In [254...

```
import pandas as pd
from sklearn.model_selection import train_test_split

capacity = pd.read_excel('./DeltaGform.xlsx')

# selecting y and x axes, splitting the data into test and training sets
X = capacity.drop(columns=['deltaG'])
y = capacity['deltaG']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)
capacity
```

Out[254...

	δ	δP	Δχ	VEC	VECP	ΔHmix	ΔSmix	deltaG
0	7.055169	6.190800	9.333133	4.400000	4.400000	0.160000	13.381350	-83.600
1	6.815252	5.657277	5.161657	5.200000	5.000000	-7.129297	10.854000	-72.660
2	6.811313	5.651942	5.147014	5.186800	4.988000	-7.480043	11.093827	-72.660
3	6.808560	5.648297	5.137222	5.178000	4.980000	-7.711943	11.211074	-72.180
4	6.797329	5.633990	5.100297	5.145000	4.950000	-8.567218	11.554649	-69.930
...	...	...	...	...	...	...	...	...
184	6.625031	6.900708	3.950173	5.566667	5.266667	-6.038330	11.102053	-50.101
185	6.379784	7.108210	3.966640	5.633333	5.266667	-6.220988	11.130616	-52.134
186	6.461134	7.137471	4.001882	5.650000	5.283333	-6.264925	10.933205	-51.371
187	6.364891	7.172624	3.927198	5.644737	5.266447	-6.385257	10.855781	-52.640
188	4.447185	3.939471	2.353244	4.666667	4.666667	-0.311111	9.134192	-114.100

189 rows × 8 columns

In [255...

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(122, 7) (67, 7) (122,) (67,)
```

In [256...

```
# Importing the Decision Tree regressor
from sklearn.tree import DecisionTreeRegressor
# Creating and fitting the model
model = DecisionTreeRegressor().fit(X_train, y_train)

# Importing the Linear regressor
from sklearn.linear_model import LinearRegression
# Creating and fitting the model
model1 = LinearRegression().fit(X_train, y_train)

# Importing the Random Forest
from sklearn.ensemble import RandomForestRegressor
# Creating and fitting the model
model2 = RandomForestRegressor().fit(X_train, y_train)
```

In [257...

```
# The coefficient of determination
print('The training r_sq is: %.3f'% model.score(X_train, y_train))
```



```
print(model1.score(X_train, y_train))
```

```
# The coefficient of determination
print(model2.score(X_train, y_train))
```

```
The training r_sq is: 1.000
0.4909606257534359
0.961069089285529
```

In [258…

```
# Training model evaluation
from sklearn.metrics import mean_absolute_error, mean_squared_error, explained_varia
```

In [259…

```
# Prediction on the training dataset
ytrain_pred = model.predict(X_train)

# Prediction on the training dataset
ytrain_pred1 = model1.predict(X_train)

# Prediction on the training dataset
ytrain_pred2 = model2.predict(X_train)
```

In [260…

```
# The r_sq
print('Desicion Tree')
print('The training r_sq is:', r2_score(y_train, ytrain_pred))
# The MAE
print('The training MAE is:', mean_absolute_error(y_train, ytrain_pred))
# The MSE
print('The training MSE is:', mean_squared_error(y_train, ytrain_pred))
# The RMSE
import numpy as np
print('The training RMSE is:', np.sqrt(mean_squared_error(y_train, ytrain_pred)))
# Explained variance score EVS
print('The training EVS is:', explained_variance_score(y_train, ytrain_pred))

# The r_sq
print('Linear Regression')
print('The training r_sq is:', r2_score(y_train, ytrain_pred1))
# The MAE
print('The training MAE is:', mean_absolute_error(y_train, ytrain_pred1))
# The MSE
print('The training MSE is:', mean_squared_error(y_train, ytrain_pred1))
# The RMSE
import numpy as np
print('The training RMSE is:', np.sqrt(mean_squared_error(y_train, ytrain_pred1)))
# Explained variance score EVS
print('The training EVS is:', explained_variance_score(y_train, ytrain_pred1))

# The r_sq
print('Random Forest')
print('The training r_sq is:', r2_score(y_train, ytrain_pred2))
# The MAE
print('The training MAE is:', mean_absolute_error(y_train, ytrain_pred2))
# The MSE
print('The training MSE is:', mean_squared_error(y_train, ytrain_pred2))
# The RMSE
import numpy as np
print('The training RMSE is:', np.sqrt(mean_squared_error(y_train, ytrain_pred2)))
# Explained variance score EVS
print('The training EVS is:', explained_variance_score(y_train, ytrain_pred2))
```



```
The training r_sq is: 0.999976596800058
```

```
The training MAE is: 0.015991803278688644
The training MSE is: 0.005310692622950879
The training RMSE is: 0.07287449912658665
The training EVS is: 0.999976596800058
Linear Regression
The training r_sq is: 0.4909606257534359
The training MAE is: 7.5641991242690105
The training MSE is: 115.51205204014565
The training RMSE is: 10.747653327128933
The training EVS is: 0.4909606257534358
Random Forest
The training r_sq is: 0.961069089285529
The training MAE is: 1.8754850667447331
The training MSE is: 8.834266290454034
The training RMSE is: 2.972249365456074
The training EVS is: 0.9610868473971406
```

In [261…

```
# Prediction on the testing data
ytest_pred = model.predict(X_test)

# Prediction on the testing data
ytest_pred1 = model1.predict(X_test)

# Prediction on the testing data
ytest_pred2 = model2.predict(X_test)
```

In [262…

```
# The r_sq
print('Desicion tree')
print('The testing r_sq is:', r2_score(y_test, ytest_pred))
# The MAE
print('The testing MAE is:', mean_absolute_error(y_test, ytest_pred))
# The MSE
print('The testing MSE is:', mean_squared_error(y_test, ytest_pred))
# The RMSE
import numpy as np
print('The testing RMSE is:', np.sqrt(mean_squared_error(y_test, ytest_pred)))
# Explained variance score EVS
print('The testing EVS is:', explained_variance_score(y_test, ytest_pred))

# The r_sq
print('Linear regression')
print('The testing r_sq is:', r2_score(y_test, ytest_pred1))
# The MAE
print('The testing MAE is:', mean_absolute_error(y_test, ytest_pred1))
# The MSE
print('The testing MSE is:', mean_squared_error(y_test, ytest_pred1))
# The RMSE
import numpy as np
print('The testing RMSE is:', np.sqrt(mean_squared_error(y_test, ytest_pred1)))
# Explained variance score EVS
print('The testing EVS is:', explained_variance_score(y_test, ytest_pred1))

# The r_sq
print('Random Forest')
print('The testing r_sq is:', r2_score(y_test, ytest_pred2))
# The MAE
print('The testing MAE is:', mean_absolute_error(y_test, ytest_pred2))
# The MSE
print('The testing MSE is:', mean_squared_error(y_test, ytest_pred2))
# The RMSE
print('The testing RMSE is:', np.sqrt(mean_squared_error(y_test, ytest_pred2)))
```

```
# Explained variance score EVS
print('The testing EVS is:', explained_variance_score(y_test, ytest_pred2))
```

```
Desicion tree
The testing r_sq is: 0.8045663056170592
The testing MAE is: 4.009365671641789
The testing MSE is: 40.721324436567144
The testing RMSE is: 6.381326228658675
The testing EVS is: 0.8046567667384337
Linear regression
The testing r_sq is: 0.4799163615448353
The testing MAE is: 7.982757549596233
The testing MSE is: 108.36664907017028
The testing RMSE is: 10.409930310533797
The testing EVS is: 0.4853437883305961
Random Forest
The testing r_sq is: 0.869533128098855
The testing MAE is: 3.504310103411503
The testing MSE is: 27.18458470370256
The testing RMSE is: 5.2138838406415
The testing EVS is: 0.8695339652248615
```

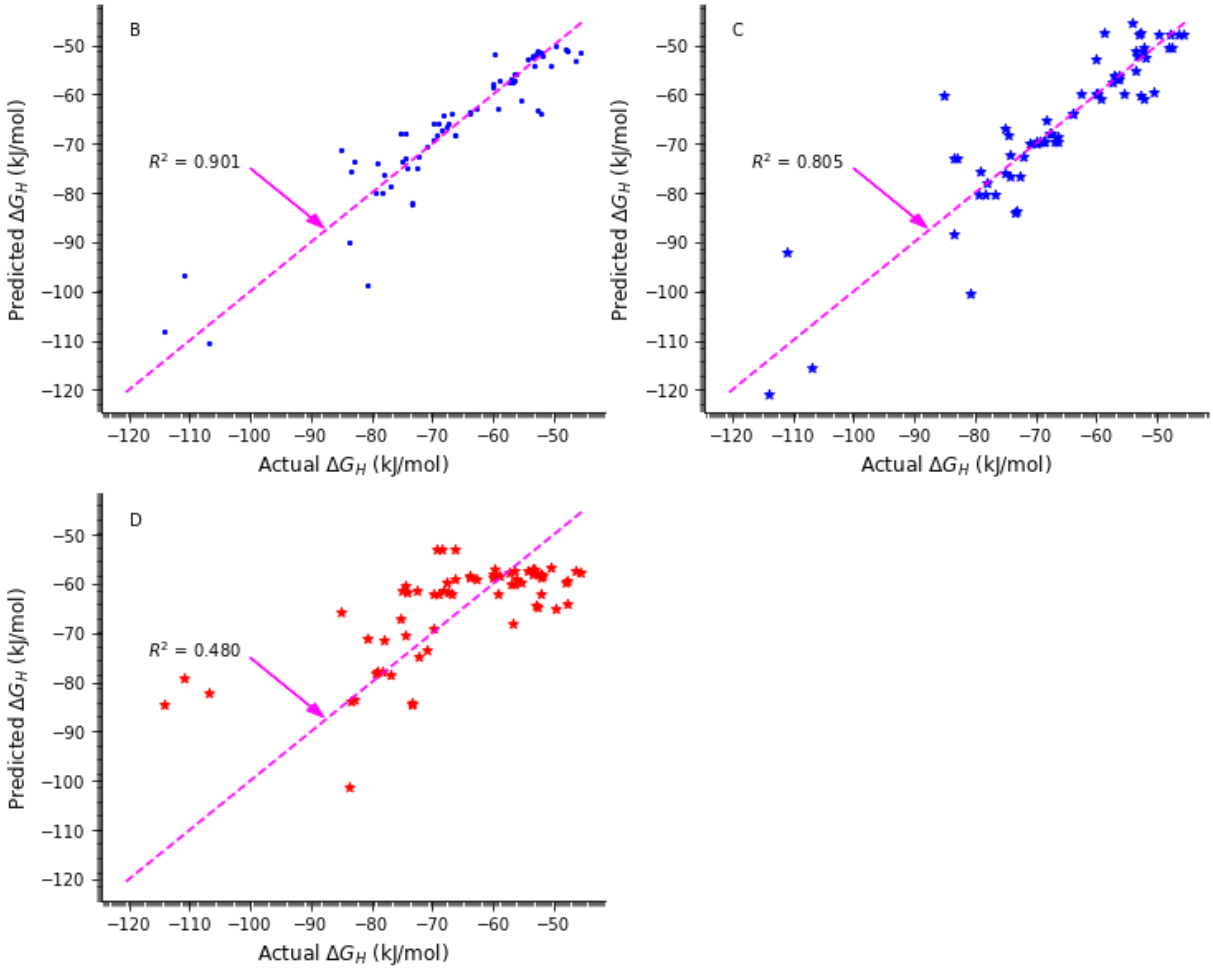
In [294…

```
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
gs = gridspec.GridSpec(2, 2)
from matplotlib.ticker import (MultipleLocator, AutoMinorLocator)

fig = plt.figure(figsize=(12,10))
ax1 = fig.add_subplot(gs[0, 0]) # row 0, col 0
ax1.scatter(y_test, ytest_pred2, label = 'Random Forest', marker='o', s=5, color='b'
p1 = max(max(ytest_pred), max(y_test))
p2 = min(min(ytest_pred), min(y_test))
ax1.plot([p1, p2], [p1, p2], '--', color='magenta')
ax1.text(-120, -48, 'B', size = 10)
ax1.tick_params(which='major', length=3, color='black')
ax1.set_xlabel('Actual $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax1.set_ylabel('Predicted $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax1.tick_params(axis='both', which='major', labelsize=10)
ax1.tick_params(which='major', length=6, color='black')
ax1.tick_params(which='minor', length=3, color='black')
ax1.xaxis.set_minor_locator(MultipleLocator(0.3))
ax1.yaxis.set_minor_locator(MultipleLocator(0.3))
ax1.arrow(-100, -75, 10, -10,
          head_width = 2,
          width = 0.1,
          color ='magenta')
ax1.text(-117, -75, '$R^{2}$ = 0.901', size = 10)

ax2 = fig.add_subplot(gs[0, 1]) # row 0, col 1
ax2.scatter(y_test, ytest_pred, label = 'Decision Tree', marker='*', c='b')
p1 = max(max(ytest_pred), max(y_test))
p2 = min(min(ytest_pred), min(y_test))
ax2.plot([p1, p2], [p1, p2], '--', color='magenta')
ax2.text(-120, -48, 'C', size = 10)
ax2.tick_params(which='major', length=3, color='black')
ax2.set_xlabel('Actual $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax2.set_ylabel('Predicted $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax2.tick_params(axis='both', which='major', labelsize=10)
ax2.tick_params(which='major', length=6, color='black')
ax2.tick_params(which='minor', length=3, color='black')
ax2.xaxis.set_minor_locator(MultipleLocator(0.3))
ax2.yaxis.set_minor_locator(MultipleLocator(0.3))
```



```
          head_width = 2,
```

```
          width = 0.1,
          color ='magenta')
ax2.text(-117, -75, '$R^{2}$ = 0.805', size = 10)

ax3 = fig.add_subplot(gs[1, 0]) # row 1, span all columns
ax3.scatter(y_test, ytest_pred1, label = 'Linear Regression', marker='*', color='r')
p1 = max(max(ytest_pred), max(y_test))
p2 = min(min(ytest_pred), min(y_test))
ax3.plot([p1, p2], [p1, p2], '--', color='magenta')
ax3.text(-120, -48, 'D', size = 10)
ax3.tick_params(which='major', length=3, color='black')
ax3.set_xlabel('Actual $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax3.set_ylabel('Predicted $\Delta$$G_{H}$ (kJ/mol)', fontsize=12)
ax3.tick_params(axis='both', which='major', labelsize=10)
ax3.tick_params(which='major', length=6, color='black')
ax3.tick_params(which='minor', length=3, color='black')
ax3.xaxis.set_minor_locator(MultipleLocator(0.3))
ax3.yaxis.set_minor_locator(MultipleLocator(0.3))
ax3.arrow(-100, -75, 10, -10,
          head_width = 2,
          width = 0.1,
          color ='magenta')
ax3.text(-117, -75, '$R^{2}$ = 0.480', size = 10)

plt.savefig('Separated DeltaGh0.35_scatter.png')
```



In [295…

```
# Saving and loading the trained model
import joblib
joblib.dump(model, 'DeltaHcal-pred-Dicision.joblib')
```

In [296…

```
import joblib
joblib.dump(model1, 'DeltaHcal-pred-Linear.joblib')
```

Out[296…

```
['DeltaHcal-pred-Linear.joblib']
```

In [297…

```
import joblib
joblib.dump(model2, 'DeltaHcal-pred-Random.joblib')
```

Out[297…

```
['DeltaHcal-pred-Random.joblib']
```

In [332…

```
model2 = joblib.load('DeltaHcal-pred-Random.joblib')
predictions = model2.predict([[8.52, 9.77, 10.03, 6.23, 5.33, -15.29, 12.93]])
predictions
```

Out[332…

```
array([-59.45296])
```

In [ ]:

In [744…

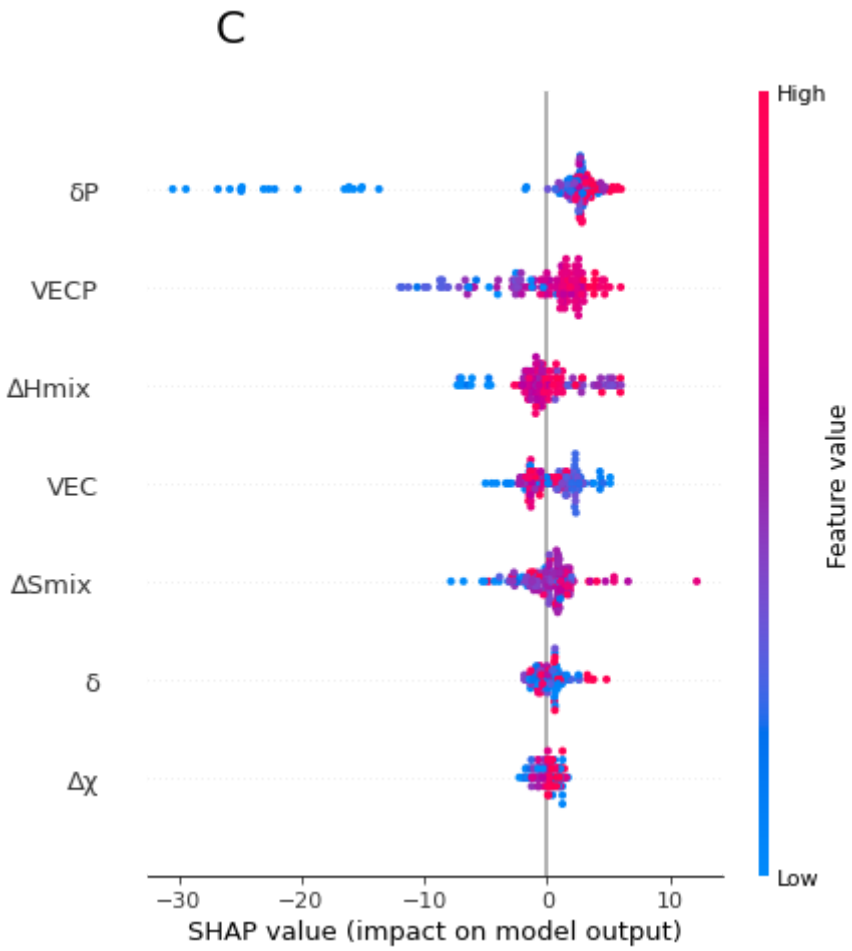
```
# Feature importance
import pandas as pd
feature_importances = pd.DataFrame(model2.feature_importances_,
                                   index = X_train.columns,
                                   columns=['importance']).sort_values('importa
feature_importances

!pip install shap
import shap
explainer = shap.TreeExplainer(model2)
shap_values = explainer.shap_values(X_train)
fig = shap.summary_plot(shap_values, X_train, show=False)

print(f'Original size: {plt.gcf().get_size_inches()}')
w, _ = plt.gcf().get_size_inches()
plt.gcf().set_size_inches(6, 6)
plt.tight_layout()
print(f'New size: {plt.gcf().get_size_inches()}')

import matplotlib.pyplot as pl
f = pl.gcf()
plt.text(-27, 7.5, 'C', fontsize = 22)
plt.savefig('feature importance.png', bbox_inches='tight',dpi=100)
```

```
Requirement already satisfied: shap in c:\programdata\anaconda3\lib\site-packages
(0.41.0)
Requirement already satisfied: slicer==0.0.7 in c:\programdata\anaconda3\lib\site-pa
ckages (from shap) (0.0.7)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages
(from shap) (1.20.3)
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-pac
kages (from shap) (0.24.2)
Requirement already satisfied: numba in c:\programdata\anaconda3\lib\site-packages
(from shap) (0.54.1)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages
(from shap) (1.7.1)
Requirement already satisfied: packaging>20.9 in c:\programdata\anaconda3\lib\site-p
```



```
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages
```

```
(from shap) (1.3.4)
Requirement already satisfied: cloudpickle in c:\programdata\anaconda3\lib\site-pack
ages (from shap) (2.0.0)
Requirement already satisfied: tqdm>4.25.0 in c:\programdata\anaconda3\lib\site-pack
ages (from shap) (4.62.3)
Requirement already satisfied: pyparsing>=2.0.2 in c:\programdata\anaconda3\lib\site
-packages (from packaging>20.9->shap) (3.0.4)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-package
s (from tqdm>4.25.0->shap) (0.4.4)
Requirement already satisfied: llvmlite<0.38,>=0.37.0rc1 in c:\programdata\anaconda3
\lib\site-packages (from numba->shap) (0.37.0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packa
ges (from numba->shap) (58.0.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\li
b\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\programdata\anaconda3\lib\site-pac
kages (from pandas->shap) (2021.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-package
s (from python-dateutil>=2.7.3->pandas->shap) (1.16.0)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-pac
kages (from scikit-learn->shap) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib
\site-packages (from scikit-learn->shap) (2.2.0)
Original size: [8.  4.3]
New size: [6. 6.]
```



In [ ]: