Predictive Modeling of Molecular Solubility: A Comparative Study of Linear Regression and Random Forest Regressor

Import pandas library

In [65]:
```python
import pandas as pd
```

Loading the data

In [66]:
```python
df = pd.read_csv('https://raw.githubusercontent.com/dataprofessor/data/master/delaney_solubility_with_descriptors.csv')
```

In [67]:
```python
df
```

Out[67]:

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion	logS
0	2.59540	167.850	0.0	0.000000	-2.180
1	2.37650	133.405	0.0	0.000000	-2.000
2	2.59380	167.850	1.0	0.000000	-1.740
3	2.02890	133.405	1.0	0.000000	-1.480
4	2.91890	187.375	1.0	0.000000	-3.040
...	...	...	...	...	...
1139	1.98820	287.343	8.0	0.000000	1.144
1140	3.42130	286.114	2.0	0.333333	-4.925
1141	3.60960	308.333	4.0	0.695652	-3.893
1142	2.56214	354.815	3.0	0.521739	-3.790
1143	2.02164	179.219	1.0	0.461538	-2.581

1144 rows × 5 columns

Data preperation as X and Y

In [68]:
```python
y = df["logS"]
```

In [69]:
```python
y
```

Out[69]:
```
0      -2.180
1      -2.000
2      -1.740
3      -1.480
4      -3.040
        ...  
1139    1.144
1140   -4.925
1141   -3.893
1142   -3.790
1143   -2.581
Name: logS, Length: 1144, dtype: float64
```

Removing the 'logS' for x

In [70]:
```python
x = df.drop('logS',axis=1)
```

In [71]:
```python
x
```

Out[71]:

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion
0	2.59540	167.850	0.0	0.000000
1	2.37650	133.405	0.0	0.000000
2	2.59380	167.850	1.0	0.000000
3	2.02890	133.405	1.0	0.000000
4	2.91890	187.375	1.0	0.000000
...	...	...	...	...
1139	1.98820	287.343	8.0	0.000000
1140	3.42130	286.114	2.0	0.333333
1141	3.60960	308.333	4.0	0.695652
1142	2.56214	354.815	3.0	0.521739
1143	2.02164	179.219	1.0	0.461538

1144 rows × 4 columns

Splitting the data for Training and Testing using Scikit learn library

In [72]:
```python
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=100)
```

In [73]:
```python
x_train
```

Out[73]:

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion
107	3.14280	112.216	5.0	0.000000
378	-2.07850	142.070	0.0	0.000000
529	-0.47730	168.152	0.0	0.000000
546	-0.86740	154.125	0.0	0.000000
320	1.62150	100.161	2.0	0.000000
...	...	...	...	...
802	3.00254	250.301	1.0	0.842105
53	2.13860	82.146	3.0	0.000000
350	5.76304	256.348	0.0	0.900000
79	3.89960	186.339	10.0	0.000000
792	2.52334	310.297	3.0	0.300000

915 rows × 4 columns

In [74]:
```python
x_test
```

Out[74]:

	MolLogP	MolWt	NumRotatableBonds	AromaticProportion
822	2.91000	172.268	7.0	0.000000
118	7.27400	360.882	1.0	0.666667
347	1.94040	145.161	0.0	0.909091
1123	1.98640	119.378	0.0	0.000000
924	1.70062	108.140	0.0	0.750000
...	...	...	...	...
1114	1.76210	478.513	4.0	0.000000
427	6.32820	276.338	0.0	1.000000
711	0.04430	218.205	5.0	0.000000
4	2.91890	187.375	1.0	0.000000
948	3.56010	318.328	2.0	0.750000

229 rows × 4 columns

Model building

Linear Regression

Importing the model

In [ ]:
```python
from sklearn.linear_model import LinearRegression
```

Training the model

In [ ]:
```python
lr = LinearRegression()
lr.fit(x_train,y_train)
```

Applying the model to make prediction

In [76]:
```python
y_lr_train_pred = lr.predict(x_train)
y_lr_test_pred = lr.predict(x_test)
```

In [77]:
```python
y_lr_train_pred
```

Out[77]:
```
array([-2.83668969,  0.84208401, -0.50930302, -0.13058053, -1.6269208 ,
       -2.86103037, -2.32208333, -4.16684323, -5.04235077, -2.29737009,
[illegible]
       -3.98606913, -1.89546347, -6.68789325, -3.88711135, -3.78407726])
```

In [78]:
```python
y_lr_test_pred
```

Out[78]:
```
array([-3.05722870e+00, -7.77785827e+00, -2.55016650e+00, -2.01523582e+00,
[illegible]
       -4.79863925e+00])
```

Evaluate model perfomance

In [79]:
```python
y_train
```

Out[79]:
```
107   -4.440
378   -1.250
529   -1.655
546   -1.886
320   -0.740
       ...  
802   -2.925
53    -2.680
350   -7.020
79    -4.800
792   -3.240
Name: logS, Length: 915, dtype: float64
```

In [80]:
```python
y_lr_train_pred
```

Out[80]:
```
array([-2.83668969,  0.84208401, -0.50930302, -0.13058053, -1.6269208 ,
       -2.86103037, -2.32208333, -4.16684323, -5.04235077, -2.29737009,
[illegible]
       -3.98606913, -1.89546347, -6.68789325, -3.88711135, -3.78407726])
```

Model perfomance evaluation

In [81]:
```python
from sklearn.metrics import mean_squared_error, r2_score

lr_train_mse = mean_squared_error(y_train, y_lr_train_pred) #  y_lr_train_pred = x_train
lr_train_r2 = r2_score(y_train, y_lr_train_pred)

lr_test_mse= mean_squared_error(y_test, y_lr_test_pred)      # y_lr_test_pred = x_test
lr_test_r2 = r2_score(y_test, y_lr_test_pred)
```

In [82]:
```python
print('LR MSE (train): ',lr_train_mse)
print( 'LR R2 (train): ',lr_train_r2)
print('LR MSE (test): ',lr_test_mse)
print( 'LR R2 (test): ',lr_test_r2)
```

```
LR MSE (train):  1.0075362951019069
LR R2 (train):  0.7645051774663391
LR MSE (test):  1.0206953660861033
LR R2 (test):  0.7891616188563282
```

In [83]:
```python
lr_results = pd.DataFrame(['Linear regression', lr_train_mse,lr_train_r2,lr_test_mse,lr_test_r2]).transpose()
lr_results.columns = ["method","training mse","training r2","test mse","test r2"]
```

In [84]:
```python
lr_results
```

Out[84]:

	method	training mse	training r2	test mse	test r2
0	Linear regression	1.007536	0.764505	1.020695	0.789162

Random forest

Importing the model

In [98]:
```python
from sklearn.ensemble import RandomForestRegressor
```

Model building

In [99]:
```python
rf = RandomForestRegressor(max_depth=2, random_state=100)
```

Out[99]:
```
RandomForestRegressor
RandomForestRegressor(max_depth=2, random_state=100)
```

Training the model

In [ ]:
```python
rf.fit(x_train, y_train)
```

Applying the model to make a prediction

In [88]:
```python
y_rf_train_pred = rf.predict(x_train)
y_rf_test_pred = rf.predict(x_test)
```

Evaluate model perfomance

In [100…
```python
from sklearn.metrics import mean_squared_error, r2_score

rf_train_mse = mean_squared_error(y_train, y_rf_train_pred)
rf_train_r2 = r2_score(y_train, y_rf_train_pred)

rf_test_mse= mean_squared_error(y_test, y_rf_test_pred)
rf_test_r2 = r2_score(y_test, y_rf_test_pred)
```

In [101…
```python
rf_results = pd.DataFrame(['Random forest', rf_train_mse,rf_train_r2,rf_test_mse,rf_test_r2]).transpose()
rf_results.columns = ["method","training mse","training r2","test mse","test r2"]
```

In [106…
```python
rf_results
```

Out[106…

	method	training mse	training r2	test mse	test r2
0	Random forest	1.028228	0.759669	1.407688	0.709223

Model comparison between Liner Regression and Random forest

In [107…
```python
df_models = pd.concat([lr_results, rf_results],axis=0).reset_index(drop=True) # "axis=0" to combine as a row
df_models                                                                     # "axis=1" to combine as a column
```

Out[107…

	method	training mse	training r2	test mse	test r2
0	Linear regression	1.007536	0.764505	1.020695	0.789162
1	Random forest	1.028228	0.759669	1.407688	0.709223

In [108…
```python
#df_models.reset_index(drop=True)
```

Data visulaisation of prediction results

Import matplot library and Numpy

In [ ]:
```python
import matplotlib.pyplot as plt
import numpy as np     #To draw the trend line
```

In [109…
```python
plt.figure(figsize=(5,5)) # size of the graph
plt.scatter(x=y_train,y=y_lr_train_pred, c="#7CAE00" ,alpha=0.3) # "C=#7CAE00" represents the Green colour

z = np.polyfit(y_train, y_lr_train_pred, 1)
p = np.poly1d(z)    # To create an polynomial object

plt.plot(y_train,p(y_train), '#F8766D')
plt.ylabel('Predict LogS')       # y label
plt.xlabel('Experimental LogS')  # x label
```

Out[109…
```
Text(0.5, 0, 'Experimental LogS')
```