

MLProject:

Covid data analysis

Team:

Anushk Naval (18046)

Subhajit Pramanik (18398)

Subhojit Pal (18400)

Submitted to:

Dr. Parthiban Srinivasan

Abstract:

For this project, we were asked to use and experiment with the drug dataset available at ChEMBL site, and to explore how to use different machine learning algorithm in a project and find pattern in data.

We were expected gain insights and experience in data cleaning and processing using common data-mining and machine learning library.

We were expected to submit a report about the dataset and different algorithms that were used.

During the project the use of neural network or deep learning techniques were restricted.

This is the report on the data and the techniques used in the project with interpretations.

Keywords: Machine learning, Data cleaning, Regression Modeling, Bioinformatics.

Opening and running the file:

*Use **Google Colab** to run the file.*

Program collects data directly from ChEMBL Database at the runtime.

Introduction:

In this project our team worked on ChEMBL drug data, we selected the compounds that target SARS coronavirus having type – single protein from the database.

The bioactivity of the data we are calculating potency of the drug that's the amount required to produce an effect of given intensity lower the value the more the potency.

We calculated the descriptors and fingerprint for the compounds imported and then ran different regression models to compute the bioactivity of the compounds

Methods and Data:

The dataset we used could be found: <https://www.ebi.ac.uk/chembl/>

The database contains the details of compounds with their drug-like properties, bringing together chemical, bioactivity and genomic data to aid the translation of genomic information into effective drugs.

We imported the data of our interest target (SARS coronavirus).

The imported dataset has information of molecules like chembl id, canonical smiles, bio-activity values for our project we took its type to be "IC50" (make data more uniform).

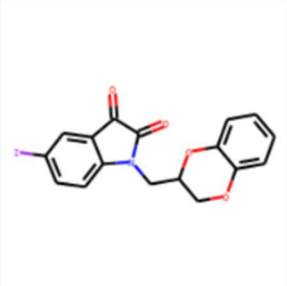
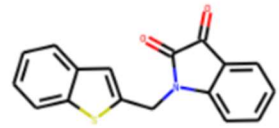
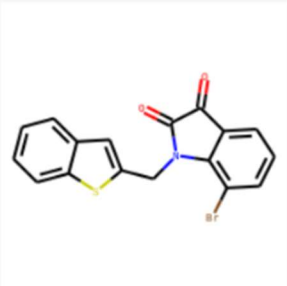
Data cleaning and processing:

From the data imported any missing values for our project has been removed and only the features required like the molecule chembl id, the smiles and their bioactivity values were taken.

The data is then categorizing into three class – active (< 1000 nm), intermediate (1001nm-10000nm) and inactive (above 10000nm) based on their bioactivity values.

Then to uniformly distribute the activity values the standard IC50 values were converted to pIC50 values.

The intermediate bioactivity class was removed to easily compare between inactive and active compounds.

molecule_chembl_id	canonical_smiles	bioactivity_class	Molecule	pIC50
CHEMBL185698	<chem>O=C1C(=O)N(CC2COc3ccccc3O2)c2ccc(I)cc21</chem>	inactive		4.869666
CHEMBL426082	<chem>O=C1C(=O)N(Cc2cc3ccccc3s2)c2ccccc21</chem>	inactive		4.882397
CHEMBL365134	<chem>O=C1C(=O)N(Cc2cc3ccccc3s2)c2c(Br)cccc21</chem>	active		6.008774

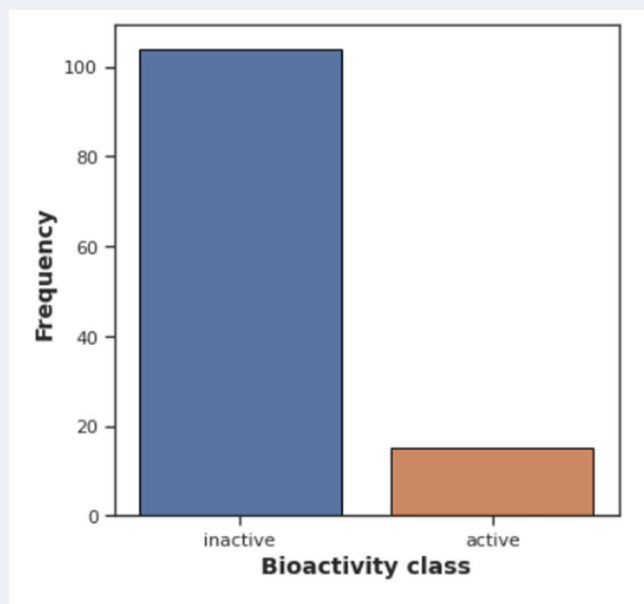
The following data frame is obtained after cleaning the data.

Molecular descriptors:

The molecule as a real object contains all the chemical information, by theoretical pathways the information encoded in the molecular structure into one or more numbers is molecular descriptors, used to establish relationship between structures and properties, biological activities and other experimental properties.

Molecular fingerprints:

Fingerprint representations of molecular structure and properties are a particularly complex forms of descriptors, encoded as binary bits “patterns” characteristic of a given molecule.



The graph shows the frequency distribution of the two bioactivity classes of our data.

In the program we calculated Mordred descriptors and PaDEL fingerprints for the molecules.

In both the processes the smiles of the molecules are used to generate the data further.

As mordred gives us about 1500 molecular descriptor of the compound we used Mann-Whitney U test which test the equality of means in two independent samples.

The test is used here to see difference between the two bioactivity classes, this will test the statistical significance of difference.

The code for that is taken and modified from:

<https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/>

By the test where the distribution was same it fails to reject the descriptor and where the distribution is different it rejected it giving us Descriptors of some significance to the prediction.

Similarly PaDEL fingerprints were calculated, we download the PaDEL-Descriptor software from: <https://github.com/dataprofessor/bioinformatics/raw/master/padel.zip>
<https://github.com/dataprofessor/bioinformatics/raw/master/padel.sh>

Then it took our data and calculated PaDEL fingerprints.

And finally each data is concatenated with pIC50 values.

The Mordred data:

AATS1se	AATS3d	AATS3p	AATS4i	AATS4p	AATS7are	AATS7dv	AATS7se	AATS8are	AATS8d	AATS8dv	AATS8pe	AATS8se	AATSC0are	AATSC0d	AATSC0dv	AATSC0p	AATSC0pe	AATSC1
7.900097	3.511905	1.672098	156.260194	1.569879	6.284945	6.081212	7.867779	6.216273	3.272727	5.710101	6.398634	7.788824	0.156451	0.656327	4.395411	0.701846	0.139730	-0.001
7.688643	3.555556	1.755245	152.508936	1.591362	6.073814	5.976744	7.628938	6.139692	3.128205	5.076923	6.271538	7.661100	0.103094	0.631836	3.756944	0.307233	0.096801	0.000
7.737836	3.708333	1.924280	151.034980	1.837967	6.164233	6.164513	7.746814	6.208923	3.282051	5.270655	6.370923	7.749395	0.102630	0.671875	3.792604	0.386123	0.099962	0.000
7.703236	3.652778	2.059236	150.907904	1.855653	6.075419	6.076486	7.663240	6.140959	3.307692	5.296676	6.332046	7.688299	0.102925	0.671875	3.806628	0.788345	0.094537	0.000
7.757921	3.694444	1.874553	151.937906	1.717295	6.146191	6.229113	7.746049	6.256000	3.230769	5.390313	6.451231	7.844130	0.104269	0.671875	3.732253	0.317469	0.106607	0.000
...
7.754465	3.869159	1.636771	154.284276	1.408736	6.280588	3.952941	7.868576	6.363788	2.363636	3.954545	6.399476	7.964010	0.153867	0.805841	4.678204	0.237588	0.141533	0.000
7.574200	3.413793	1.534285	156.147312	1.310601	6.198169	1.408451	7.803633	6.097627	2.169492	0.440678	6.144542	7.752329	0.109184	0.793651	3.990930	0.241771	0.104283	0.001
7.676596	4.189189	1.812421	149.607933	1.570025	6.206818	2.727273	7.788700	6.089697	2.242424	1.060606	6.153606	7.716690	0.123306	0.754821	4.231405	0.237586	0.114405	0.000
7.647176	3.882353	1.675820	151.280664	1.502956	6.095600	2.120000	7.692205	5.965128	2.205128	0.692308	6.030487	7.604057	0.120539	0.747755	4.199184	0.240875	0.112835	0.000
7.444589	3.137931	1.525722	156.457371	1.280467	5.958082	0.958904	7.584108	5.708305	2.000000	0.406780	5.772568	7.378679	0.081114	0.755003	3.504597	0.243440	0.080562	0.000

