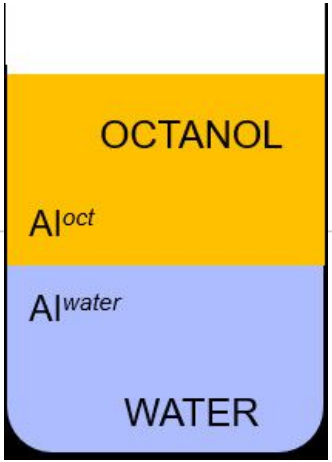
CM4044 Project 2
LogP Prediction

Khoo Kai Shin Lynette
Ong Kai Min Kayne
Wayne Sow



$$LogP = Log \frac{[Analyte]_{octanol}}{[Analyte]_{water}}$$

LogP, or octanol-water partition coefficient, is a measure of how hydrophilic or hydrophobic a molecule is.

Lipinski's Rule of 5

- Guidelines for bioavailability of oral drugs
- MW is less than 500 Da
- LogP < 5
- Sum of N and O atoms < 10
- Hydrogen-bond donors <5

Aim

Create an algorithm **f(x)** that takes the **SMILES** data as input and returns its **logP value** as the output for prediction.

Models used for prediction

Ridge Regression

Neural Network Model (Activation function = ReLU)

Support Vector Regression

Neural Network Model (Activation function = Sigmoid)

Multiple Linear Regression

Machine learning models used for prediction

Ridge Regression

Shrinks the estimated coefficients in the learning process to discourage overfitting

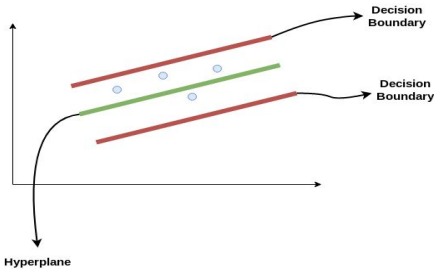
By adding 'squared magnitude' of coefficients as penalty term to loss function

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x_i))^2 + \lambda\sum_{j}^{p} w_j^2$$

Support Vector Regression

A hyperplane is defined along with two decision boundaries (ϵ) above and below the hyperplane

Kernel can be tuned as well for finding the ideal hyperplane for your dataset



Multiple Linear Regression

Estimates the relationship between a quantitative dependent variable and two or more independent variables using a straight line

$$y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \epsilon$$

Activation functions used for neural network models (NNM)

