PythonByCasviRandom

September 4, 2024

[35]:
```
import pandas as pd
!pip install scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: scikit-learn in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from scikit-learn) (2.1.0)
Requirement already satisfied: scipy>=1.6.0 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from scikit-learn) (3.5.0)
```

[36]:
```
df=pd.read_csv('https://raw.githubusercontent.com/dataprofessor/data/master/
↪delaney_solubility_with_descriptors.csv')
```

[37]:
```
df
```

[37]:

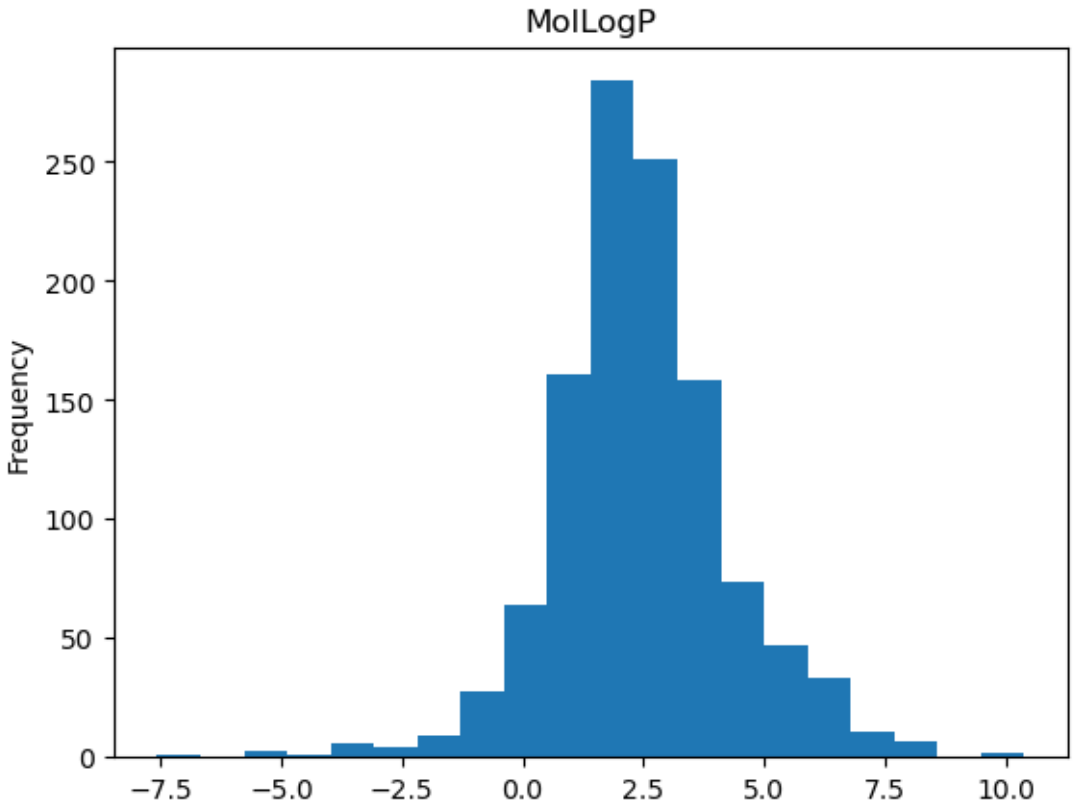
	MolLogP	MolWt	NumRotatableBonds	AromaticProportion	logS
0	2.59540	167.850	0.0	0.000000	-2.180
1	2.37650	133.405	0.0	0.000000	-2.000
2	2.59380	167.850	1.0	0.000000	-1.740
3	2.02890	133.405	1.0	0.000000	-1.480
4	2.91890	187.375	1.0	0.000000	-3.040
…	…	…	…	…	…
1139	1.98820	287.343	8.0	0.000000	1.144
1140	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 x 5 columns]

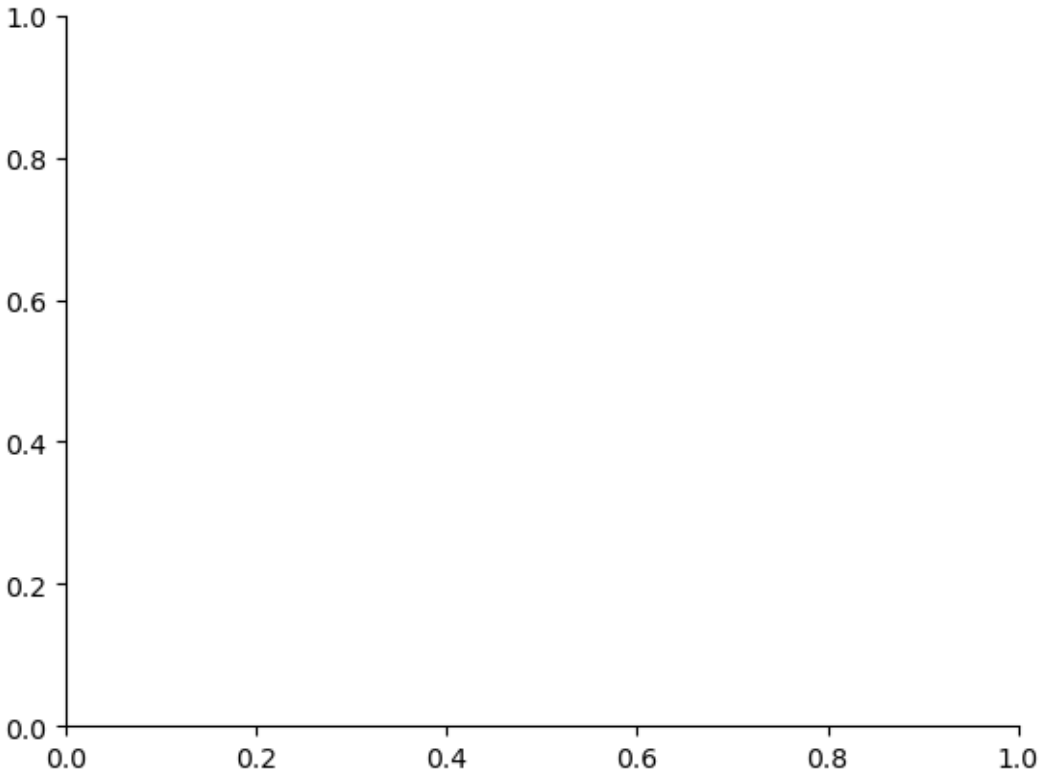
[38]:
```
from matplotlib import pyplot as plt
```

[39]:
```
df['MolLogP'].plot(kind='hist', bins=20, title='MolLogP')
```

[39]: <Axes: title={'center': 'MolLogP'}, ylabel='Frequency'>



[40]:
```
plt.gca().spines[['top', 'right',]].set_visible(False)
```



[41]:
```
import pandas as pd
from sklearn.model_selection import train_test_split


y= df['logS']
x= df.drop('logS', axis=1)


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)
```

[42]:
```
x_train
```

[42]:

	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 x 4 columns]
```

[43]:
```
x_test
```

[43]:
```
      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 x 4 columns]
```

[44]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train, y_train)
```

[44]:
```
LinearRegression()
```

[45]:
```
y_train_pred = lr.predict(x_train)
y_test_pred = lr.predict(x_test)
```

[46]:
```
y_lr_test_pred = lr.predict(x_test)
y_lr_test_pred  # Now you can display the predictions
```

[46]:
```
array([-3.05722870e+00, -7.77785827e+00, -2.55016650e+00, -2.01523582e+00,
       -2.06375990e+00, -9.99672215e-01, -5.94603364e-01, -5.53626003e-01,
       -5.72200956e+00, -3.94006681e+00, -3.95496755e+00, -2.29737009e+00,
       -1.48980354e+00, -1.48988982e+00, -4.64510806e+00, -1.90396018e+00,
       -1.51566313e+00, -3.16424605e+00, -3.70863920e+00, -5.58105660e+00,
       -3.25038467e+00, -5.04235077e+00, -5.69194881e+00, -2.14339849e+00,
       -4.35689341e+00, -5.03964756e+00, -3.10383618e+00, -4.40286964e+00,
       -4.21276272e+00,  5.56508349e-01, -1.45537678e+00, -4.41027396e+00,
       -2.59668773e+00, -1.53336276e+00, -5.55749874e-01, -1.67111795e+00,
       -2.78163675e+00, -3.15395565e+00, -5.27083361e+00, -1.75321446e+00,
```

