…

```
 # 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 [354…
```
df_2class.columns
```

Out[354…
```
Index(['molecule_chembl_id', 'canonical_smiles', 'class', 'MW', 'LogP',
       'NumHDonors', 'NumHAcceptors', 'pIC50'],
      dtype='object')
```

In [357…
```
mannwhitney('pIC50')
```

Out[357...

	Descriptor	Statistics	p	alpha	Interpretation
0	pIC50	127428.0	2.584761e-84	0.05	Different distribution (reject H0)

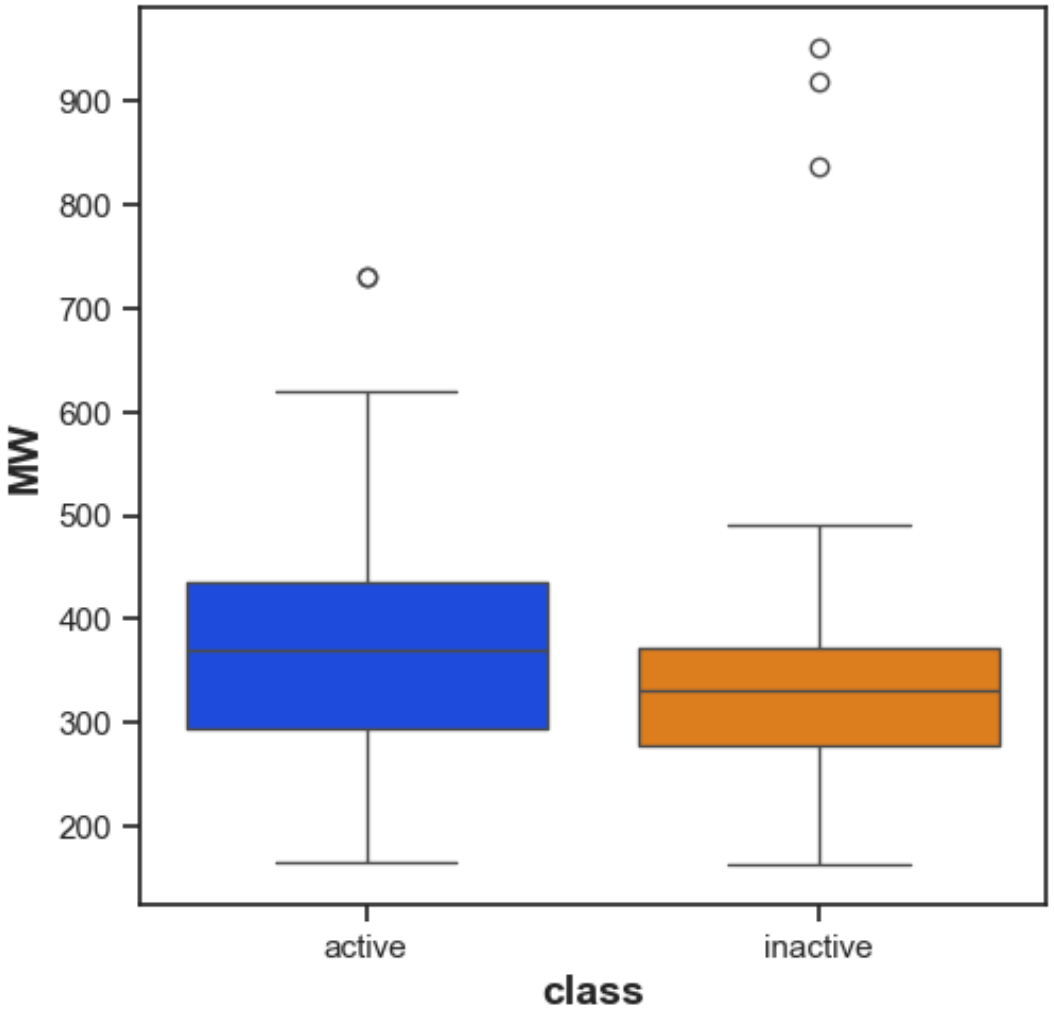
MW

In [372...

```
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[372...

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



In [377…
```
mannwhitney('MW')
```

Out[377…

	Descriptor	Statistics	p	alpha	Interpretation
0	MW	81119.5	1.068780e-07	0.05	Different distribution (reject H0)