The PaDEL data:

PubchemFP0	PubchemFP1	PubchemFP2	PubchemFP3	PubchemFP4	PubchemFP5	PubchemFP6	PubchemFP7	PubchemFP8	PubchemFP9	PubchemFP10	PubchemFP11	PubchemFP12	PubchemFP13	PubchemFP14
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1
...
1	1	1	0	0	0	0	0	0	1	1	1	1	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	0	0
1	1	0	0	0	0	0	0	0	1	1	1	1	0	0
1	1	0	0	0	0	0	0	0	1	1	1	1	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	0	0

After the generation of datasets both sets are processed and split into 80:20 ration that is 80% data for training and 20% data for test.

This is done using `train_test_split` from sklearn library.

Then the models are trained using the train data and are tested for their r^2 values.

r^2 value –

The values range from 0 to 1 commonly stated as percentages from 0% to 100%, It's the statistical measure of how close the data are to the fitted regression lines. Also known as coefficient of determination.

- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

In our program we computed r^2 between predicted and actual values of bioactivity to get the performance of model, more techniques like mean square error, mean absolute error, variance score is also computed.

Model:

In the project we applied 6 different regression models using:

- Random forest regressor
- Support vector regressor
- Decision tree regressor
- k-Neighbour regressor
- Adabooster regressor
- Voting regressor

All the models are applied on both the datasets (Mordred and PaDEL data).

➤ Regression using Random Forest:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Yrm_pred)
rms = metrics.mean_squared_error(Ym_test, Yrm_pred)
vs = metrics.explained_variance_score(Ym_test, Yrm_pred)
r2 = metrics.r2_score(Ym_test, Yrm_pred)
print(mae,rms,vs,r2)

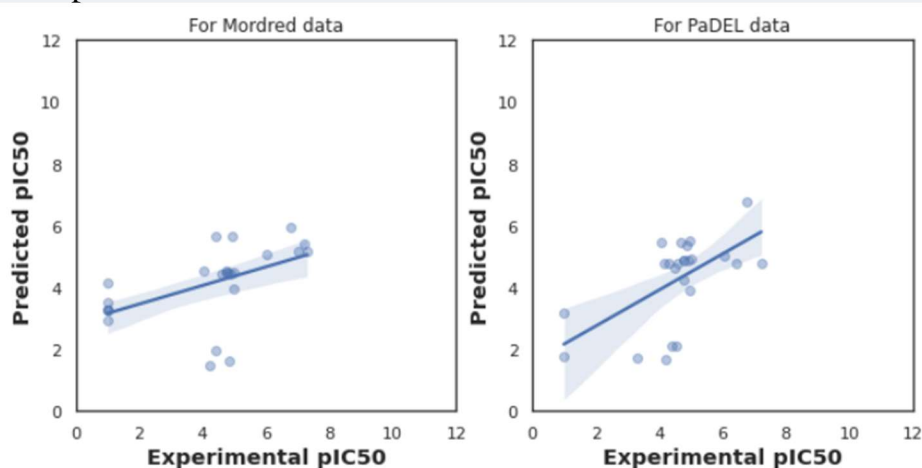
1.4926186742068086 3.1831322895315015 0.25250115671477924 0.2506390079677737
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Yrp_pred)
rms = metrics.mean_squared_error(Yp_test, Yrp_pred)
vs = metrics.explained_variance_score(Yp_test, Yrp_pred)
r2 = metrics.r2_score(Yp_test, Yrp_pred)
print(mae,rms,vs,r2)

0.9733699104300912 1.6735559236712862 0.1887758616798918 0.12942952626690107
```

Regression plots for both the data:



➤ Regression using Support Vector Regressor:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Ysm_pred)
rms = metrics.mean_squared_error(Ym_test, Ysm_pred)
vs = metrics.explained_variance_score(Ym_test, Ysm_pred)
r2 = metrics.r2_score(Ym_test, Ysm_pred)
print(mae,rms,vs,r2)

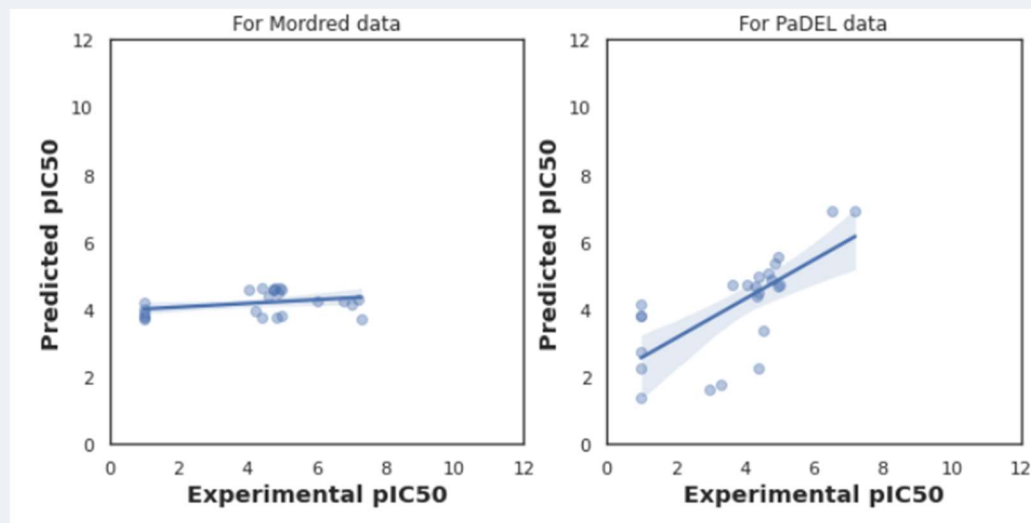
1.5345931965806834 3.897949057681071 0.08252318393149305 0.08235954177547478
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Ysp_pred)
rms = metrics.mean_squared_error(Yp_test, Ysp_pred)
vs = metrics.explained_variance_score(Yp_test, Ysp_pred)
r2 = metrics.r2_score(Yp_test, Ysp_pred)
print(mae,rms,vs,r2)

1.0041413585850354 1.841571803719205 0.4724404829024974 0.4188505735106315
```

Regression plots for both the data:



➤ Regression using Decision Tree Regressor:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Ydm_pred)
rms = metrics.mean_squared_error(Ym_test, Ydm_pred)
vs = metrics.explained_variance_score(Ym_test, Ydm_pred)
r2 = metrics.r2_score(Ym_test, Ydm_pred)
print(mae,rms,vs,r2)
```

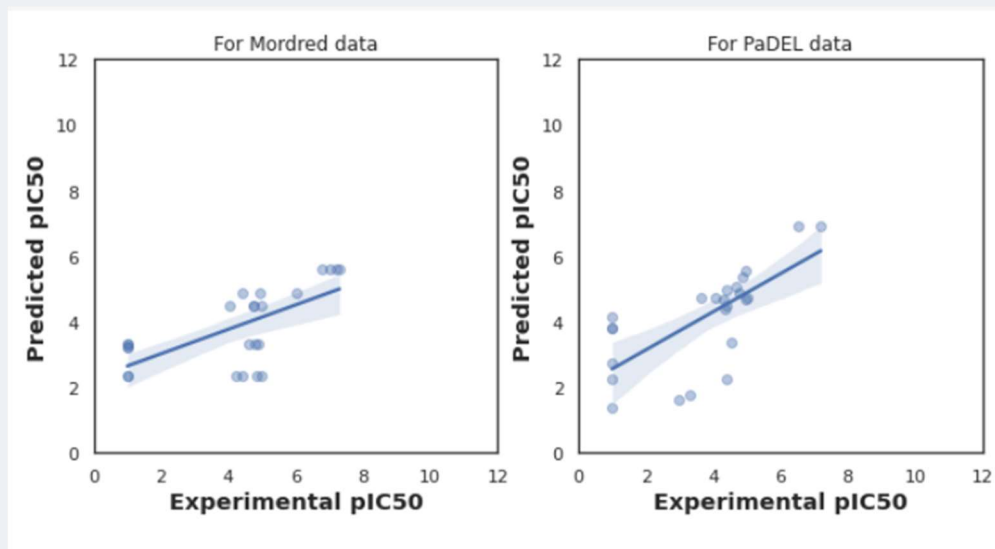
```
1.4230203668033747 2.602573129190407 0.4192703422135968 0.3873120547517105
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Ydp_pred)
rms = metrics.mean_squared_error(Yp_test, Ydp_pred)
vs = metrics.explained_variance_score(Yp_test, Ydp_pred)
r2 = metrics.r2_score(Yp_test, Ydp_pred)
print(mae,rms,vs,r2)
```

```
1.0041413585850354 1.841571803719205 0.4724404829024974 0.4188505735106315
```

Regression plots for both the data:



➤ Regression using KNeighbour Regressor:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Ykm_pred)
rms = metrics.mean_squared_error(Ym_test, Ykm_pred)
vs = metrics.explained_variance_score(Ym_test, Ykm_pred)
r2 = metrics.r2_score(Ym_test, Ykm_pred)
print(mae,rms,vs,r2)

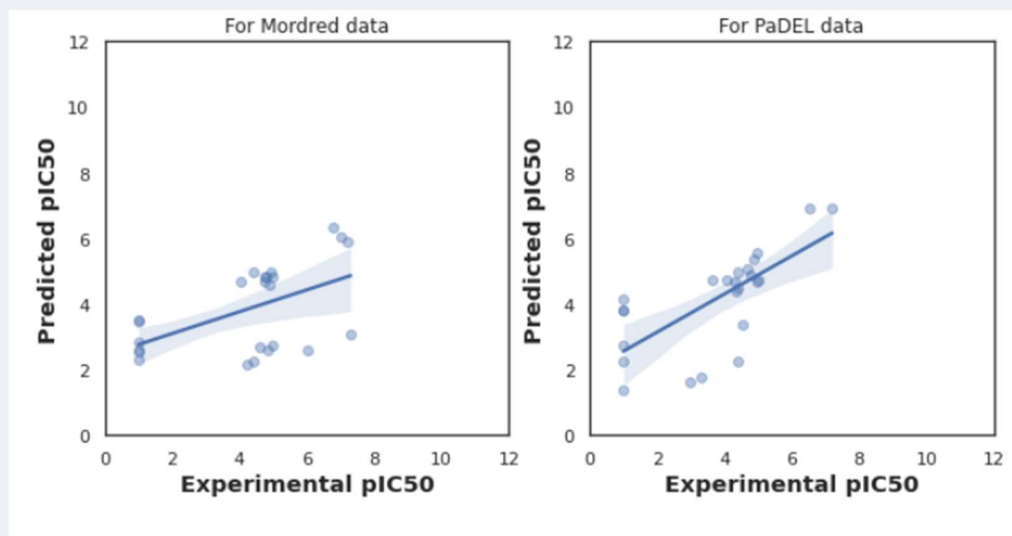
1.4269510329685355 3.2826982602361157 0.26140863540030024 0.22719956285729648
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Ykp_pred)
rms = metrics.mean_squared_error(Yp_test, Ykp_pred)
vs = metrics.explained_variance_score(Yp_test, Ykp_pred)
r2 = metrics.r2_score(Yp_test, Ykp_pred)
print(mae,rms,vs,r2)

1.0041413585850354 1.841571803719205 0.4724404829024974 0.4188505735106315
```

Regression plots for both the data:



➤ Regression using Adabooster Regressor:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Yadm_pred)
rms = metrics.mean_squared_error(Ym_test, Yadm_pred)
vs = metrics.explained_variance_score(Ym_test, Yadm_pred)
r2 = metrics.r2_score(Ym_test, Yadm_pred)
print(mae,rms,vs,r2)

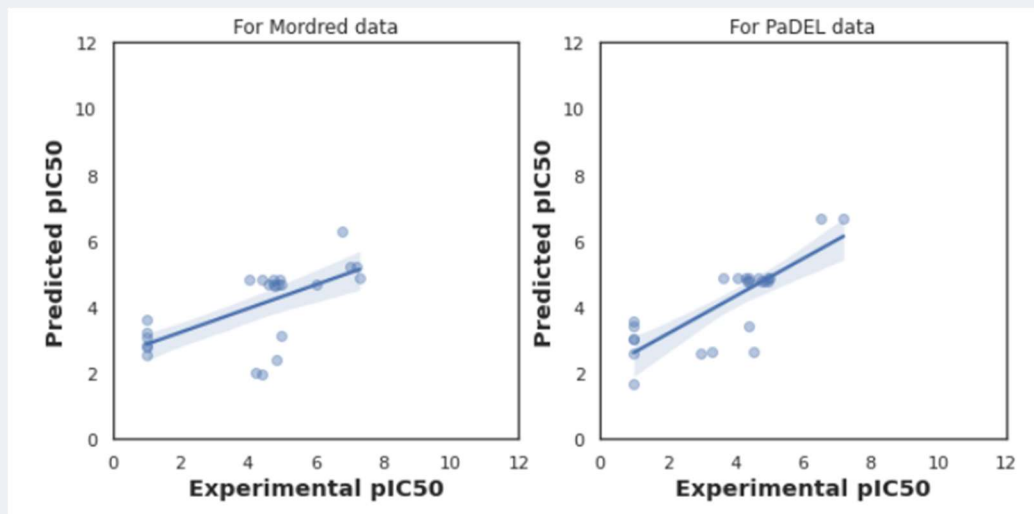
1.3048877562503118 2.5710000432818956 0.40336766080029873 0.39474487149505844
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Yadp_pred)
rms = metrics.mean_squared_error(Yp_test, Yadp_pred)
vs = metrics.explained_variance_score(Yp_test, Yadp_pred)
r2 = metrics.r2_score(Yp_test, Yadp_pred)
print(mae,rms,vs,r2)

0.8495190401525887 1.3234185832105076 0.6412174545854554 0.5823654830700108
```

Regression plots for both the data:



➤ Regression using Voting Regressor:

Statistics for mordred data:

```
mae = metrics.mean_absolute_error(Ym_test, Yem_pred)
rms = metrics.mean_squared_error(Ym_test, Yem_pred)
vs = metrics.explained_variance_score(Ym_test, Yem_pred)
r2 = metrics.r2_score(Ym_test, Yem_pred)
print(mae,rms,vs,r2)

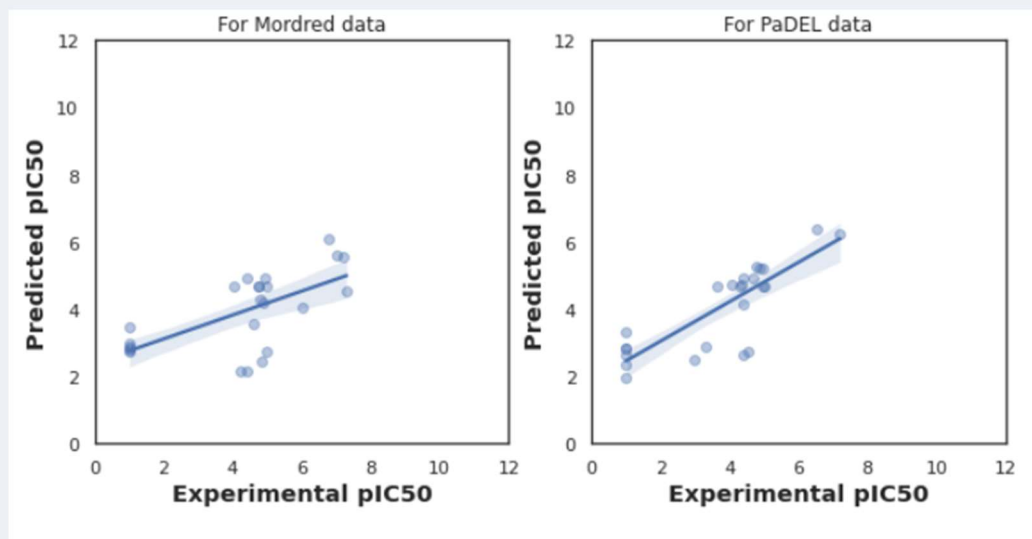
1.377937533273677 2.6104488197227234 0.40862128036662027 0.3854579894055593
```

Statistics for PaDEL data:

```
mae = metrics.mean_absolute_error(Yp_test, Yep_pred)
rms = metrics.mean_squared_error(Yp_test, Yep_pred)
vs = metrics.explained_variance_score(Yp_test, Yep_pred)
r2 = metrics.r2_score(Yp_test, Yep_pred)
print(mae,rms,vs,r2)

0.8616502894032884 1.1698986009310648 0.6649851195328419 0.6308121683831618
```

Regression plots for both the data:



All Model Statistics:

These are the best statistics obtained running the program:

For Mordred Data:

```
The r2 scores of models are:
Random Forest Regression      0.2506390079677737
Support Vector Regression      0.08235954177547478
Decision tree Regression       0.3873120547517105
KNeighbour Regression         0.22719956285729648
Adabooster Regressor          0.39474487149505844
Voting Regressor              0.3854579894055593
```

For PaDEL Data:

```
The r2 scores of models are:
Random Forest Regression      0.4188505735106315
Support Vector Regression      0.279464602815251
Decision tree Regression       0.5136909950397026
KNeighbour Regression         0.6419637917254151
Adabooster Regressor          0.5823654830700108
Voting Regressor              0.6308121683831618
```

Conclusion:

- The PaDel fingerprint calculations were faster and better for predicting the bioactivity values as compared to Mordred descriptors.
- After running and observing different models to fit the data the decision tree regressor was better in most of the iterations for Mordred data.
- After running and observing different models to fit the data the voting regressor was better in most of the iterations for PaDEL data.
- The max r^2 achieved for Mordred data was: 0.395
- The max r^2 achieved for PaDEL data was: 0.642

**** The values of r^2 can change at the time of run.**

References:

- <https://www.ebi.ac.uk/chembl/>
- <https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/>
- <https://github.com/dataprofessor/bioinformatics/raw/master/padel.zip>
- <https://github.com/dataprofessor/bioinformatics/raw/master/padel.sh>
- <https://www.youtube.com/watch?v=plVLRashaA8>
- <https://www.youtube.com/watch?v=qWVTxfLq2ak>
- <https://www.youtube.com/watch?v=zD2focOkQ48>
- <https://www.youtube.com/watch?v=wGaGmosj04M>

Thank You.