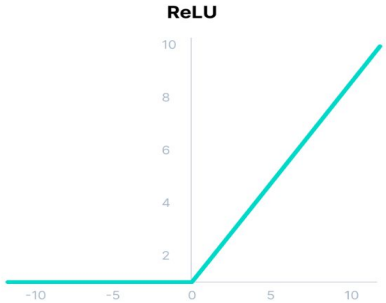
ReLU

Has a derivative function and allows for backpropagation

Neurons will be deactivated if the output is less than 0

Dying ReLU problem: inputs are negative, gradient becomes 0

Weights and biases for some neurons are not updated

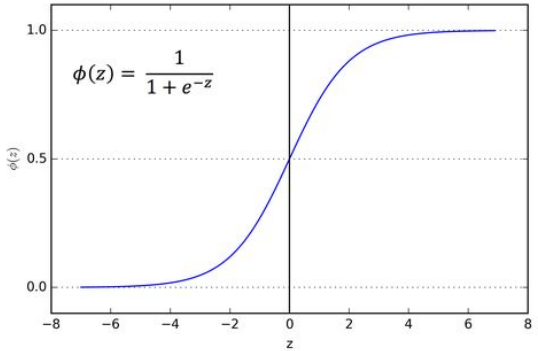


Sigmoid

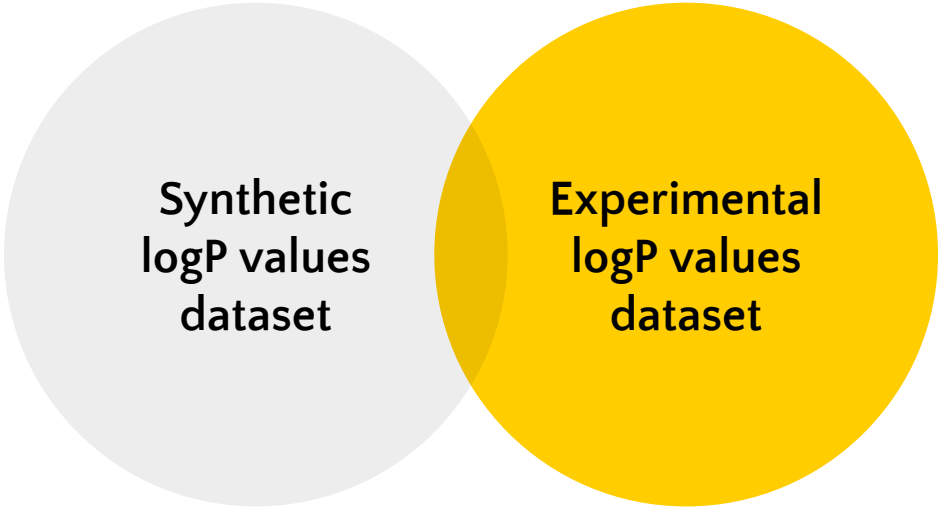
Output exists between 0 to 1

Commonly used for models where we have to predict the probability as an output

Vanishing gradient problem at very high or low values of input



Datasets used for prediction



*Note that they are completely different datasets containing different compounds

Importing datasets

Dataset with synthetic logP values

```
df = pd.read_csv('logP_dataset.csv', names=['smiles', 'logP'])
print(df.shape)
print(df.head())
```

```
(14610, 2)
                                        smiles  logP
0                          C[C@H]([C@@H](C)Cl)Cl   2.3
1                                    C(C=CBr)N   0.3
2                                    CCC(CO)Br   1.3
3  [13CH3][13CH2][13CH2][13CH2][13CH2][13CH2]O   2.0
4                                    CCCOCCP   0.6
```

Dataset with experimental logP values

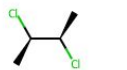
```
expt_df = pd.read_csv('logP_expt_dataset.csv', names=['smiles', 'logP'])
print(expt_df.shape)
print(expt_df.head())
```

```
(753, 2)
                                          smiles  logP
0          Cc1cc2c(cc1C)NC(=O)C[C@H]2c3ccccc3OC  4.17
1        COc1ccc2c(c1)O[C@@](CC2=O)(C(F)(F)F)O  2.79
2  CC1(O[C@H]([C@H](O1)C(=O)N)C(=O)N)C(C)(C)C  1.60
3        CCOc1cc(cc(c1OCC)OCC)c2nnc(o2)c3ccco3  3.96
4          CN(C)c1ccc(cc1)C(=C)c2ccc(cc2)N(C)C  5.30
```

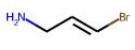
- Using RD-Kit package, SMILES notation is converted to *Mol* (a python object)

Visualization of compounds

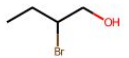
Dataset with synthetic logP values



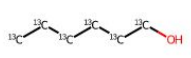
C[C@H][C@@H](C)Cl)Cl



C(C=CBr)N

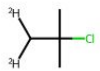


CCC(CO)Br



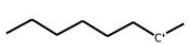
[13CH3][13CH2][13CH2][13CH2][13CH2][13CH2

C(C(F)(F)F)F



[2H]C([2H])C(C)(C)Cl

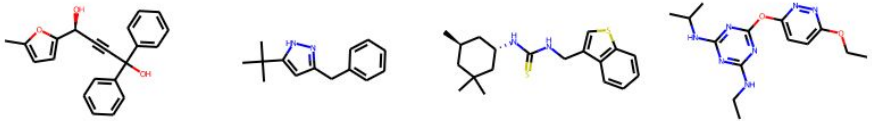
CCCC(Cl)O



CCCCCC[CH+]C

Dataset with experimental logP values

Cc1cc2c(cc1C)NC(=O)C[C@H]2c3ccccc3OCCOc1ccc2c(c1)O[C@@](CC2=O)(C(F)(F)F)OCC1(O[C@H][C@H](O1)C(=O)N)C(C)(C)CCCOc1cc(cc(c1OCC)OCC)c2nnc(o2)c3ccc



Cc1ccc(o1)[C@H](C#CC(c2ccccc2)(c3ccccc3)O)O CC(C)(C)c1cc(n[nH]1)Cc2ccccc2 C[C@H]1C[C@@H](CC(C1)(C)C)NC(=S)NCc2csc3c2cccc3CCNc1nc(nc(n1)Oc2ccc(nn2)OCC)NC(C

Visualization of compounds

Creating features for models (same for both datasets)

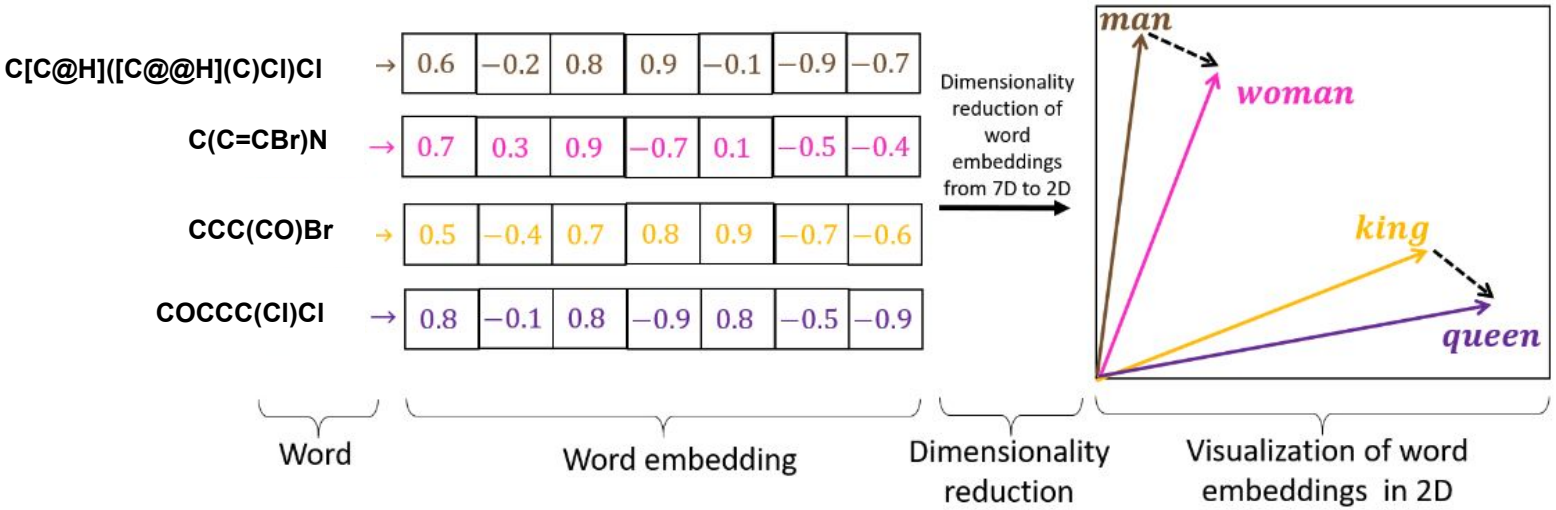
1. Number of atoms
2. Number of heavy atoms
3. Number of carbon atoms
4. Number of oxygen atoms
5. Number of nitrogen atoms
6. Number of chlorine atoms
7. Number of phosphorus atoms
8. Number of bromine atoms
9. Number of fluorine atoms
10. TPSA
11. Molecular weight
12. Number of valence electrons
13. Number of heteroatoms
14. Mol2Vec sentences

```python
# AddHs function adds H atoms to a MOL (as Hs in SMILES are usualy ignored)
df['mol'] = df['mol'].apply(lambda x: Chem.AddHs(x))
```

```python
# GetNumAtoms() method returns a general nubmer of all atoms in a molecule
df['num_of_atoms'] = df['mol'].apply(lambda x: x.GetNumAtoms())

# GetNumHeavyAtoms() method returns a nubmer of all atoms in a molecule with molecular weig
df['num_of_heavy_atoms'] = df['mol'].apply(lambda x: x.GetNumHeavyAtoms())

#Searching for patterns and use it for a list of most common atoms only
def number_of_atoms(atom_list, df):
    for i in atom_list:
        df['num_of_{}_atoms'.format(i)] = df['mol'].apply(lambda x: len(x.GetSubstructMatch

number_of_atoms(['C', 'O', 'N', 'Cl', 'P', 'Br', 'F'], df)

df['tpsa'] = df['mol'].apply(lambda x: Descriptors.TPSA(x)) #https://en.wikipedia.org/wiki/
df['mol_w'] = df['mol'].apply(lambda x: Descriptors.ExactMolWt(x)) # https://en.wikipedia.c
df['num_valence_electrons'] = df['mol'].apply(lambda x: Descriptors.NumValenceElectrons(x))
df['num_heteroatoms'] = df['mol'].apply(lambda x: Descriptors.NumHeteroatoms(x))
```

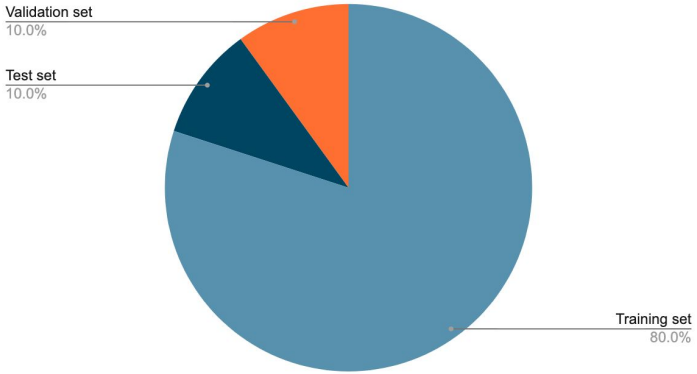
Creating features for models (same for both datasets)

- Mol2Vec converts *Mol* (python object) into a vector (a list of numbers) that we will use as an additional feature of the model.



Splitting datasets (same for both datasets)

- X > Features
- y > LogP values
- Both datasets are then split into the subsequent subsets:
  - Training set (0.8)
  - Validation set (0.1)
  - Test set (0.1)

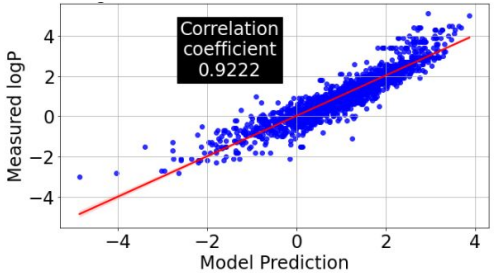



```
1 X_train, X_remain, y_train, y_remain = train_test_split(X, y, test_size=.2, randor
2 X_val, X_test, y_val, y_test = train_test_split(X_remain, y_remain, test_size=.5,
```

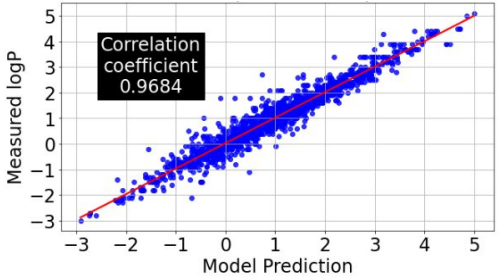
Evaluation

Utilizing **MSE** and **Correlation Coefficients** to evaluate the **ML** and **NN** models

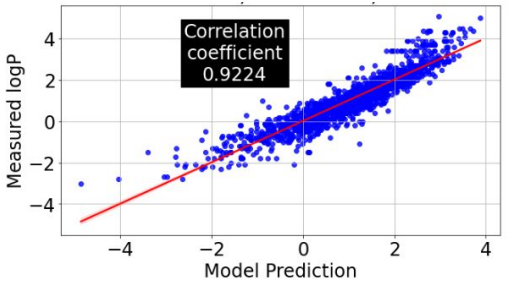
Plots of Synthetic LogP Data



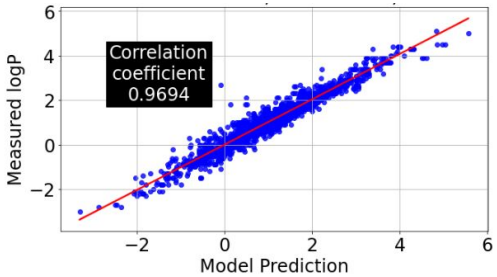
Ridge



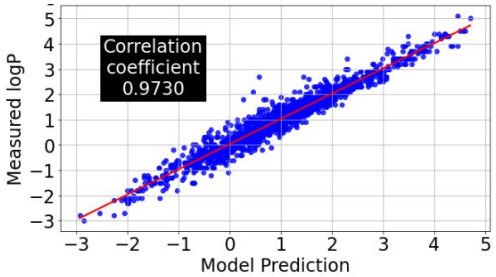
SVM



MLR



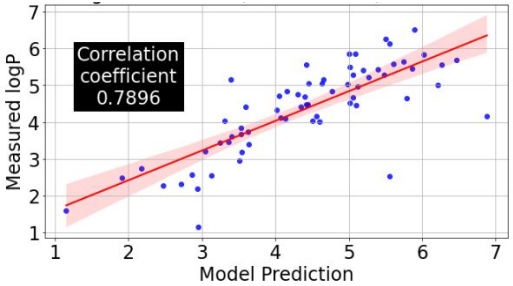
ReLU



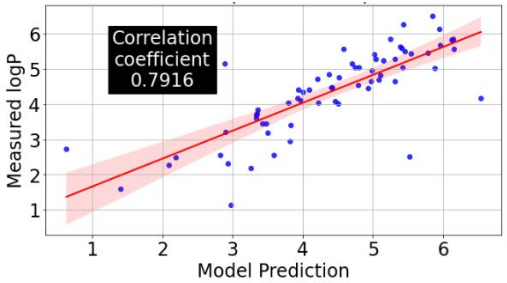
Sigmoid

*14610 Data Points

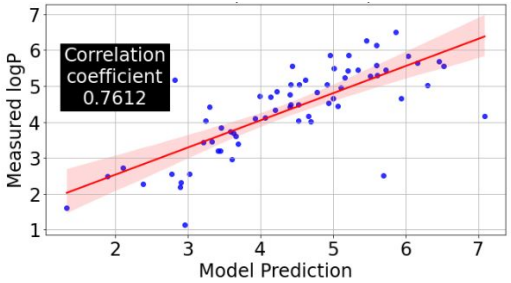
Plots of Experimental LogP Data



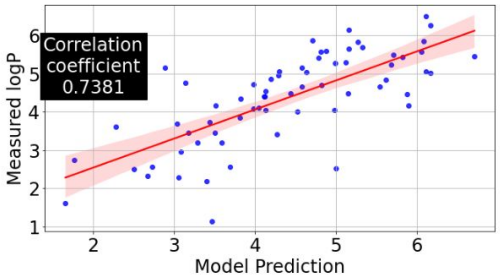
Ridge



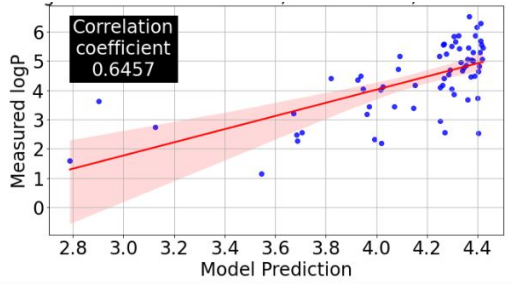
SVM



MLR



ReLU



Sigmoid

*753 Data Points

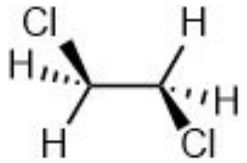
Evaluation Parameters Table

		Ridge Regression	Support Vector Regressor	Multiple Linear Regression	ReLU NNM	Sigmoid NNM
MSE	Synthetic dataset	0.256	0.107	0.256	0.104	0.092
	Experimental dataset	0.591	0.598	0.691	0.730	1.059
Correlation coefficient	Synthetic dataset	0.9222	0.9684	0.9224	0.9694	0.9730
	Experimental dataset	0.7896	0.7916	0.7612	0.7381	0.6457

Limitations

- Data Sets
  - Synthetic
    - A “Predicted” Model
    - Based on Heteroatoms
  - Experimental
    - Small Sample Size
    - No External Reaction Parameters

- Real-life Deviations
  - Intermolecular Interactions
  - Intramolecular Interactions
  - Entropic Considerations



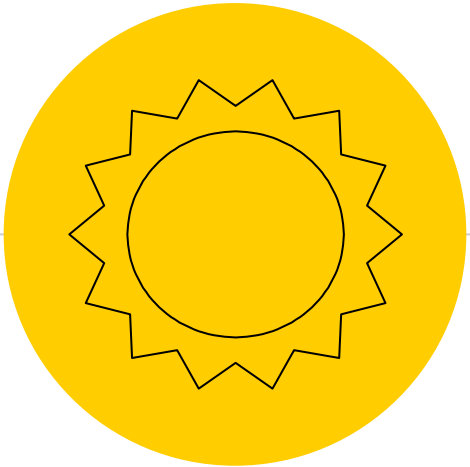
1,2-dichloroethane

Computed logP = 2.3

Experimental logP = 1.48

Conclusion

- LogP
  - Impact for Medicinal Chemistry
- Machine and Neural Network Models
  - Ridge Regression
  - Support Vector Regression
  - Multiple Linear Regression
  - ReLU Function NNM
  - Sigmoid Function NNM
- Predictions
  - Synthetic
  - Experimental
- Evaluation
  - Mean Squared Error
  - Correlation Coefficient
- Limitations
  - Data Sets
  - Experimental Deviations



Thank You!