LogP

In [386…

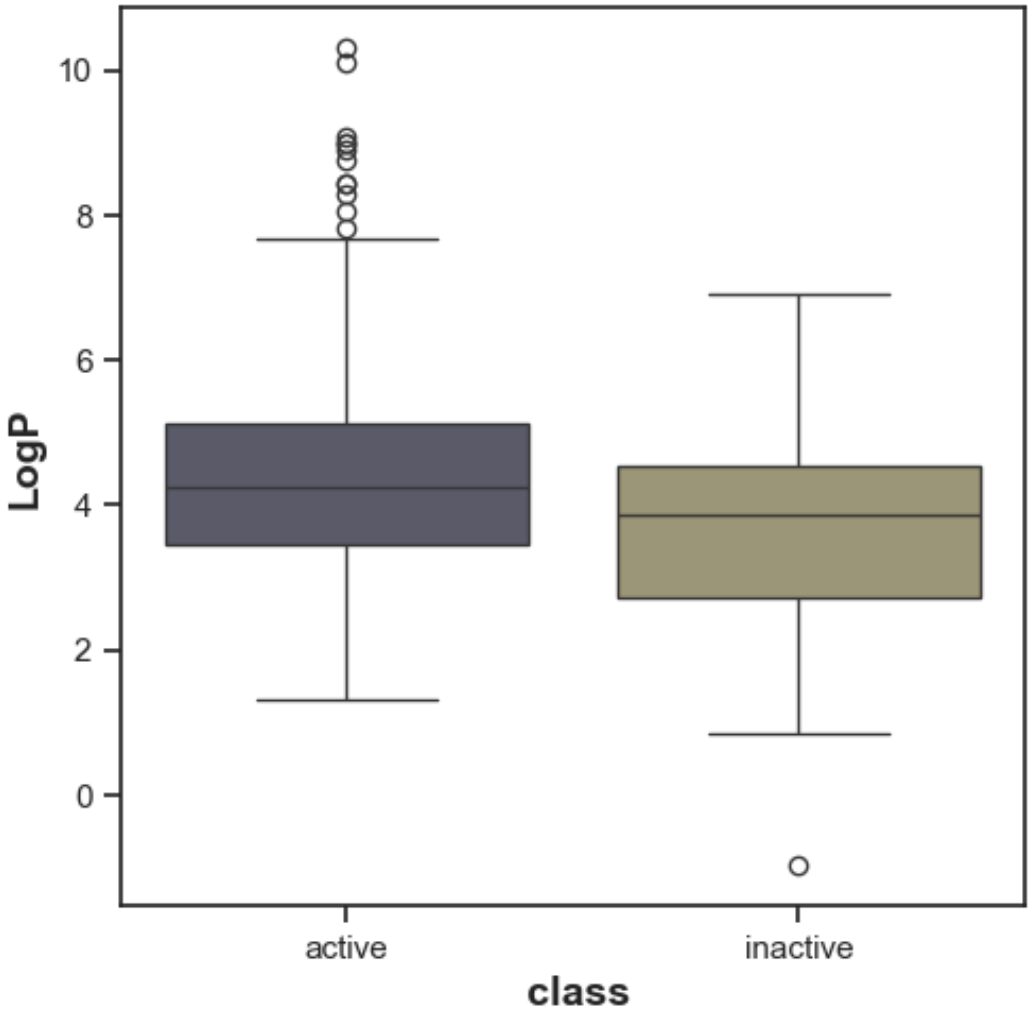
```
plt.figure(figsize=(5.5, 5.5))

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

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

Out[386…

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



In [390…
```
mannwhitney('LogP')
```

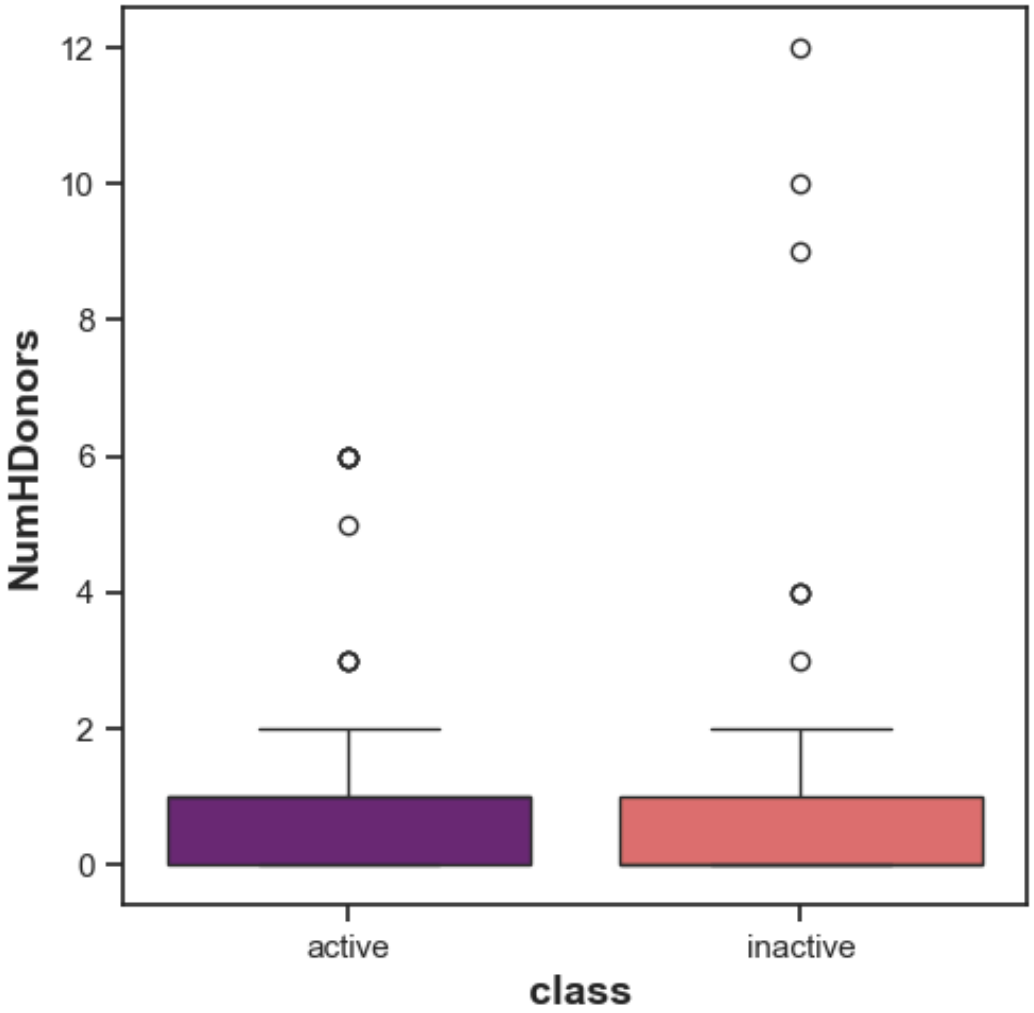
Out[390…

	Descriptor	Statistics	p	alpha	Interpretation
0	LogP	79842.5	8.449541e-07	0.05	Different distribution (reject H0)

NumHDonors

In [401…
```
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[401…
```
Text(0, 0.5, 'NumHDonors')
<Figure size 550x550 with 0 Axes>
```



Statistical analysis | Mann-Whitney U Test

In [409…
```
mannwhitney('NumHDonors')
```

Out[409…

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHDonors	67788.0	0.16914	0.05	Same distribution (fail to reject H0)

NumHAcceptors

In [414…

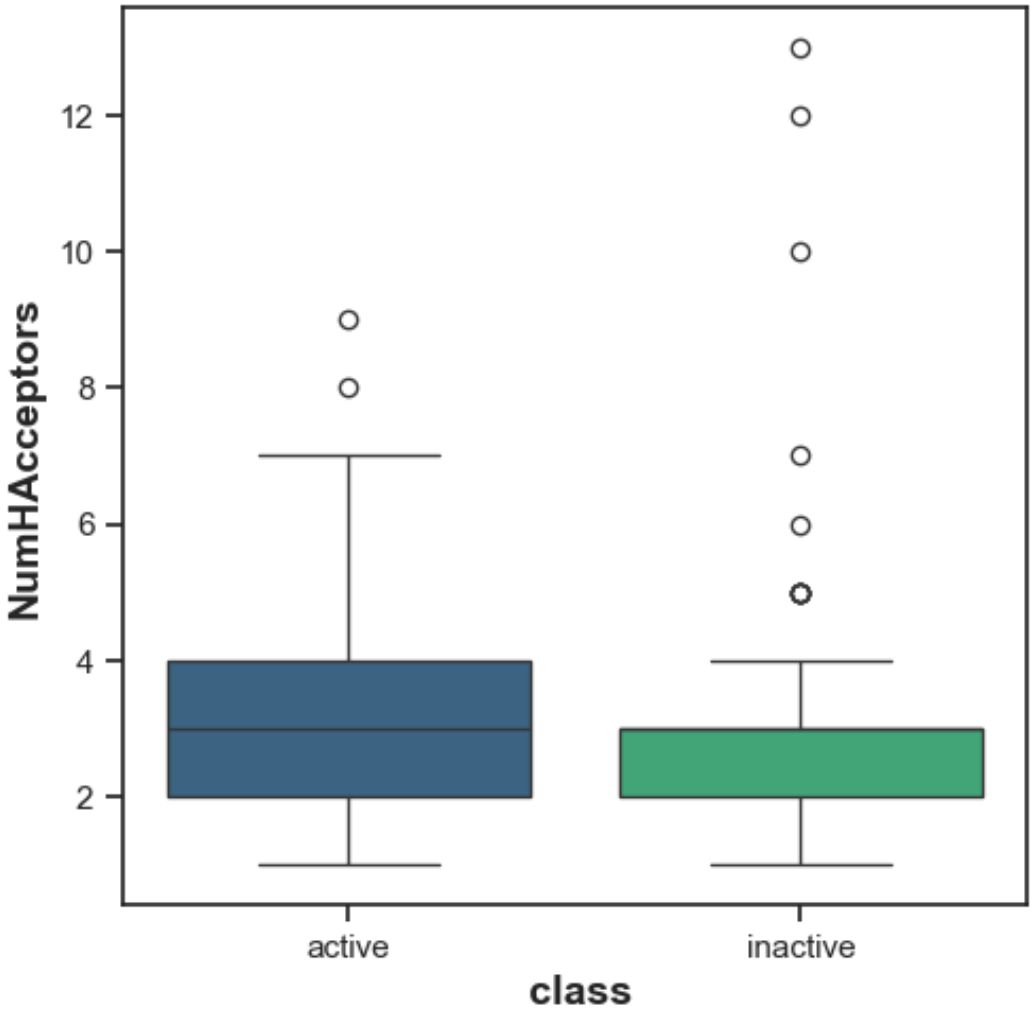
```
plt.figure(figsize=(5.5, 5.5))

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[414…

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



Statistical analysis | Mann-Whitney U Test

In [418...
```
mannwhitney('NumHAcceptors')
```

Out[418...

	Descriptor	Statistics	p	alpha	Interpretation
0	NumHAcceptors	67966.0	0.179462	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 MW, LogP exhibited difference between the actives and inactives while the other descriptors (NumHDonors and NumHAcceptors) shows statistically significant same difference between actives and inactives.

In [430…
```
df3 = pd.read_csv(r"C:\Users\manoj\OneDrive\Desktop\Serotonin\Serotonin_04_bioactivity_data_3class_pIC50.csv")
```

In [432…
```
df3
```

Out[432…

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAccep
0	0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	active	477.605	5.31370	1.0	
1	1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	active	489.616	5.43740	1.0	
2	2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	active	407.504	5.98300	0.0	
3	3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	active	457.599	5.39930	1.0	
4	4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	active	399.578	6.13740	0.0	
...	...	...	...	...	...	...	...	
1004	1365	CHEMBL4851065	N#Cc1ccc2[nH]cc(CCCN3CCN(c4ccc(-n5ccc6occc6c5=...	active	477.568	4.69148	1.0	
1005	1366	CHEMBL4864918	N#Cc1ccc2[nH]cc(CCCCN3CCN(c4ccc(-n5ccc6occc6c5...	active	491.595	5.08158	1.0	
1006	1367	CHEMBL5175117	O=C(/C=C/c1ccc(O)cc1)NCCCCN(CCCNC(=O)/C=C/c1cc...	active	583.685	4.47470	5.0	
1007	1368	CHEMBL5220371	O=C(c1ccc(F)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F)...	active	445.941	5.17410	2.0	
1008	1369	CHEMBL5220872	O=C(c1ccc(Cl)c(Cl)c1)N1CCC(CNCCCc2c[nH]c3ccc(F...	active	462.396	5.68840	2.0	

1009 rows × 9 columns

In [435…
```
df3.head()
```

Out[435...

	Unnamed: 0	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors
0	0	CHEMBL18501	O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3c...	active	477.605	5.3137	1.0	3.0
1	1	CHEMBL19226	O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3c...	active	489.616	5.4374	1.0	3.0
2	2	CHEMBL162006	Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1	active	407.504	5.9830	0.0	2.0
3	3	CHEMBL19203	O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3...	active	457.599	5.3993	1.0	3.0
4	4	CHEMBL351215	c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1	active	399.578	6.1374	0.0	2.0

In [437...

```
df3.describe()
```

Out[437...

	Unnamed: 0	MW	LogP	NumHDonors	NumHAcceptors	pIC50
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000
mean	703.633300	361.950905	4.245717	0.769078	3.061447	6.820464
std	411.081276	93.716762	1.339763	0.958046	1.366474	1.322238
min	0.000000	163.264000	-0.968500	0.000000	1.000000	2.443697
25%	338.000000	291.438000	3.371500	0.000000	2.000000	6.142668
50%	729.000000	362.344000	4.204300	1.000000	3.000000	6.739929
75%	1059.000000	430.542000	5.081580	1.000000	4.000000	7.769551
max	1369.000000	952.075000	10.314500	12.000000	13.000000	10.522879

In [457...

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

In [459...

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

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

```
O=S1(=O)c2cccc3cccc(c23)N1CCN1CCC(CCc2c[nH]c3ccc(F)cc23)CC1     CHEMBL18501
O=S1(=O)c2ccccc2-c2ccccc2N1CCN1CCC(Cc2c[nH]c3ccc(F)cc23)CC1     CHEMBL19226
Fc1ccc(C(OCC2CCN(Cc3ccccc3)CC2)c2ccc(F)cc2)cc1  CHEMBL162006
O=S1(=O)c2cccc3cccc(c23)N1CCCCN1CC=C(c2c[nH]c3ccccc23)CC1       CHEMBL19203
c1ccc(CCN2CCC(CCOC(c3ccccc3)c3ccccc3)CC2)cc1    CHEMBL351215
```

In [461…

```
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: 1009
```

Preparing the X and Y Data Matrices

In [526…

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

In [527…

```
X
```

Out[527…

	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 [530…

```
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 [532…

```
X.dtypes
```

Out[532…

```
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 [534…

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

Out[534...

	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 [537...

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

In [539...

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

Split dataset

In [545...

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

```
Shape of X: (6946, 881)
Shape of y: (1009,)
```

In [549…
```python
X = X.iloc[:y.shape[0], :]  # Trim X to match y
```

In [551…
```python
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (1009, 881)
Shape of y: (1009,)
```

In [554…
```python
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 [556…
```python
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 [558…
```python
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [572…
```python
lr = LogisticRegression(max_iter=600)
```

In [574…
```python
lr.fit(X_train, y_train)
```

Out[574…
```
LogisticRegression
LogisticRegression(max_iter=600)
```

In [576…
```python
y_pred_lr = lr.predict(X_test)
y_pred_lr
```

Out[576…

```
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, 0, 1, 1, 1, 1, 1, 0, 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, 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, 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, 0, 1, 1, 1, 1, 0, 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, 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, 1, 1, 1, 1, 1, 1, 1, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0], dtype=int64)
```

In [586…

```
print(classification_report(y_test, y_pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.15      0.05      0.07        41
           1       0.87      0.96      0.91       262

    accuracy                           0.83       303
   macro avg       0.51      0.50      0.49       303
weighted avg       0.77      0.83      0.80       303
```

In [588…

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

```
Logistic Regression Model Accuracy: 83.50%
```

In [590…

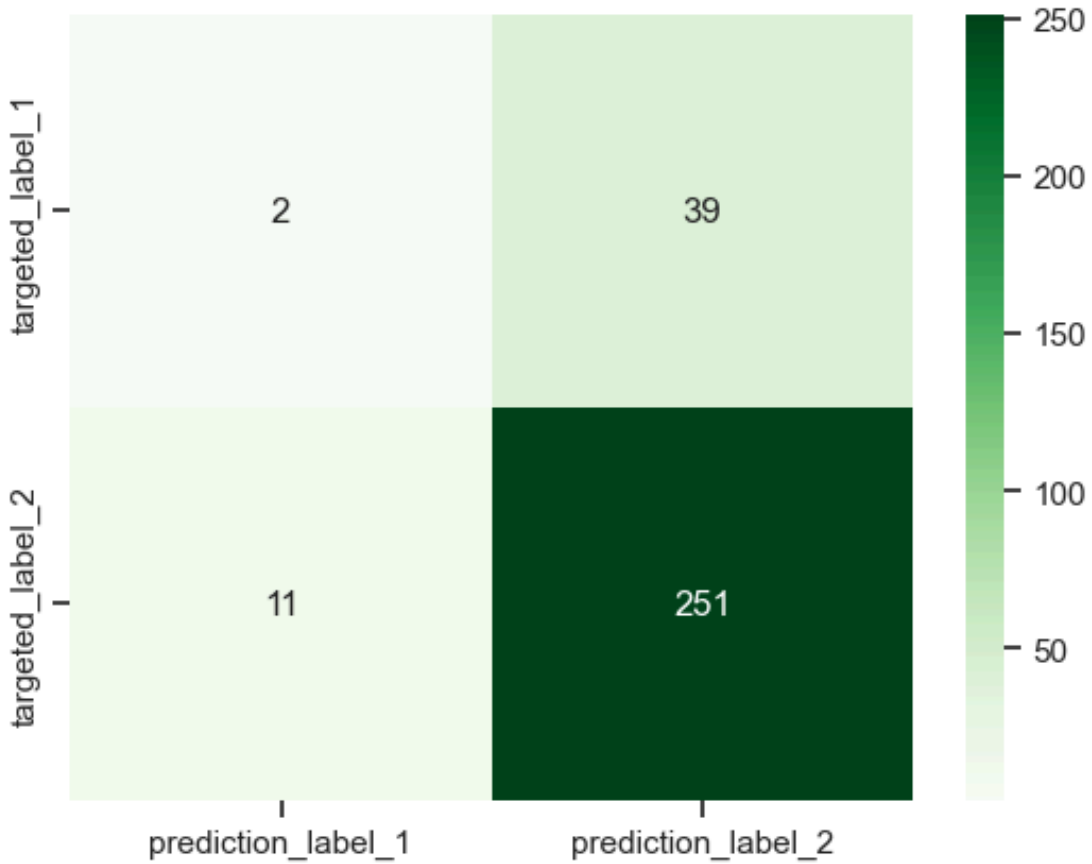
```
y_train_pred = lr.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print("Training Accuracy:", train_accuracy)
```

```
Training Accuracy: 0.8796033994334278
```

In [600…

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

Out[600… [Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]



Logistic Regression Model Accuracy: 83.50%

DecisionTreeClassifier

In [623…

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

Out[623…

DecisionTreeClassifier

DecisionTreeClassifier()

In [626…

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

Out[626…

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

In [629…

```
print(classification_report(y_test,y_pred_DT))
```

```
              precision    recall  f1-score   support

           0       0.21      0.27      0.24        41
           1       0.88      0.84      0.86       262

    accuracy                           0.77       303
   macro avg       0.55      0.56      0.55       303
weighted avg       0.79      0.77      0.78       303
```

In [631…

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

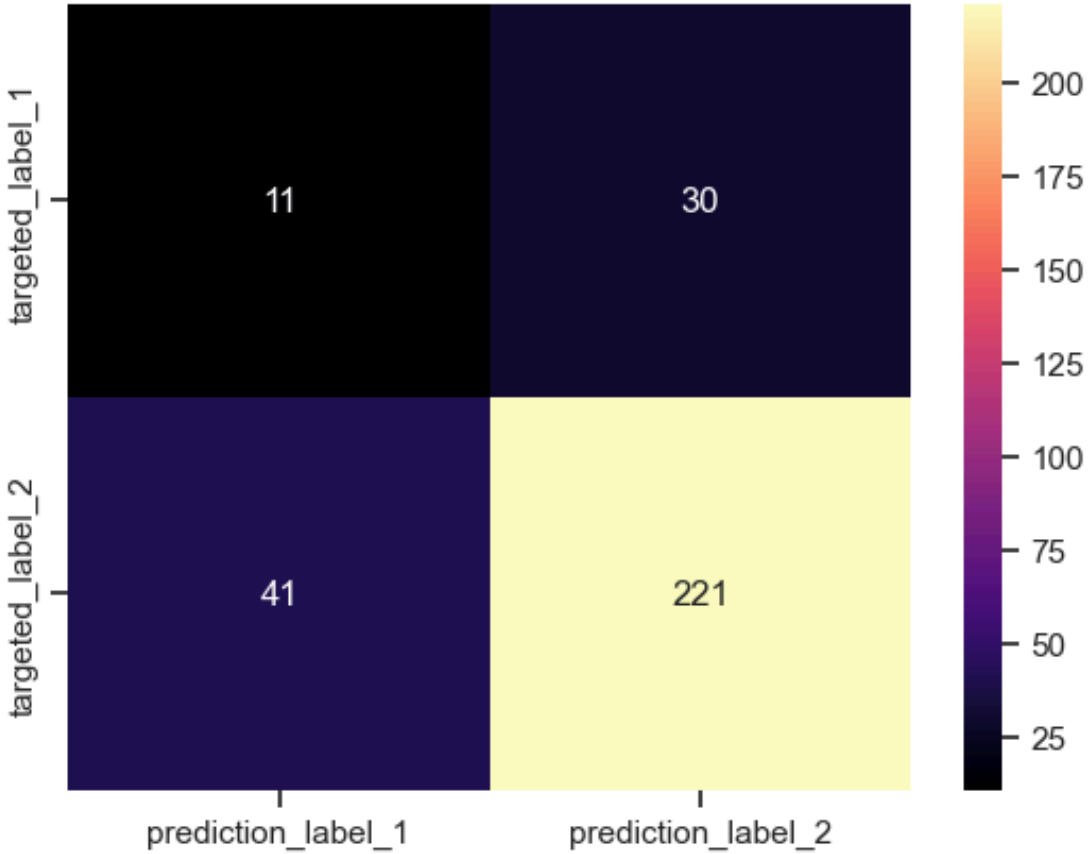
```
DecisionTreeClassifier Model Accuracy: 76.57%
```

In [657…

```
cm = confusion_matrix(y_test, y_pred_DT)
ax = sns.heatmap(cm, annot=True, fmt='d', cmap="magma")
```

```
ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

Out[657… `[Text(0, 0.5, 'targeted_label_1'), Text(0, 1.5, 'targeted_label_2')]`



DecisionTreeClassifier Model Accuracy: 76.57%

RandomForestClassifier

In [695…

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

Out[695…

RandomForestClassifier

RandomForestClassifier()

In [699…

```
y_pred_rnf = rnf.predict(X_test)
y_pred_rnf
```

Out[699…

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

In [702…

```
print(classification_report(y_test,y_pred_rnf))
```

```
              precision    recall  f1-score   support

           0       0.23      0.17      0.19        41
           1       0.88      0.91      0.89       262

    accuracy                           0.81       303
   macro avg       0.55      0.54      0.54       303
weighted avg       0.79      0.81      0.80       303
```

In [723…

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

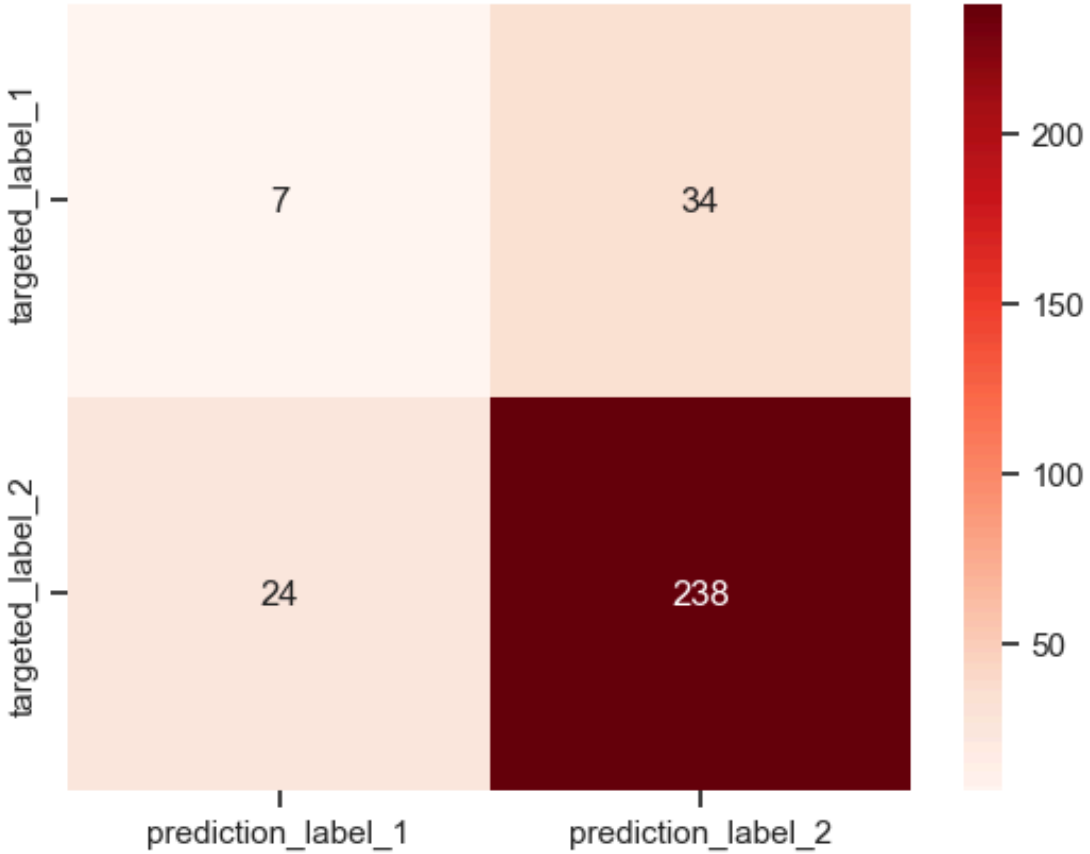
```
RandomForestClassifier Model Accuracy: 80.86%
```

In [719…

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



RandomForestClassifier Model Accuracy: 80.86%

K-Nearest Neighbors (KNN)

In [732…

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
```

Out[732…

KNeighborsClassifier

KNeighborsClassifier()

In [734…

```
y_pred_knn = knn.predict(X_test)
y_pred_knn
```

Out[734…

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

In [736…

```
print(classification_report(y_test,y_pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.25      0.12      0.16        41
           1       0.87      0.94      0.91       262

    accuracy                           0.83       303
   macro avg       0.56      0.53      0.54       303
weighted avg       0.79      0.83      0.81       303
```

In [740…

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

```
KNeighborsClassifier Model Accuracy: 83.17%
```

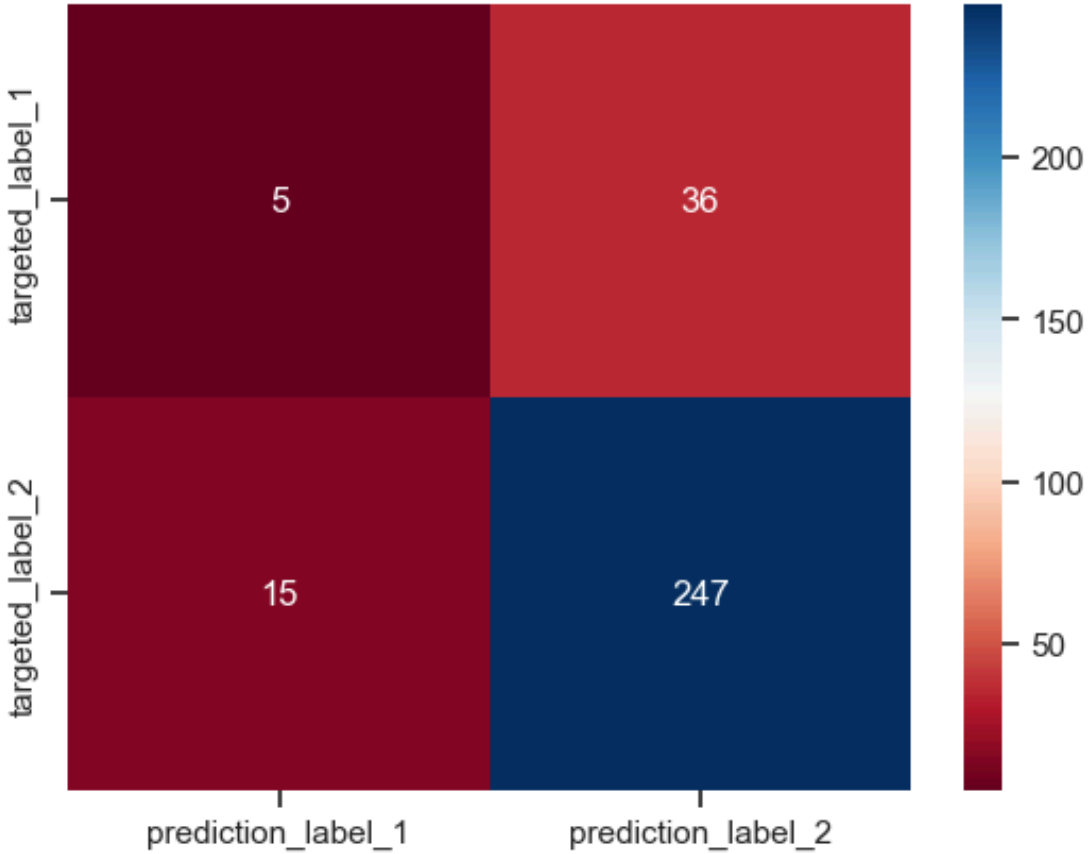
In [751…

```
cm = confusion_matrix(y_test, y_pred_knn)
ax = sns.heatmap(cm, annot=True, fmt='d', cmap="RdBu")
```

```
ax.xaxis.set_ticklabels(['prediction_label_1', 'prediction_label_2' ])
ax.yaxis.set_ticklabels(['targeted_label_1', 'targeted_label_2' ])
```

Out[751…

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



KNeighborsClassifier Model Accuracy: 83.17%

In [755…

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

In [757…

```
df3_X
```

Out[757...

	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 [759...

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

Out[759...

	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 [779...

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

Out[779…

```
0       8.853872
1       7.455932
2       6.118615
3       9.096910
4       6.991400
          ...
1004    8.376751
1005    9.275724
1006    6.130768
1007    8.086186
1008    7.769551
Name: pIC50, Length: 1009, dtype: float64
```

Combining X and Y variable

In [782…

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

Out[782...

	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 [784...
```
from sklearn.model_selection import train_test_split
import lazypredict
from lazypredict.Supervised import LazyRegressor
```

In [785...
```
X.shape
```

Out[785... (1009, 131)

In [786...
```
# Remove low variance features
from sklearn.feature_selection import VarianceThreshold
selection = VarianceThreshold(threshold=(.8 * (1 - .8)))
X = selection.fit_transform(X)
X.shape
```

Out[786… (1009, 131)

In [787…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [799…
```
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)
```

```
98%|█████████▊| 41/42 [00:37<00:00,  2.76it/s]
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001666 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 393
[LightGBM] [Info] Number of data points in the train set: 807, number of used features: 131
[LightGBM] [Info] Start training from score 6.800314
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

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

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best 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 [00:37<00:00,  1.12it/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
[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)
```

```
 98%|█████████▊| 41/42 [00:35<00:00,  2.85it/s]
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001641 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 393
[LightGBM] [Info] Number of data points in the train set: 807, number of used features: 131
[LightGBM] [Info] Start training from score 6.800314
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

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

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best 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 [00:36<00:00,  1.16it/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
```

In [802…

```
predictions_train
```

Out[802...

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
DecisionTreeRegressor	0.45	0.54	0.89	0.06
ExtraTreeRegressor	0.45	0.54	0.89	0.07
ExtraTreesRegressor	0.45	0.54	0.89	2.82
GaussianProcessRegressor	0.45	0.54	0.89	0.40
XGBRegressor	0.45	0.54	0.89	0.39
RandomForestRegressor	0.39	0.49	0.93	2.43
BaggingRegressor	0.36	0.47	0.96	0.27
HistGradientBoostingRegressor	0.29	0.40	1.01	1.96
LGBMRegressor	0.25	0.37	1.04	0.58
GradientBoostingRegressor	0.15	0.29	1.10	0.95
KNeighborsRegressor	0.12	0.26	1.12	0.12
MLPRegressor	0.09	0.23	1.14	2.93
SVR	0.06	0.21	1.16	0.43
NuSVR	0.05	0.20	1.17	0.33
LinearRegression	0.04	0.19	1.18	0.20
TransformedTargetRegressor	0.04	0.19	1.18	0.16
Ridge	0.03	0.19	1.18	0.06
RidgeCV	0.02	0.18	1.19	0.23
HuberRegressor	0.01	0.17	1.19	0.29
LinearSVR	-0.02	0.15	1.21	0.30
PoissonRegressor	-0.05	0.12	1.23	0.09

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
SGDRegressor	-0.08	0.10	1.24	0.06
OrthogonalMatchingPursuit	-0.10	0.08	1.25	0.05
AdaBoostRegressor	-0.11	0.07	1.26	0.28
TweedieRegressor	-0.11	0.07	1.26	0.06
GammaRegressor	-0.11	0.07	1.26	0.26
LassoLarsIC	-0.13	0.06	1.27	0.30
BayesianRidge	-0.14	0.05	1.28	0.14
OrthogonalMatchingPursuitCV	-0.18	0.01	1.30	0.10
LarsCV	-0.18	0.01	1.30	0.65
DummyRegressor	-0.19	0.00	1.31	0.04
ElasticNet	-0.19	0.00	1.31	0.04
ElasticNetCV	-0.19	0.00	1.31	10.34
LassoLarsCV	-0.19	0.00	1.31	0.30
LassoLars	-0.19	0.00	1.31	0.04
Lasso	-0.19	0.00	1.31	0.04
LassoCV	-0.19	0.00	1.31	7.85
QuantileRegressor	-0.20	-0.00	1.31	0.37
PassiveAggressiveRegressor	-0.51	-0.26	1.47	0.05
KernelRidge	-32.21	-26.81	6.90	0.10
Lars	-900536249470.28	-754171176665.39	1136412.52	0.12
RANSACRegressor	-1016575687993907184926720.00	-851350607190927280701440.00	381817006489.17	1.27

In [805…

```
predictions_test
```

Out[805...

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
NuSVR	-1.76	0.04	1.34	0.21
LassoLarsIC	-1.77	0.04	1.34	0.23
PoissonRegressor	-1.79	0.03	1.35	0.09
TweedieRegressor	-1.79	0.03	1.35	0.06
GammaRegressor	-1.79	0.03	1.35	0.06
BayesianRidge	-1.81	0.02	1.35	0.31
SVR	-1.82	0.02	1.36	0.28
OrthogonalMatchingPursuit	-1.83	0.01	1.36	0.05
GradientBoostingRegressor	-1.84	0.01	1.36	0.93
LassoLarsCV	-1.89	-0.01	1.37	0.29
LassoCV	-1.89	-0.01	1.37	8.46
Lasso	-1.89	-0.01	1.37	0.05
LassoLars	-1.89	-0.01	1.37	0.05
ElasticNetCV	-1.89	-0.01	1.37	8.98
ElasticNet	-1.89	-0.01	1.37	0.04
DummyRegressor	-1.89	-0.01	1.37	0.04
LarsCV	-1.89	-0.01	1.37	0.58
HuberRegressor	-1.90	-0.01	1.38	0.28
OrthogonalMatchingPursuitCV	-1.91	-0.01	1.38	0.10
QuantileRegressor	-1.92	-0.02	1.38	0.35
RidgeCV	-1.94	-0.02	1.39	0.23

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
AdaBoostRegressor	-2.00	-0.04	1.40	0.33
LinearSVR	-2.00	-0.05	1.40	0.28
Ridge	-2.02	-0.05	1.40	0.04
LinearRegression	-2.03	-0.06	1.41	0.32
TransformedTargetRegressor	-2.03	-0.06	1.41	0.21
SGDRegressor	-2.05	-0.06	1.41	0.06
HistGradientBoostingRegressor	-2.06	-0.06	1.41	1.87
LGBMRegressor	-2.06	-0.07	1.41	0.52
MLPRegressor	-2.13	-0.09	1.43	2.98
RandomForestRegressor	-2.19	-0.11	1.44	2.46
BaggingRegressor	-2.29	-0.15	1.47	0.32
KNeighborsRegressor	-2.34	-0.16	1.48	0.06
XGBRegressor	-2.78	-0.32	1.57	0.39
ExtraTreesRegressor	-3.05	-0.41	1.63	2.81
ExtraTreeRegressor	-3.08	-0.42	1.63	0.07
PassiveAggressiveRegressor	-3.21	-0.47	1.66	0.05
DecisionTreeRegressor	-3.21	-0.47	1.66	0.07
GaussianProcessRegressor	-23.90	-7.67	4.03	0.29
KernelRidge	-74.56	-25.32	7.02	0.08
Lars	-2040877408643.06	-710753326392.45	1154436.77	0.13
RANSACRegressor	-2848372154123122942935040.00	-9919704019334258596249600.00	431280731910.59	1.22

Bar plot of R-squared values

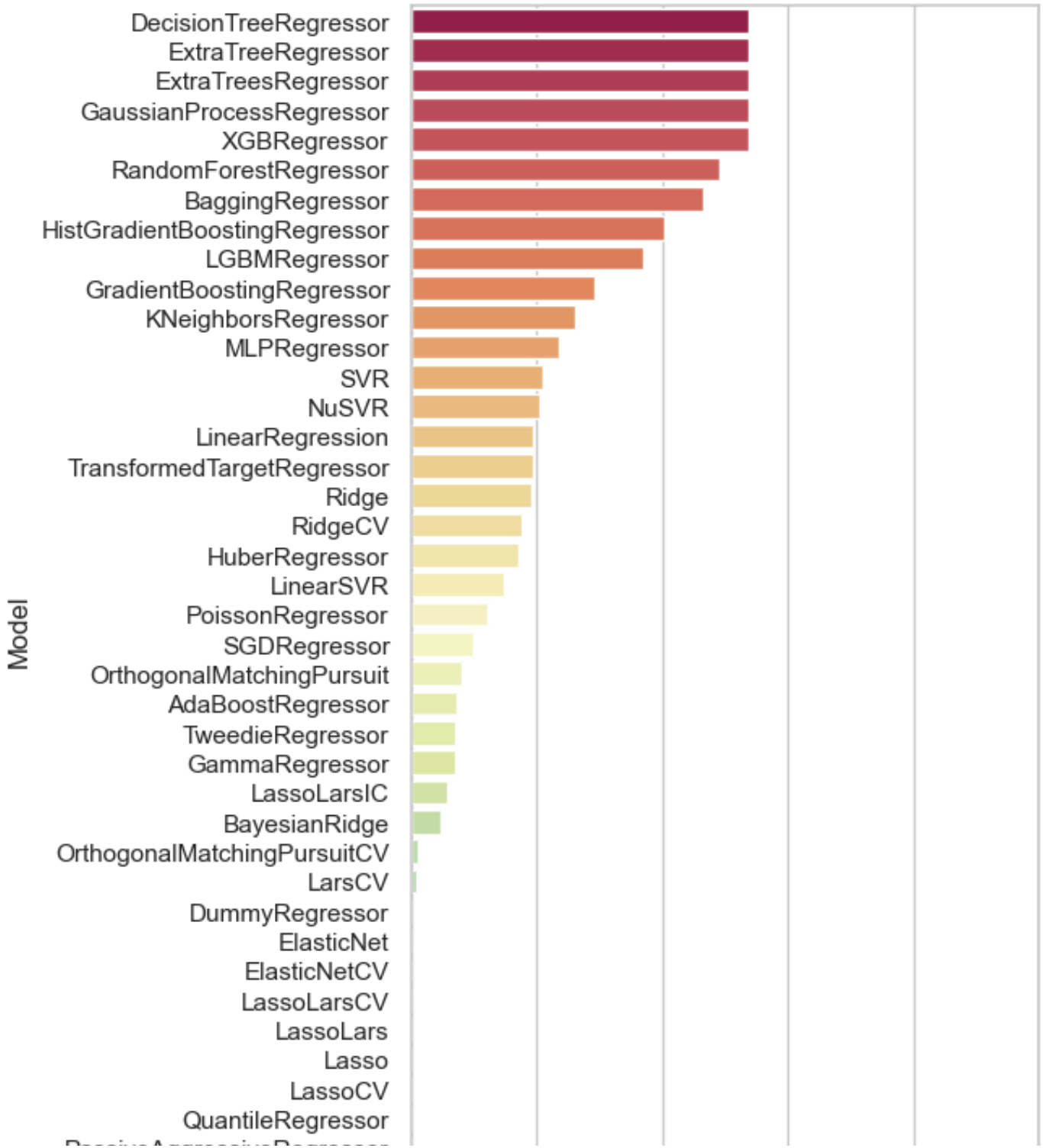
In [811…

```
#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[811…

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


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

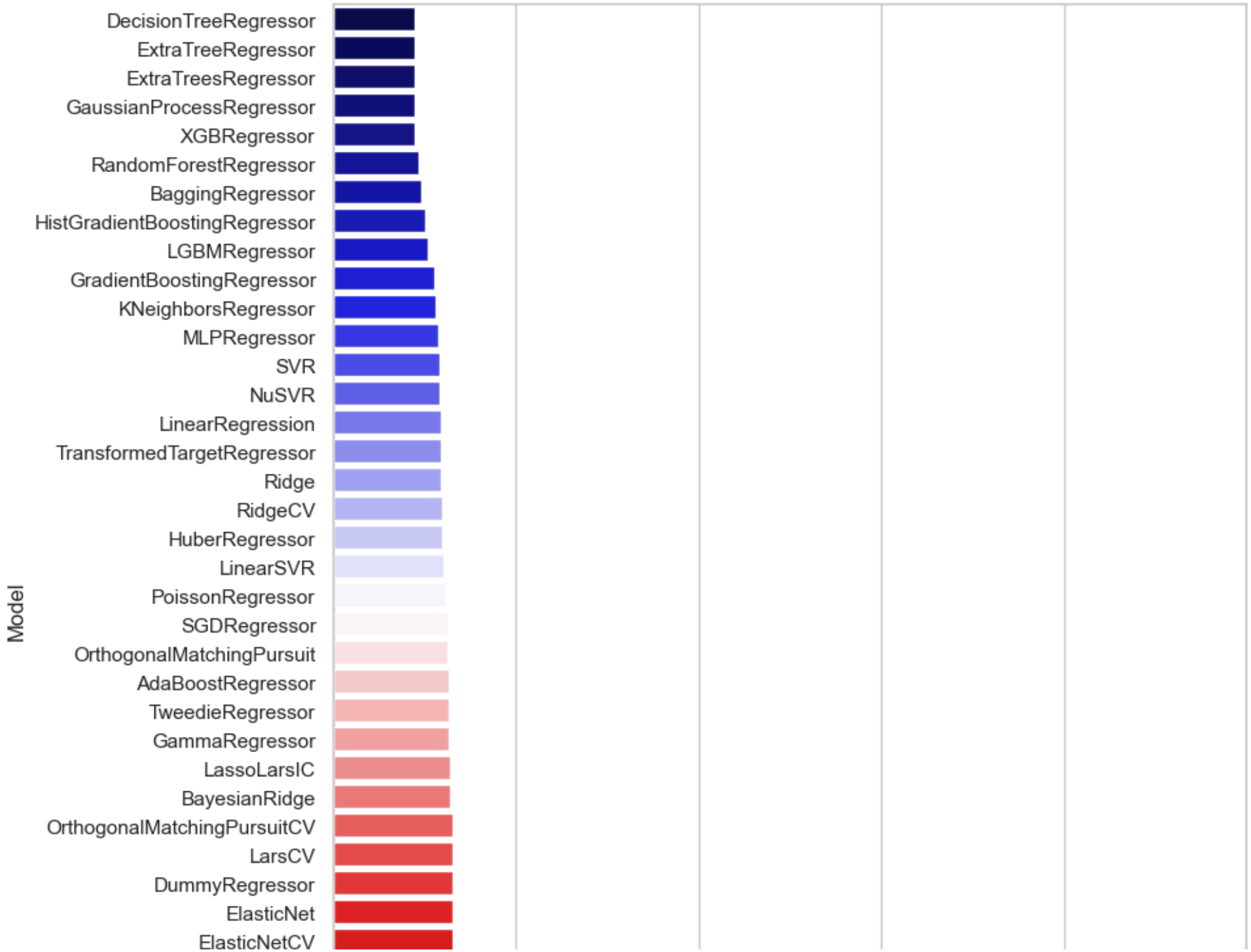
Bar plot of RMSE values

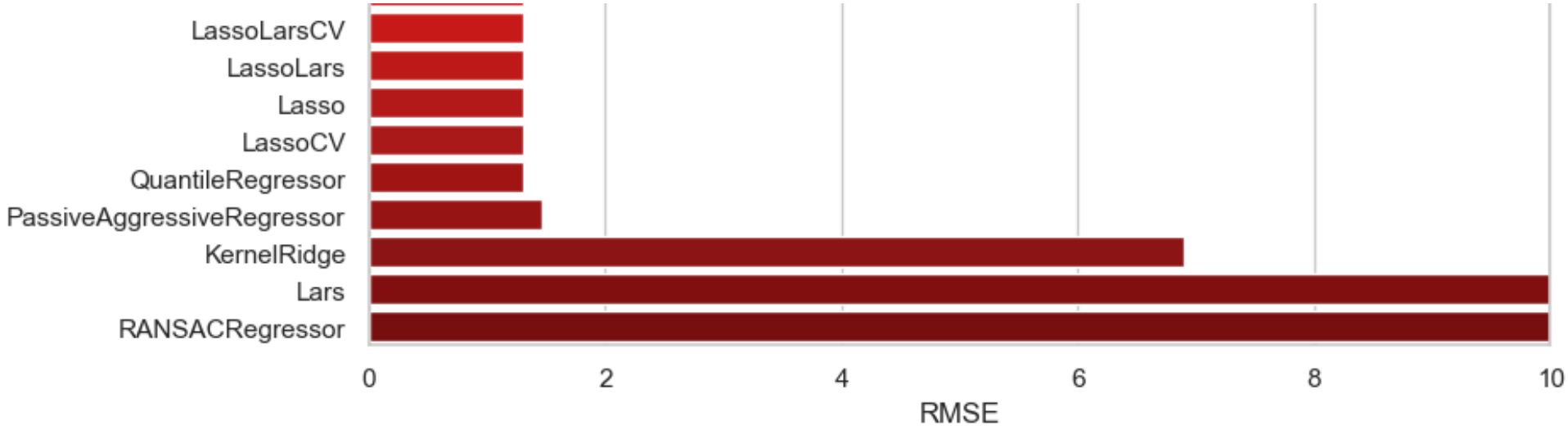
In [816…

```
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[816…

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


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



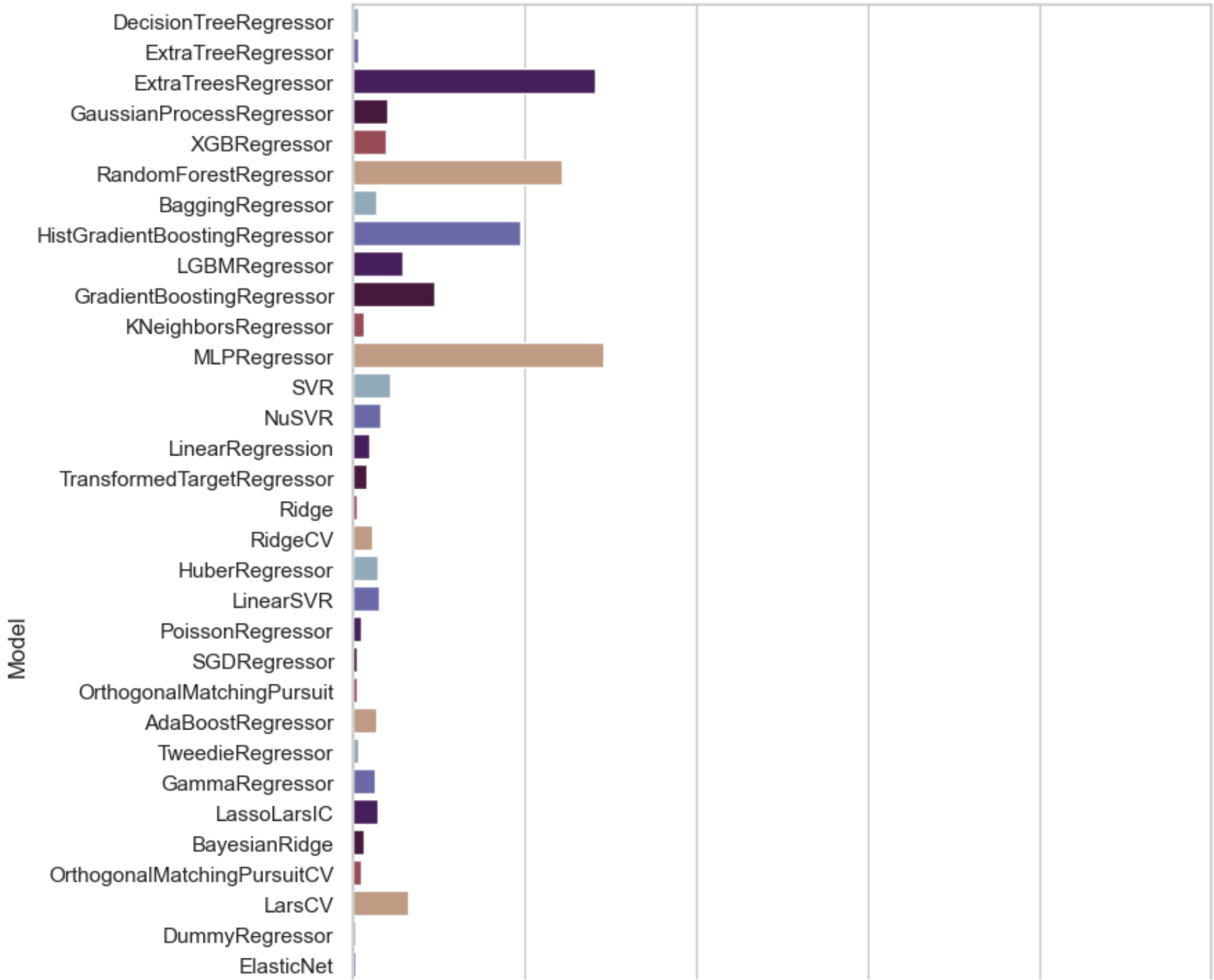
Bar plot of calculation time

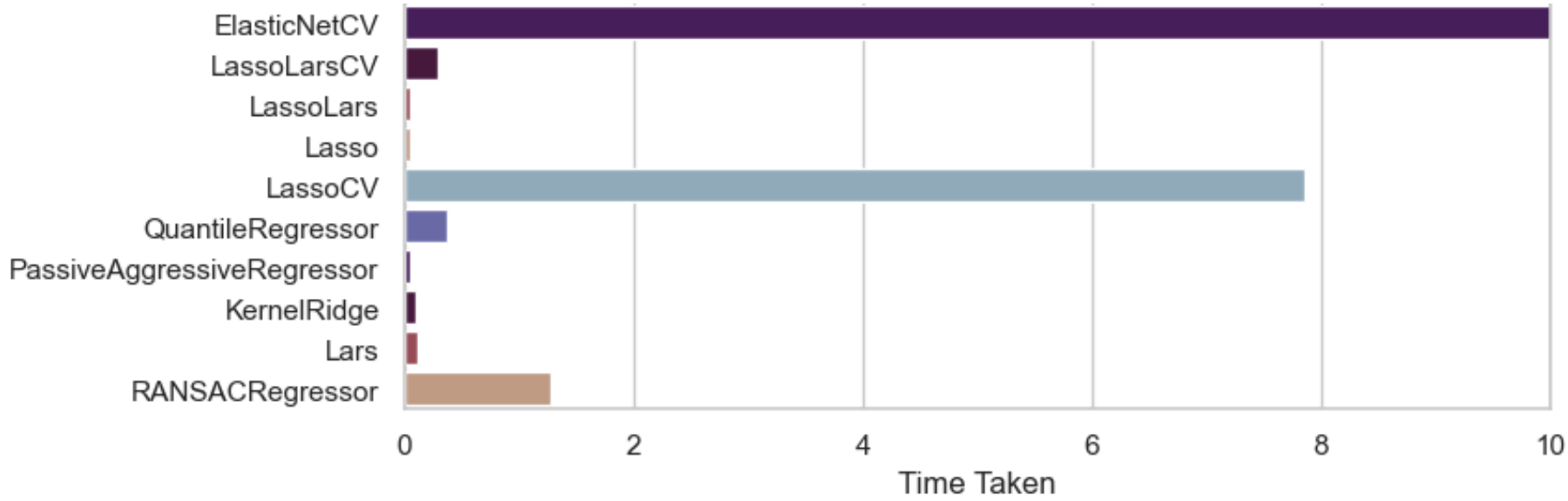
In [821…

```
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
HistGradientBoostingRegressor
LGBMRegressor
GradientBoostingRegressor
KNeighborsRegressor
MLPRegressor
SVR
NuSVR
LinearRegression
TransformedTargetRegressor
Ridge
RidgeCV
HuberRegressor
LinearSVR
PoissonRegressor
SGDRegressor
OrthogonalMatchingPursuit
AdaBoostRegressor
TweedieRegressor
GammaRegressor
LassoLarsIC
BayesianRidge
OrthogonalMatchingPursuitCV
LarsCV
DummyRegressor
ElasticNet
Model


ElasticNetCV
LassoLarsCV
LassoLars
Lasso
LassoCV
QuantileRegressor
PassiveAggressiveRegressor
KernelRidge
Lars
RANSACRegressor
0
2
4
6
8
10
Time Taken

In [ ]: