Imports

In [1]:
```
import numpy as np
import pandas as pd

from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
!ls
```

```
ML.ipynb     metadata.csv results.csv
```

Load datasets

In [3]:
```
df_extracted = pd.read_csv("results.csv")
df_extracted.head()
```

Out[3]:

	filename	R	G	B	BC_TOT
0	RCV1.2.4_P1_12h_d.jpg	219.772414	215.270069	209.003862	to_be_calculated
1	RCV1.2.4_P1_24h_d.jpg	189.561748	180.045175	167.519507	to_be_calculated
2	RCV1.2.4_P1_36h_d.jpg	201.165241	193.114483	180.889103	to_be_calculated
3	RCV1.2.4_P1_6h_L.jpg	194.706072	191.158404	181.623057	to_be_calculated
4	RCV1.2.4_P1_36h_C.jpg	181.702069	172.303724	152.748966	to_be_calculated

In [4]:
```
df_metadata = pd.read_csv("metadata.csv")
df_metadata.sample(5)
```

Out[4]:

	filename	BC	ring_light
11	RCV1.2.4_P1_48h_L.jpg	2.35	False
9	RCV1.2.4_P1_60h_B.jpg	5.68	False
22	RCV1.2.4_P1_12h_C.jpg	0.50	False
10	RCV1.2.4_P1_48h_C.jpg	2.35	False
3	RCV1.2.4_P1_72h_d.jpg	1.73	True

In [5]:
```
df_combined = pd.merge(df_extracted, df_metadata, on=["filename"])
df_combined.head()
```

Out[5]:

	filename	R	G	B	BC_TOT	BC	ring_light
0	RCV1.2.4_P1_12h_d.jpg	219.772414	215.270069	209.003862	to_be_calculated	0.50	True
1	RCV1.2.4_P1_24h_d.jpg	189.561748	180.045175	167.519507	to_be_calculated	2.09	True
2	RCV1.2.4_P1_36h_d.jpg	201.165241	193.114483	180.889103	to_be_calculated	1.50	True
3	RCV1.2.4_P1_6h_L.jpg	194.706072	191.158404	181.623057	to_be_calculated	0.16	False
4	RCV1.2.4_P1_36h_C.jpg	181.702069	172.303724	152.748966	to_be_calculated	1.50	False

In [6]:
```
df_ring = df_combined.query("ring_light == True")
df_ring
```

Out[6]:

	filename	R	G	B	BC_TOT	BC	ring_light
0	RCV1.2.4_P1_12h_d.jpg	219.772414	215.270069	209.003862	to_be_calculated	0.50	True
1	RCV1.2.4_P1_24h_d.jpg	189.561748	180.045175	167.519507	to_be_calculated	2.09	True
2	RCV1.2.4_P1_36h_d.jpg	201.165241	193.114483	180.889103	to_be_calculated	1.50	True
19	RCV1.2.4_P1_48h_d.jpg	175.695172	164.277793	149.140966	to_be_calculated	2.35	True
20	RCV1.2.4_P1_72h_d.jpg	197.425678	188.686507	176.555313	to_be_calculated	1.73	True
21	RCV1.2.4_P1_60h_d.jpg	145.976533	131.761220	114.446465	to_be_calculated	5.68	True
25	RCV1.2.4_P1_6h_d.jpg	226.354004	223.266127	216.743222	to_be_calculated	0.16	True

Merge datasets

In [7]:
```
df_no_ring = df_combined.query("ring_light == False")
df_no_ring.head()
```

Out[7]:

	filename	R	G	B	BC_TOT	BC	ring_light
3	RCV1.2.4_P1_6h_L.jpg	194.706072	191.158404	181.623057	to_be_calculated	0.16	False
4	RCV1.2.4_P1_36h_C.jpg	181.702069	172.303724	152.748966	to_be_calculated	1.50	False
5	RCV1.2.4_P1_24h_C.jpg	165.438345	156.818207	137.574345	to_be_calculated	2.09	False
6	RCV1.2.4_P1_12h_B.jpg	192.197793	187.399724	174.828414	to_be_calculated	0.50	False
7	RCV1.2.4_P1_12h_C.jpg	192.154565	187.761296	176.264568	to_be_calculated	0.50	False

In [8]:
```
df_no_ring.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 3 to 27
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   filename    21 non-null     object 
 1   R           21 non-null     float64
 2   G           21 non-null     float64
 3   B           21 non-null     float64
 4   BC_TOT      21 non-null     object 
 5   BC          21 non-null     float64
 6   ring_light  21 non-null     bool   
dtypes: bool(1), float64(4), object(2)
memory usage: 1.2+ KB
```

Extract R, G, B values as X, and BC values as y

In [9]:
```
X, y = df_no_ring[["R", "G", "B"]], df_no_ring["BC"]

X_ring, y_ring = df_ring[["R", "G", "B"]], df_ring["BC"]
```

ML with Scikit-Learn

In [10]:
```
from sklearn.preprocessing import Normalizer, StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score, cross_validate, GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score # true, pred
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
X_ring_train, X_ring_test, y_ring_train, y_ring_test = train_test_split(X_ring, y_ring, random_state=33)

print("Dataset - No Ring Light")
print("Train size: ", len(X_train), "\nTest size: ", len(X_test))

print("\nDataset - Ring Light")
print("Train size: ", len(X_ring_train), "\nTest size: ", len(X_ring_test))
```

```
Dataset - No Ring Light
Train size:  15
Test size:  6

Dataset - Ring Light
Train size:  5
Test size:  2
```

Create Ensemble Model

In [12]:
```
from sklearn.linear_model import LinearRegression, RidgeCV, Lasso, ElasticNetCV
from sklearn.svm import SVR
from sklearn.ensemble import VotingRegressor, RandomForestRegressor, GradientBoostingRegressor


gb = GradientBoostingRegressor(random_state=1, n_estimators=100, learning_rate=0.1, max_depth=1)
rf = RandomForestRegressor(random_state=1, n_estimators=10, max_features=1, max_leaf_nodes=5)
lr = LinearRegression()
ridge = RidgeCV()
enet = ElasticNetCV()
svr = SVR()

regressors = VotingRegressor(estimators=[\
    ('gb', gb), \
    ('rf', rf), \
    ('lr', lr), \
    ('ridge', ridge), \
    ('enet', enet), \
    ('svm', svr)\
  ])

model = make_pipeline(StandardScaler(), regressors)
# model = make_pipeline(MinMaxScaler(), regressors)
# model = make_pipeline(Normalizer(), regressors)

cross_validate(model, X_train, y_train, cv=4, scoring=("r2", "neg_mean_absolute_error", "neg_mean_squared_err
or", "neg_root_mean_squared_error"))
```

Out[12]:
```
{'fit_time': array([0.08719516, 0.07031035, 0.0730288 , 0.09804273]),
 'score_time': array([0.00470686, 0.00363183, 0.00497508, 0.00574422]),
 'test_r2': array([0.97771192, 0.93170525, 0.95794095, 0.95457944]),
 'test_neg_mean_absolute_error': array([-0.0960905 , -0.29734085, -0.10989247, -0.35576574]),
 'test_neg_mean_squared_error': array([-0.01651032, -0.19346026, -0.01865109, -0.23761714]),
 'test_neg_root_mean_squared_error': array([-0.12849248, -0.43984117, -0.13656898, -0.48745989])}
```

Check R2 score after cross-validation

In [13]:
```
cross_val_score(model, X_train, y_train, cv=4)
```

Out[13]:
```
array([0.97771192, 0.93170525, 0.95794095, 0.95457944])
```

Fit Model

In [14]:
```
model = model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mae, mse, r2 = mean_absolute_error(y_test, y_pred), mean_squared_error(y_test, y_pred), r2_score(y_test, y_pr
ed)

print("MAE: ", mae, "\nMSE: ", mse, "\n R2: ", r2, "\nPredictions: ", y_pred)
```

```
MAE:  0.23632322106513956
MSE:  0.07330189566831313
 R2:  0.9771605854239414
Predictions:  [2.22034775 1.7792501  5.23886048 0.143234   1.89732555 0.40776151]
```

Test Model on ring-light data

In [15]:
```
y_ring_pred = model.predict(X_ring_test)
mae, mse, r2 = mean_absolute_error(y_ring_test, y_ring_pred), mean_squared_error(y_ring_test, y_ring_pred), r
2_score(y_ring_test, y_ring_pred)

print("MAE: ", mae, "\nMSE: ", mse, "\n R2: ", r2, "\nPredictions: ", y_ring_pred)
```

```
MAE:  1.6285515586080108
MSE:  3.482857358930799
 R2:  -0.25634007302105677
Predictions:  [3.14003351 1.63286337]
```

Performance is pretty terrible

In [ ]:

Dataset is too small for images using ring light

But we fit a model anyways

In [16]:
```
reg1 = GradientBoostingRegressor(random_state=1, n_estimators=100, learning_rate=0.1, max_depth=1)
reg2 = RandomForestRegressor(random_state=1, n_estimators=10, max_features=1, max_leaf_nodes=2)
reg3 = LinearRegression()
reg4 = SVR()

regressors = VotingRegressor(estimators=[\
    ('gb', reg1), \
    ('rf', reg2), \
    ('lr', reg3), \
    ('svm', reg4)\
  ])

model = make_pipeline(StandardScaler(), regressors)
model = model.fit(X_ring_train, y_ring_train)
cross_val_score(model, X_ring_train, y_ring_train, cv=2)
```

Out[16]:
```
array([0.82446025, 0.98283886])
```

In [17]:
```
model = model.fit(X_ring_train, y_ring_train)

y_ring_pred = model.predict(X_ring_test)
mae, mse, r2 = mean_absolute_error(y_ring_test, y_ring_pred), mean_squared_error(y_ring_test, y_ring_pred), r
2_score(y_ring_test, y_ring_pred)

print("MAE: ", mae, "\nMSE: ", mse, "\n R2: ", r2, "\nPredictions: ", y_ring_pred)
```

```
MAE:  1.8657399964733092
MSE:  5.815473008348217
 R2:  -1.0977637126669797
Predictions:  [2.2863571 2.0121629]
```

Use Model trained on Ring light data to predict BC for older images

In [18]:
```
y_pred = model.predict(X_test)
mae, mse, r2 = mean_absolute_error(y_test, y_pred), mean_squared_error(y_test, y_pred), r2_score(y_test, y_pr
ed)

print("MAE: ", mae, "\nMSE: ", mse, "\n R2: ", r2, "\nPredictions: ", y_pred)
```

```
MAE:  1.0135603620607447
MSE:  2.194693028245988
 R2:  0.3161772491381297
Predictions:  [2.03354878 1.99529221 2.56016037 1.60982121 2.01751777 1.61747568]
```

Hyperparameter tuning

In [19]:
```
# params = {"param": np.arange(1,3), "kernel": ["linear", "rbf"]}

# grid = GridSearchCV(estimator=model, param_grid=params)
# grid.fit(X_train, y_train)

# print(grid.best_score_)
# print(grid.best_estimator_)
```

In [ ]:
```

```