-1.53350725e+00, -2.01255666e+00, -6.57559167e+00, -7.89433046e+00,
-5.76437127e+00, -4.16422068e+00, -3.43694663e+00, 1.43834212e+00,
-1.12679105e-02, -2.34521849e+00, -1.86480046e+00, -5.03964756e+00,
 8.55886378e-01, -3.17679292e+00, -5.06764094e+00, -1.99464442e+00,
-7.77785827e+00, -1.21764693e+00, -9.09541075e-01, -5.04235077e+00,
-2.43898748e+00, -2.84034045e+00, -2.53403538e+00, -2.36170311e+00,
-1.63103729e+00, -1.53182046e+00, -3.23931568e+00, -2.88008616e+00,
-1.88300518e+00, -3.21582220e+00, -3.40245202e+00, -9.01813905e-01,
-4.82308940e+00, -7.69116343e-01, -7.12894308e+00, -1.05440427e+01,
-1.95444152e+00, -3.50194744e+00, -7.18167736e+00, -6.01555673e+00,
-2.08189806e+00, -2.31652280e+00, -3.44556948e+00, -2.05480142e+00,
-6.01555673e+00, -2.88308299e+00, -4.84867198e+00, -3.51006495e-01,
-3.54726250e+00, -1.21057919e+00, -4.36658559e+00, -4.21815903e-01,
-1.63103729e+00, -2.51604291e+00, -2.16707077e+00, -1.48726025e+00,
-3.20864450e+00, -1.51411141e+00, -1.65033691e+00, -3.66287663e+00,
-3.26968347e+00, -3.94492313e+00, -4.22509088e+00, -3.68794650e+00,
-5.98734972e+00, -1.43710934e+00, -1.97653920e+00, -1.85076729e+00,
-1.14179382e+00, -3.07730828e+00, -4.84867198e+00, -2.19679345e+00,
-1.68737438e+00, -2.20390218e+00, 1.89408269e+00, -3.61322115e+00,
-2.79173430e+00, -2.41564138e+00, -7.53910534e-01, -8.54744860e-01,
-9.20407401e-02, -6.14209981e+00, -3.79386016e+00, -7.77785827e+00,
-1.79874130e+00, -2.50544035e+00, -3.77102985e+00, -2.25250766e+00,
-2.57788713e+00, -2.06375990e+00, -3.33843958e+00, -1.03912484e+00,
-6.68946164e+00, -1.91112045e+00, -2.58735850e+00, -2.19902800e+00,
-1.90219551e+00, -2.81396751e+00, -4.16972153e+00, -5.72200956e+00,
-1.60816482e+00, -3.68121117e+00, -4.60929775e+00, -2.45888480e+00,
 1.13185484e+00, -1.69279625e+00, -7.09025955e+00, -3.79386016e+00,
-2.99712058e+00, -5.79600137e+00, -2.44845780e+00, -5.29399242e+00,
-5.29389899e-01, -3.53652118e+00, -3.51200413e+00, -2.02419300e+00,
-4.47466933e+00, -3.63836536e+00, -4.56596939e+00, -5.57842703e+00,
-5.30676136e+00, -2.39225449e+00, -4.88290144e+00, -2.61359308e+00,
-3.11841945e+00, -2.05580278e+00, -1.64987419e+00, -4.91881901e+00,
-3.93774254e+00, -4.26411548e+00, -3.15082845e+00, -3.49352203e+00,
-3.81768831e+00, -3.77197350e+00, -2.55016650e+00, -1.97653920e+00,
-2.59432621e+00, -5.38480406e+00, -5.44932525e+00, -3.04107137e+00,
-1.87252408e+00, -2.25124657e+00, -2.89215707e+00, -3.46087334e+00,
-6.06861986e+00, -1.89916369e+00, -1.98035105e+00, -2.45036038e+00,
-2.79393037e+00, -4.76010415e+00, -1.72379306e+00, -7.09025955e+00,
-2.86880150e+00, -2.70674744e+00, -4.36825704e+00, -3.11841945e+00,
-3.85805633e+00, 9.78662246e-03, -4.47466933e+00, -6.08708502e+00,
-5.19970454e+00, -6.40483191e+00, -4.10155218e+00, -1.07044752e+00,
 1.99249372e+00, -3.63698515e+00, -8.50933009e-01, -3.26770298e+00,
-4.78343575e+00, -1.48988982e+00, -2.24629276e+00, -4.13247222e+00,
-4.36873484e+00, -1.89724815e+00, -1.50596465e+00, -1.16713539e-02,
-1.73602998e+00, -2.34521849e+00, -4.54942814e-01, -4.18812419e+00,
-1.87770440e+00, -3.70838271e+00, -1.59899899e+00, -3.26143822e+00,
-4.17623614e+00, -6.67872053e+00, -1.23069039e+00, -3.14545964e+00,

```
        -4.79863925e+00])
```

[47]:
```
y_lr_test_pred
```

[47]:
```
array([-3.05722870e+00, -7.77785827e+00, -2.55016650e+00, -2.01523582e+00,
       -2.06375990e+00, -9.99672215e-01, -5.94603364e-01, -5.53626003e-01,
       -5.72200956e+00, -3.94006681e+00, -3.95496755e+00, -2.29737009e+00,
       -1.48980354e+00, -1.48988982e+00, -4.64510806e+00, -1.90396018e+00,
       -1.51566313e+00, -3.16424605e+00, -3.70863920e+00, -5.58105660e+00,
       -3.25038467e+00, -5.04235077e+00, -5.69194881e+00, -2.14339849e+00,
       -4.35689341e+00, -5.03964756e+00, -3.10383618e+00, -4.40286964e+00,
       -4.21276272e+00,  5.56508349e-01, -1.45537678e+00, -4.41027396e+00,
       -2.59668773e+00, -1.53336276e+00, -5.55749874e-01, -1.67111795e+00,
       -2.78163675e+00, -3.15395565e+00, -5.27083361e+00, -1.75321446e+00,
       -1.53350725e+00, -2.01255666e+00, -6.57559167e+00, -7.89433046e+00,
       -5.76437127e+00, -4.16422068e+00, -3.43694663e+00,  1.43834212e+00,
       -1.12679105e-02, -2.34521849e+00, -1.86480046e+00, -5.03964756e+00,
        8.55886378e-01, -3.17679292e+00, -5.06764094e+00, -1.99464442e+00,
       -7.77785827e+00, -1.21764693e+00, -9.09541075e-01, -5.04235077e+00,
       -2.43898748e+00, -2.84034045e+00, -2.53403538e+00, -2.36170311e+00,
       -1.63103729e+00, -1.53182046e+00, -3.23931568e+00, -2.88008616e+00,
       -1.88300518e+00, -3.21582220e+00, -3.40245202e+00, -9.01813905e-01,
       -4.82308940e+00, -7.69116343e-01, -7.12894308e+00, -1.05440427e+01,
       -1.95444152e+00, -3.50194744e+00, -7.18167736e+00, -6.01555673e+00,
       -2.08189806e+00, -2.31652280e+00, -3.44556948e+00, -2.05480142e+00,
       -6.01555673e+00, -2.88308299e+00, -4.84867198e+00, -3.51006495e-01,
       -3.54726250e+00, -1.21057919e+00, -4.36658559e+00, -4.21815903e-01,
       -1.63103729e+00, -2.51604291e+00, -2.16707077e+00, -1.48726025e+00,
       -3.20864450e+00, -1.51411141e+00, -1.65033691e+00, -3.66287663e+00,
       -3.26968347e+00, -3.94492313e+00, -4.22509088e+00, -3.68794650e+00,
       -5.98734972e+00, -1.43710934e+00, -1.97653920e+00, -1.85076729e+00,
       -1.14179382e+00, -3.07730828e+00, -4.84867198e+00, -2.19679345e+00,
       -1.68737438e+00, -2.20390218e+00,  1.89408269e+00, -3.61322115e+00,
       -2.79173430e+00, -2.41564138e+00, -7.53910534e-01, -8.54744860e-01,
       -9.20407401e-02, -6.14209981e+00, -3.79386016e+00, -7.77785827e+00,
       -1.79874130e+00, -2.50544035e+00, -3.77102985e+00, -2.25250766e+00,
       -2.57788713e+00, -2.06375990e+00, -3.33843958e+00, -1.03912484e+00,
       -6.68946164e+00, -1.91112045e+00, -2.58735850e+00, -2.19902800e+00,
       -1.90219551e+00, -2.81396751e+00, -4.16972153e+00, -5.72200956e+00,
       -1.60816482e+00, -3.68121117e+00, -4.60929775e+00, -2.45888480e+00,
        1.13185484e+00, -1.69279625e+00, -7.09025955e+00, -3.79386016e+00,
       -2.99712058e+00, -5.79600137e+00, -2.44845780e+00, -5.29399242e+00,
       -5.29389899e-01, -3.53652118e+00, -3.51200413e+00, -2.02419300e+00,
       -4.47466933e+00, -3.63836536e+00, -4.56596939e+00, -5.57842703e+00,
       -5.30676136e+00, -2.39225449e+00, -4.88290144e+00, -2.61359308e+00,
       -3.11841945e+00, -2.05580278e+00, -1.64987419e+00, -4.91881901e+00,
       -3.93774254e+00, -4.26411548e+00, -3.15082845e+00, -3.49352203e+00,
```

```
-3.81768831e+00, -3.77197350e+00, -2.55016650e+00, -1.97653920e+00,
-2.59432621e+00, -5.38480406e+00, -5.44932525e+00, -3.04107137e+00,
-1.87252408e+00, -2.25124657e+00, -2.89215707e+00, -3.46087334e+00,
-6.06861986e+00, -1.89916369e+00, -1.98035105e+00, -2.45036038e+00,
-2.79393037e+00, -4.76010415e+00, -1.72379306e+00, -7.09025955e+00,
-2.86880150e+00, -2.70674744e+00, -4.36825704e+00, -3.11841945e+00,
-3.85805633e+00,  9.78662246e-03, -4.47466933e+00, -6.08708502e+00,
-5.19970454e+00, -6.40483191e+00, -4.10155218e+00, -1.07044752e+00,
 1.99249372e+00, -3.63698515e+00, -8.50933009e-01, -3.26770298e+00,
-4.78343575e+00, -1.48988982e+00, -2.24629276e+00, -4.13247222e+00,
-4.36873484e+00, -1.89724815e+00, -1.50596465e+00, -1.16713539e-02,
-1.73602998e+00, -2.34521849e+00, -4.54942814e-01, -4.18812419e+00,
-1.87770440e+00, -3.70838271e+00, -1.59899899e+00, -3.26143822e+00,
-4.17623614e+00, -6.67872053e+00, -1.23069039e+00, -3.14545964e+00,
-4.79863925e+00])
```

[48]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Make predictions (if you haven't already)
y_lr_train_pred = lr.predict(x_train)
y_lr_test_pred = lr.predict(x_test)

# Calculate metrics
lr_train_mse = mean_squared_error(y_train, y_lr_train_pred)
lr_train_r2 = r2_score(y_train, y_lr_train_pred)
lr_test_mse = mean_squared_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)

# Print the results
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):  1.0075362951093687
LR R2 (Train):  0.7645051774663391
LR MSE (Test):  1.0206953660861033
LR R2 (Test):  0.7891616188563282
```

[49]:
```
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):  1.0075362951093687
LR R2 (Train):  0.7645051774663391
LR MSE (Test):  1.0206953660861033
LR R2 (Test):  0.7891616188563282
```

[50]:
```
lr_results = pd.DataFrame([ 'Linear regression' , lr_train_mse, lr_train_r2,␣
↪lr_test_mse, lr_test_r2]).transpose()
```

[51]:
```
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']
```

[52]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

rf = RandomForestRegressor()
rf.fit(x_train, y_train)

y_rf_train_pred = rf.predict(x_train)
y_rf_test_pred = rf.predict(x_test)

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)

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']
```

[53]:
```
rf_results
```

[53]:

	Method	Training MSE	Training R2	Test MSE	Test R2
0	Random Forest	0.085346	0.980052	0.634092	0.86902

[54]:
```
lr_results.columns = ['Method' , 'Training MSE' , 'Training R2', 'Test MSE' ,␣
↪'Test R2']
```

[55]:
```
lr_results
```

[55]:

	Method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	1.007536	0.764505	1.020695	0.789162

[56]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor() #create the RandomForestRegressor object
rf.fit(x_train, y_train)
```

[56]:
```
RandomForestRegressor()
```

[57]:
```
# prompt: import random numpy or pandas

import pandas as pd
import numpy as np
```

[58]:
```
y_rf_train_pred = rf.predict(x_train)
y_rf_test_pred = rf.predict(x_test)
```

[59]:
```
# prompt: 10 random numbers using numpy

import numpy as np
random_numbers = np.random.rand(10)
print(random_numbers)
```

```
[0.01292749 0.59930049 0.64971771 0.92813352 0.98965713 0.96061211
 0.45377325 0.16806282 0.60949223 0.6867629 ]
```

[60]:
```
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)
```

[61]:
```
# prompt: a slider using jupyter widgets

import ipywidgets as widgets
from IPython.display import display

# Create a slider
slider = widgets.IntSlider(
    value=7,
    min=0,
    max=10,
    step=1,
    description='Value:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='d'
)

# Display the slider
display(slider)
```

```
IntSlider(value=7, continuous_update=False, description='Value:', max=10)
```

[62]:
```
y_train
```

[62]:
```
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
```

[63]:
```
new_var0 =  df_models = pd.concat([ lr_results, rf_results], axis=0)
df_models # remove any extra spaces before this line
new_var0
```

[63]:

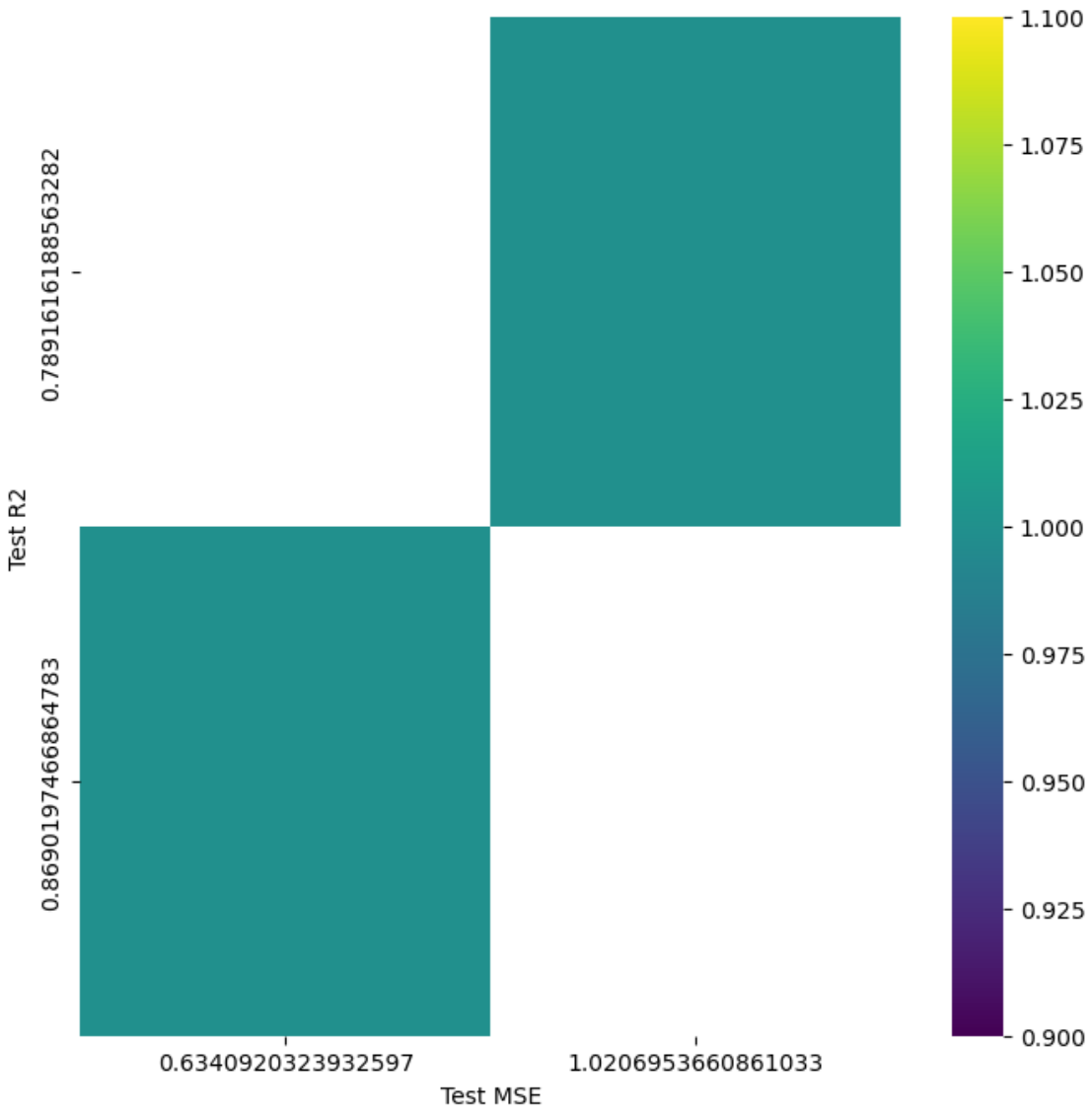
	Method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	1.007536	0.764505	1.020695	0.789162
0	Random Forest	0.085346	0.980052	0.634092	0.86902

[64]:
```
# @title Test MSE vs Test R2

from matplotlib import pyplot as plt
!pip install seaborn -i https://pypi.tuna.tsinghua.edu.cn/simple
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['Test R2'].value_counts()
    for x_label, grp in df_models.groupby('Test MSE')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Test MSE')
_ = plt.ylabel('Test R2')
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: seaborn in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from seaborn) (2.1.0)
Requirement already satisfied: pandas>=1.2 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.2.1)
```

```
Requirement already satisfied: cycler>=0.10 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from pandas>=1.2->seaborn)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from pandas>=1.2->seaborn)
(2024.1)
Requirement already satisfied: six>=1.5 in ./.config/jupyterlab-
desktop/jlab_server/lib/python3.12/site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```



[65]:
```
# prompt: Using dataframe df_models: convert Pandas to LaTex

print(df_models.to_latex(index=False)) #Converts the dataframe to a LaTeX table␣
↪and prints the output.
```

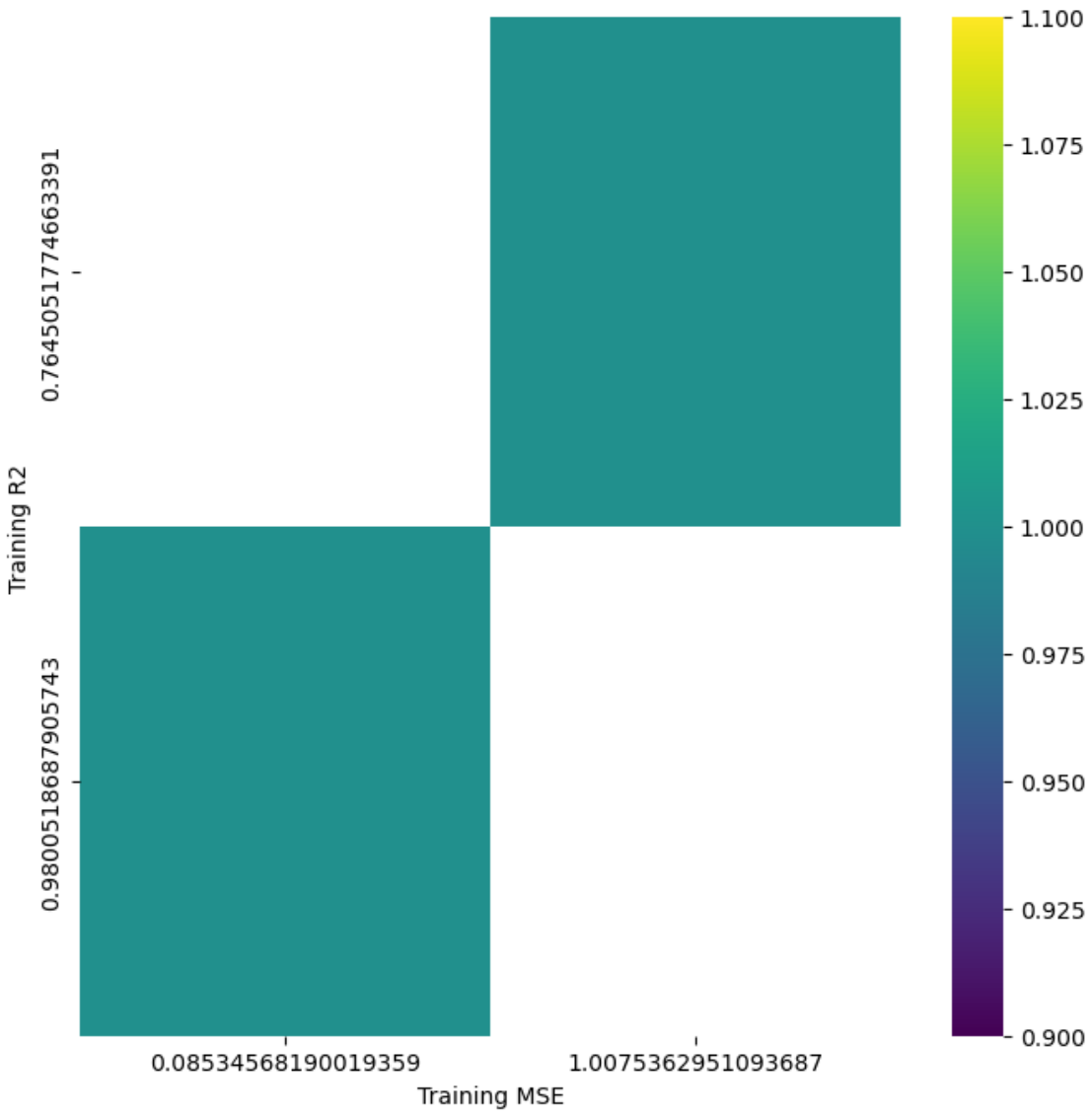
```
\begin{tabular}{lllll}
\toprule
Method & Training MSE & Training R2 & Test MSE & Test R2 \\
\midrule
Linear regression & 1.007536 & 0.764505 & 1.020695 & 0.789162 \\
Random Forest & 0.085346 & 0.980052 & 0.634092 & 0.869020 \\
\bottomrule
\end{tabular}
```

[66]:
```
df_models.reset_index(drop=True)
```

[66]:

	Method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	1.007536	0.764505	1.020695	0.789162
1	Random Forest	0.085346	0.980052	0.634092	0.86902

[67]:
```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['Training R2'].value_counts()
    for x_label, grp in df_models.groupby('Training MSE') # Use df_models here
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Training MSE')
_ = plt.ylabel('Training R2')
```

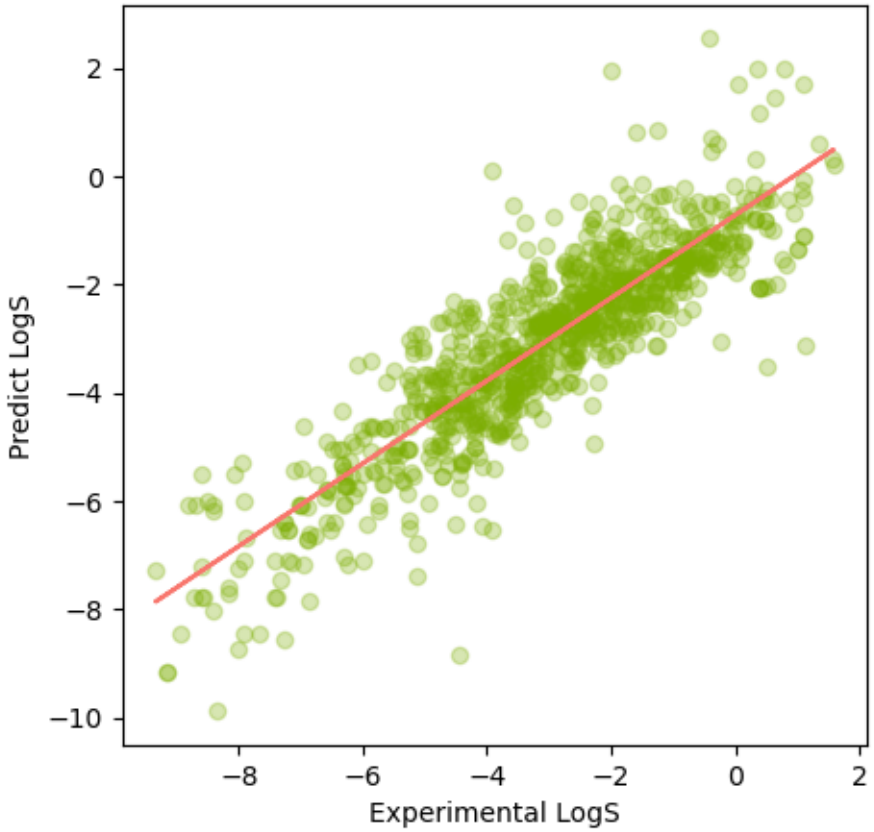


[68]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(5,5) )
plt.scatter(x=y_train, y=y_lr_train_pred, c="#7CAE00", alpha=0.3)

z = np.polyfit(y_train, y_lr_train_pred, 1)
p = np.poly1d(z)

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

[68]: Text(0.5, 0, 'Experimental LogS')



[ ]:

[ ]: