##Importing Libraries

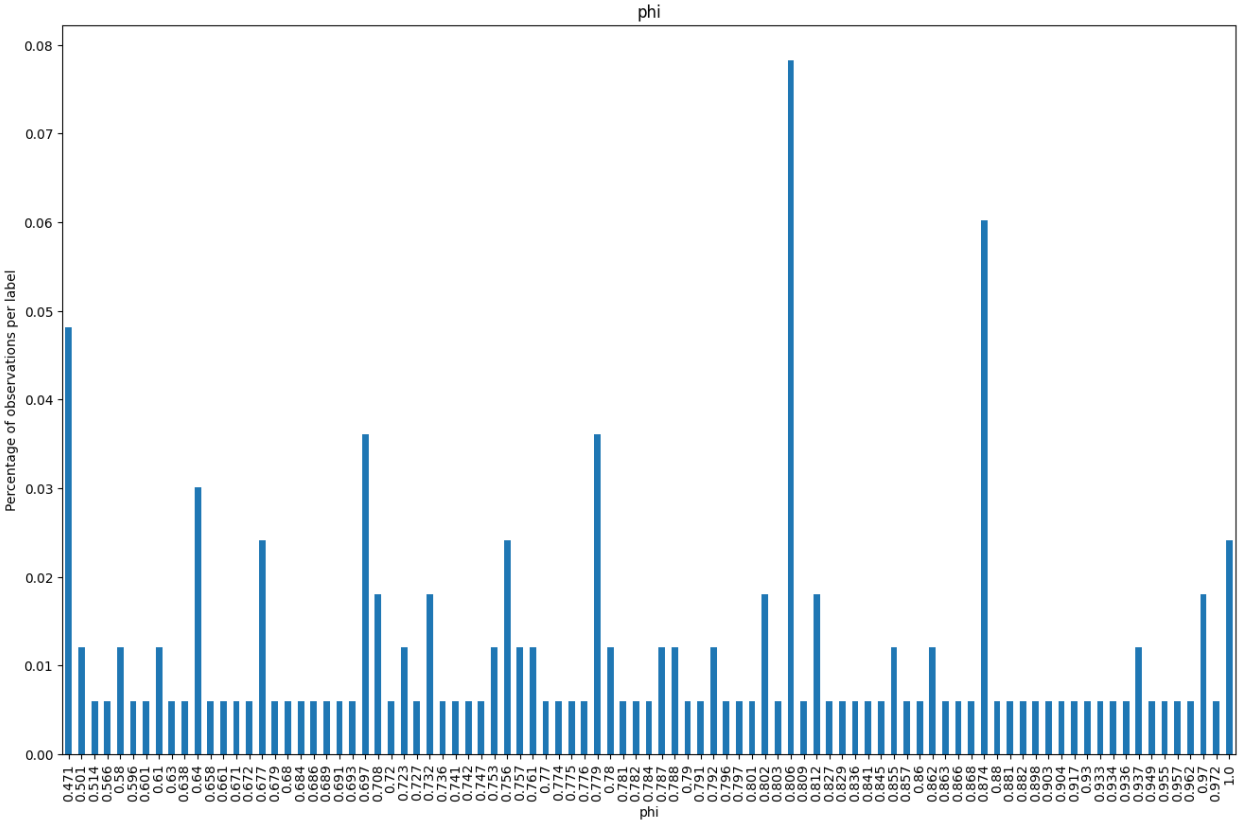
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

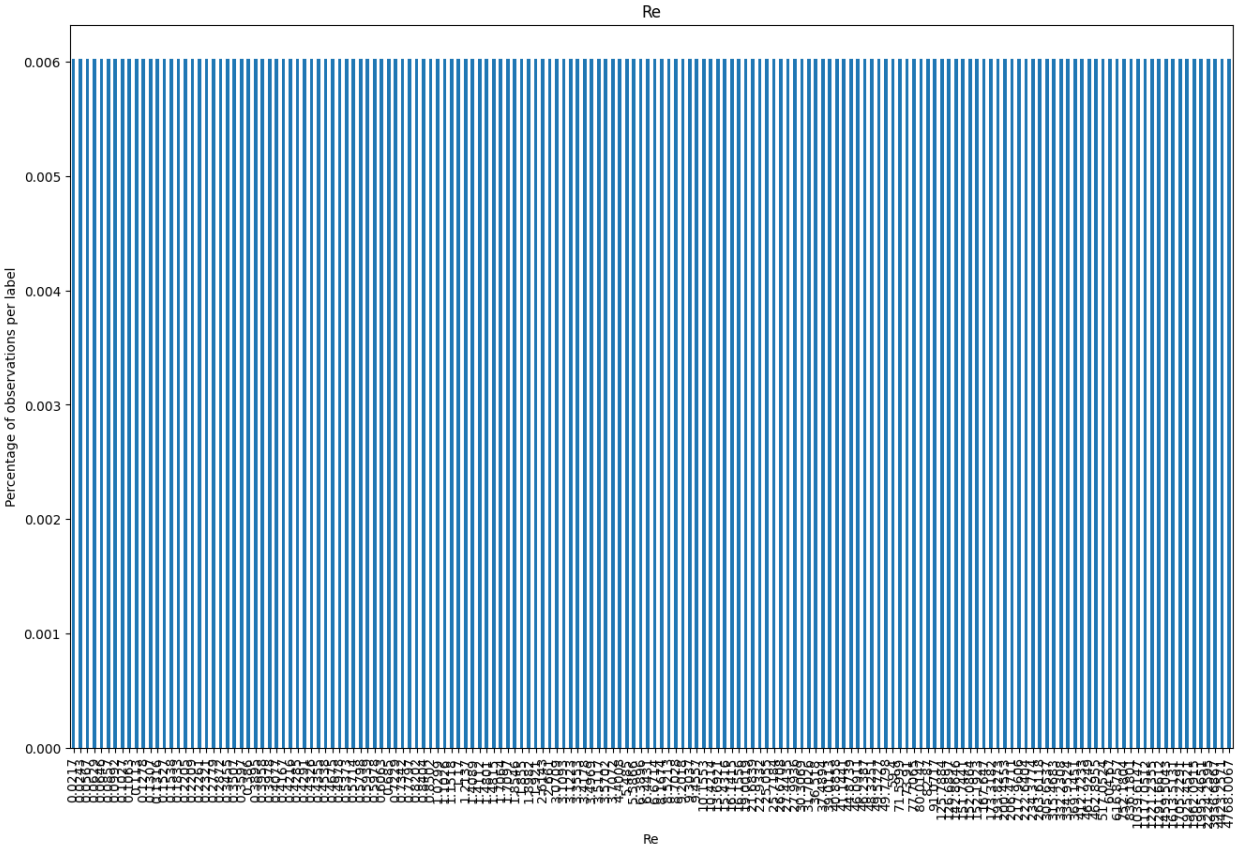
##Importing the Dataset

```
df = pd.read_csv('drag_coef.csv')

X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
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 = 0)
```

##Detecting Outliers by Discrete Variables on Testset

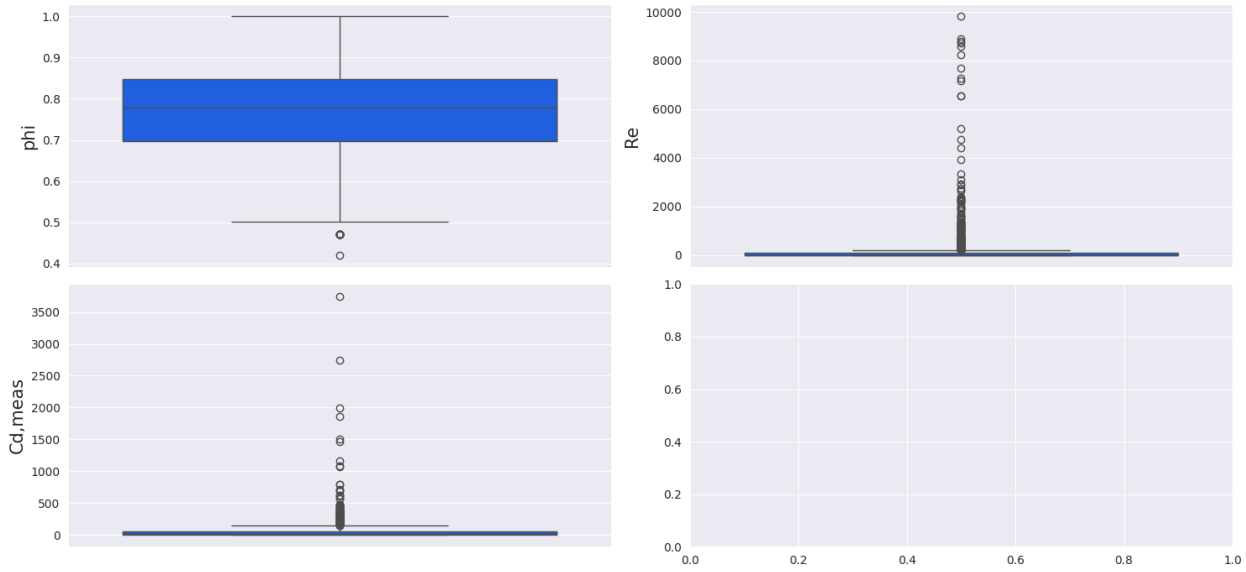
```
X_test_df = pd.DataFrame(X_test, columns=['phi', 'Re'])

for var in ['phi', 'Re']:
    plt.figure(figsize=(16,10))
    (X_test_df.groupby(var)[var].count() /
float(len(X_test_df))).plot.bar()
    plt.ylabel('Percentage of observations per label')
    plt.title(var)
    plt.show()
```


phi
Percentage of observations per label
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
phi


Re
Percentage of observations per label
0.000
0.001
0.002
0.003
0.004
0.005
0.006
Re

##Detecting the Outliers by Interquartile Range Method

```
sns.set_style('darkgrid')
colors = ['#0055ff', '#ff7000', '#23bf00']
CustomPalette = sns.set_palette(sns.color_palette(colors))

OrderedCols =
np.concatenate([df.select_dtypes(exclude='object').columns.values,

df.select_dtypes(include='object').columns.values])

fig, ax = plt.subplots(2, 2, figsize=(15,7),dpi=100)

for i,col in enumerate(OrderedCols):
    x = i//2
    y = i%2
    if i<5:
        sns.boxplot(data=df, y=col, ax=ax[x,y])
        ax[x,y].yaxis.label.set_size(15)
    else:
        sns.boxplot(data=df, x=col, y='C_d_exp', ax=ax[x,y])
        ax[x,y].xaxis.label.set_size(15)
        ax[x,y].yaxis.label.set_size(15)

plt.tight_layout()
plt.show()
```

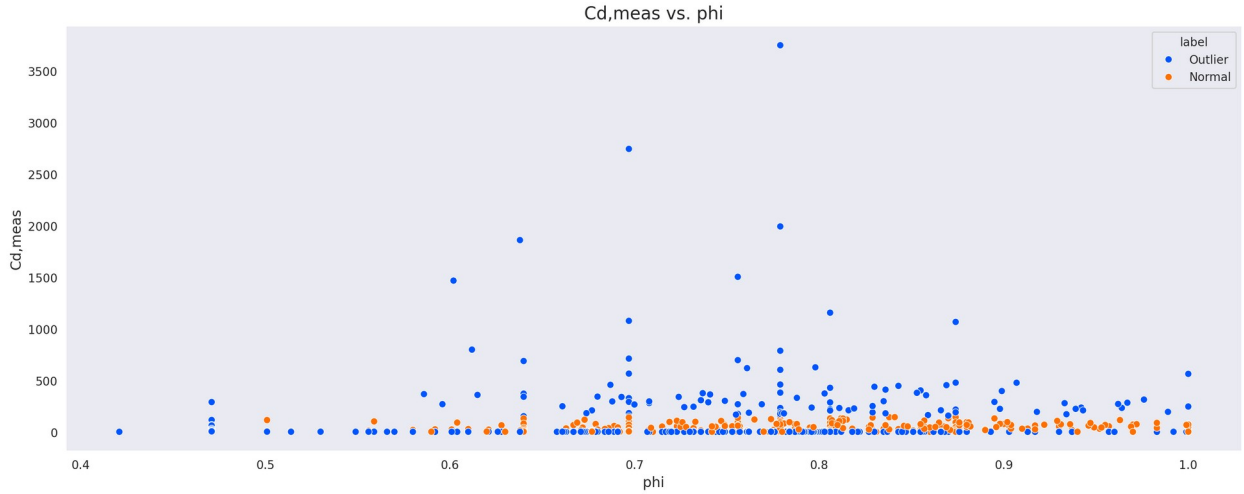


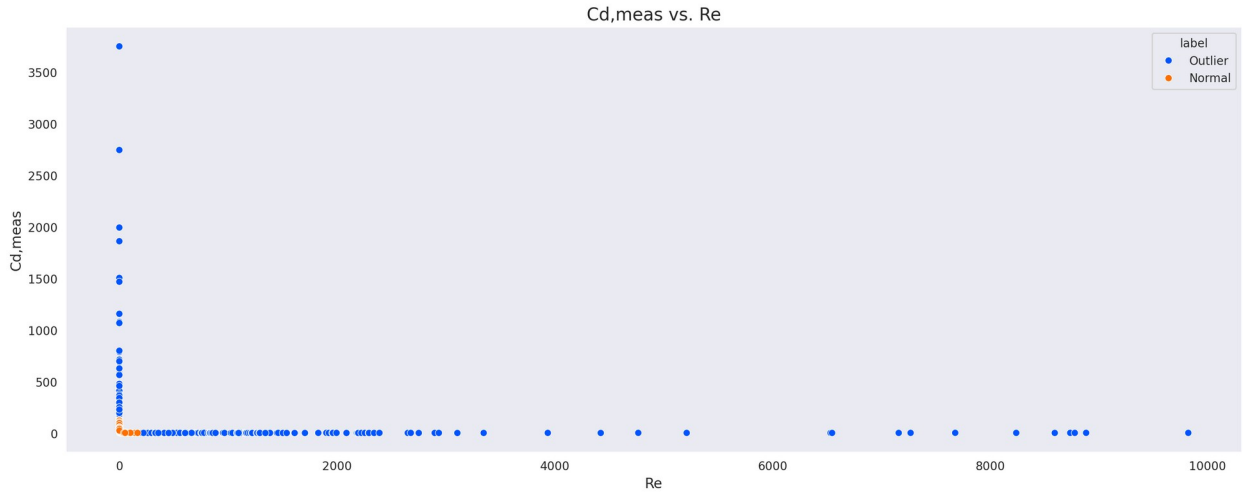

```
outliers_indexes = []
target = 'C_d_exp'

for col in df.select_dtypes(include='object').columns:
    for cat in df[col].unique():
```

```
        df1 = df[df[col] == cat]
        q1 = df1[target].quantile(0.25)
        q3 = df1[target].quantile(0.75)
        iqr = q3-q1
        maximum = q3 + (1.5 * iqr)
        minimum = q1 - (1.5 * iqr)
        outlier_samples = df1[(df1[target] < minimum) | (df1[target] >
maximum)]
        outliers_indexes.extend(outlier_samples.index.tolist())


for col in df.select_dtypes(exclude='object').columns:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3-q1
    maximum = q3 + (1.5 * iqr)
    minimum = q1 - (1.5 * iqr)
    outlier_samples = df[(df[col] < minimum) | (df[col] > maximum)]
    outliers_indexes.extend(outlier_samples.index.tolist())

outliers_indexes = list(set(outliers_indexes))
print('{} outliers were identified, whose indices are:\n\
n{}'.format(len(outliers_indexes), outliers_indexes))
```

```
296 outliers were identified, whose indices are:

[512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525,
526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539,
540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553,
554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567,
568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581,
582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595,
596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609,
610, 611, 612, 613, 614, 615, 96, 101, 106, 111, 627, 628, 629, 630,
631, 633, 634, 635, 636, 637, 638, 639, 126, 130, 5, 134, 651, 652,
653, 654, 655, 657, 658, 659, 660, 661, 662, 663, 0, 162, 163, 677,
678, 167, 679, 682, 683, 684, 686, 176, 688, 689, 690, 691, 692, 693,
694, 706, 708, 710, 713, 714, 204, 209, 214, 219, 736, 737, 45, 739,
46, 744, 50, 763, 49, 249, 250, 251, 252, 253, 254, 255, 256, 257,
258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271,
272, 273, 274, 275, 53, 276, 277, 278, 279, 280, 281, 283, 284, 285,
286, 10, 57, 287, 288, 289, 769, 770, 797, 798, 771, 800, 772, 817,
773, 819, 61, 821, 774, 825, 826, 827, 775, 776, 777, 65, 778, 779,
780, 781, 782, 767, 783, 784, 785, 786, 787, 15, 768, 788, 789, 790,
791, 792, 793, 81, 794, 82, 795, 83, 796, 86, 799, 87, 91, 481, 482,
483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496,
497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510,
511]
```

```
# Outliers Labeling
df1 = df.copy()
df1['label'] = 'Normal'
df1.loc[outliers_indexes,'label'] = 'Outlier'

# Plot
target = 'Cd,meas'
features = df.columns.drop(target)
colors = ['#0055ff','#ff7000','#23bf00']
CustomPalette = sns.set_palette(sns.color_palette(colors))
fig, ax = plt.subplots(nrows=len(features) ,ncols=1, figsize=(15,12),
dpi=200)

for i in range(len(features)):
    sns.scatterplot(data=df1, x=features[i], y=target, hue='label',
ax=ax[i]) # Modified this line
    ax[i].set_title('{} vs. {}'.format(target, features[i]), size =
15)
    ax[i].set_xlabel(features[i], size = 12)
    ax[i].set_ylabel(target, size = 12)
    ax[i].grid()

plt.tight_layout()
plt.show()
```


Cd,meas vs. phi
label
Outlier
Normal
Cd,meas
3500
3000
2500
2000
1500
1000
500
0
0.4
0.5
0.6
0.7
0.8
0.9
1.0
phi


Cd,meas vs. Re
label
Outlier
Normal
Cd,meas
3500
3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
Re

correlation

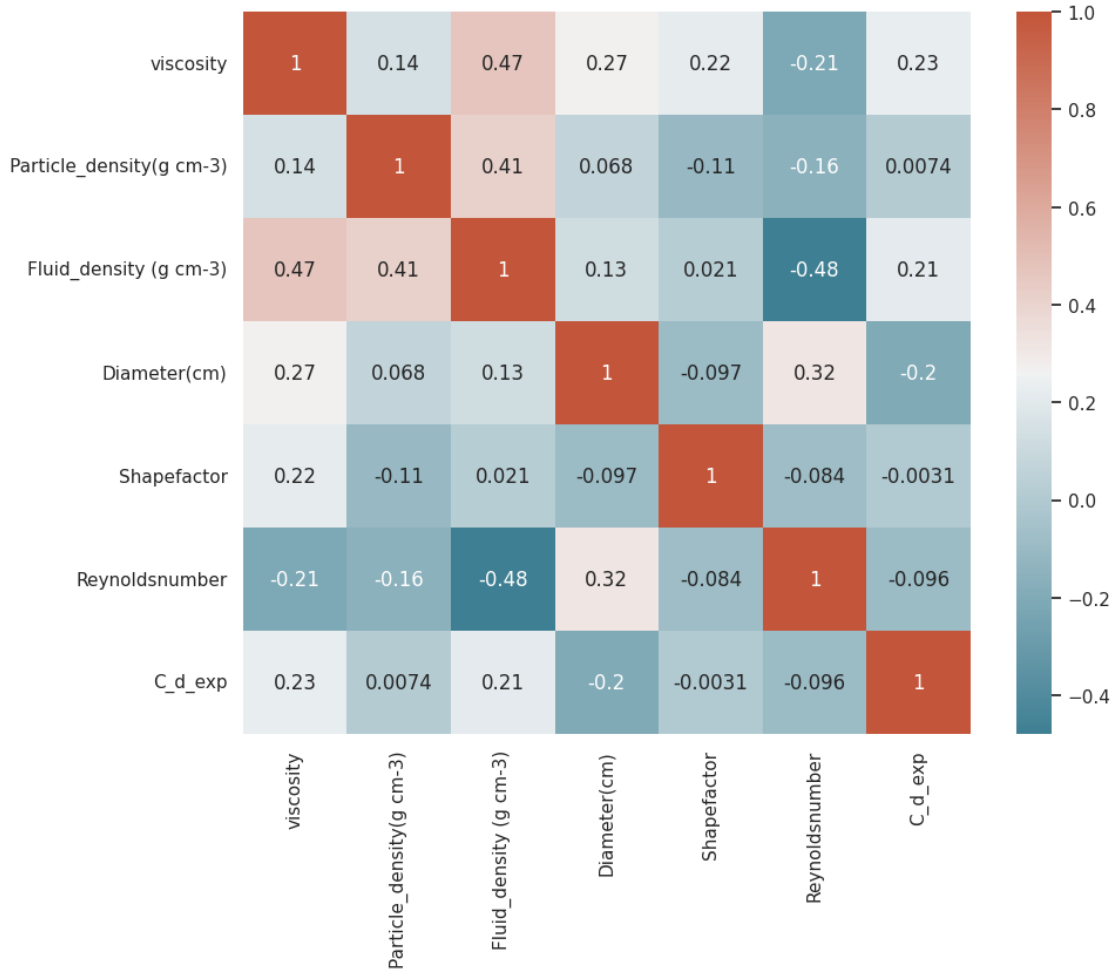
July 15, 2024

[1]:
```
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

[2]:
```
data = pd.read_csv('Dragdata.csv')
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

[3]:
```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3,␣
 ↪random_state=0,)
```

[4]:
```
# the default correlation method of pandas.corr is pearson
corrmat = data.corr(method='pearson')

# To modify image size
fig, ax = plt.subplots()
fig.set_size_inches(10, 8)

# customized color map
cmap = sns.diverging_palette(220, 20, as_cmap=True)

# we can make a heatmap with seaborn
sns.heatmap(corrmat, cmap=cmap, annot=True)
plt.show()
```



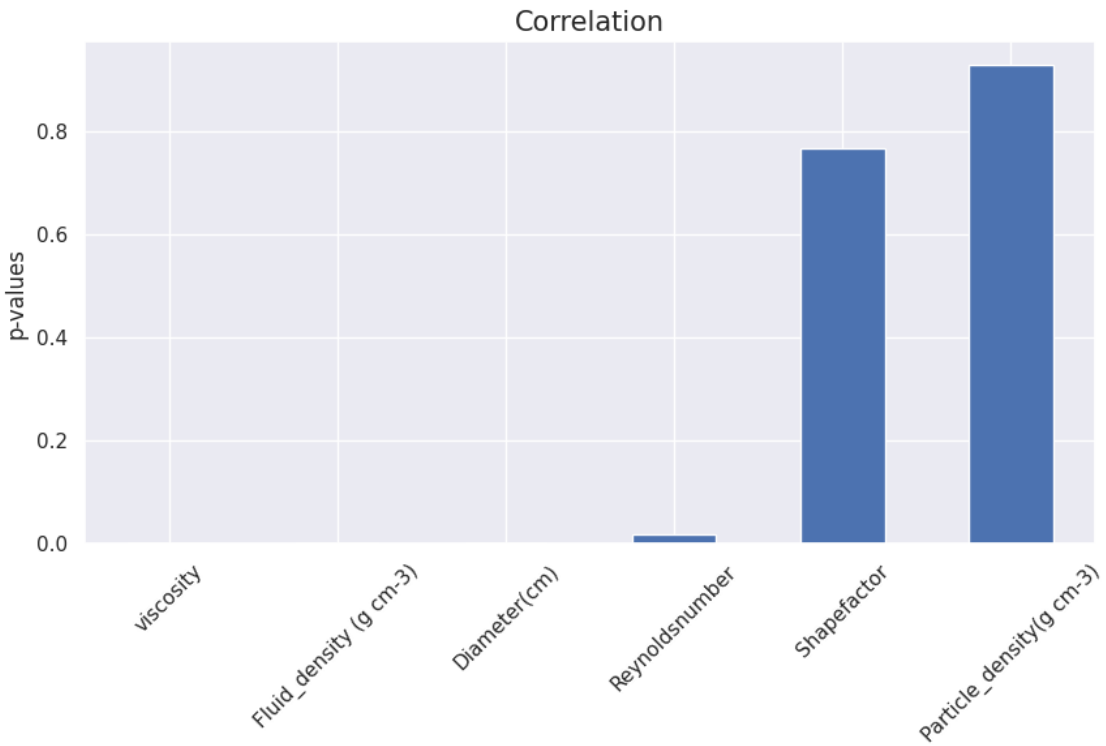
[5]:
```
from sklearn.feature_selection import (f_regression, SelectPercentile,)
```

[6]:
```
univariate = f_regression(X_train, y_train)

# the output is one array with f-scores
# and one array with the pvalues

univariate
```

[6]:
```
(array([3.56790514e+01, 8.06067800e-03, 2.69851622e+01, 2.26045373e+01,
        8.74506189e-02, 5.64658556e+00]),
 array([4.07082769e-09, 9.28492181e-01, 2.84760964e-07, 2.51650738e-06,
        7.67549925e-01, 1.78147777e-02]))
```

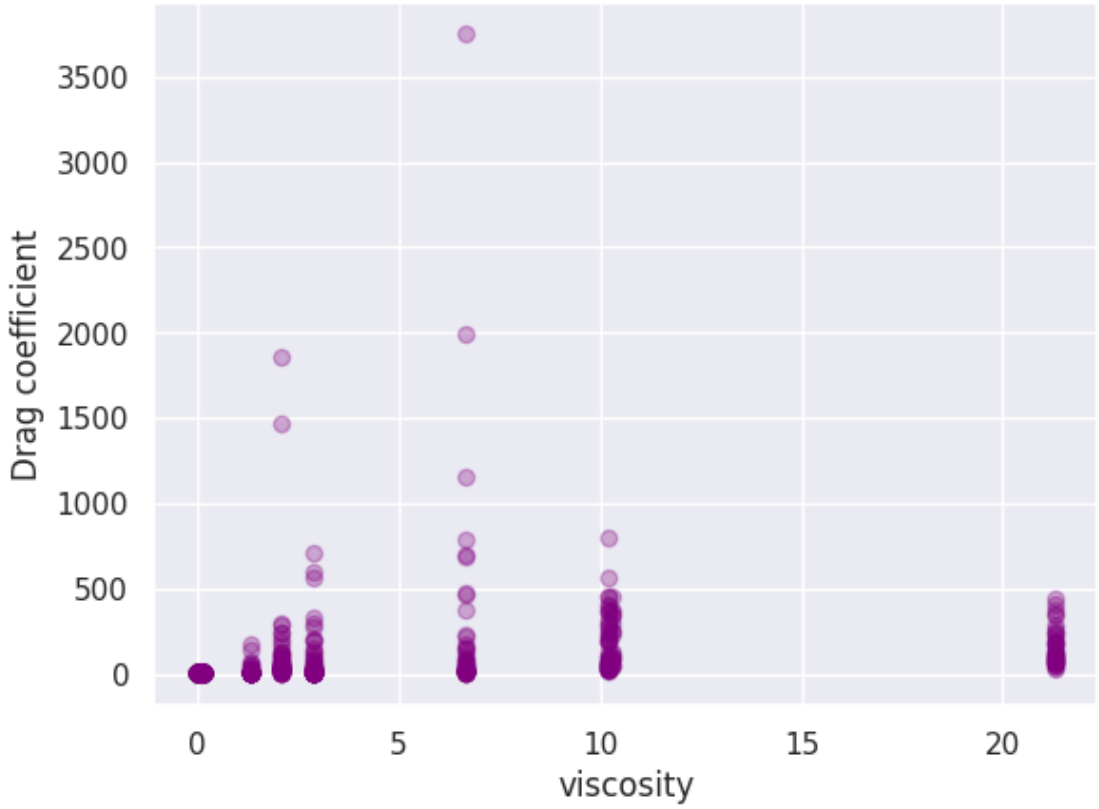
[7]:
```
# 1)capture the pvalues in a pandas series
# 2)add the variable names in the index
# 3)sort the features based on their anova pvalues
```

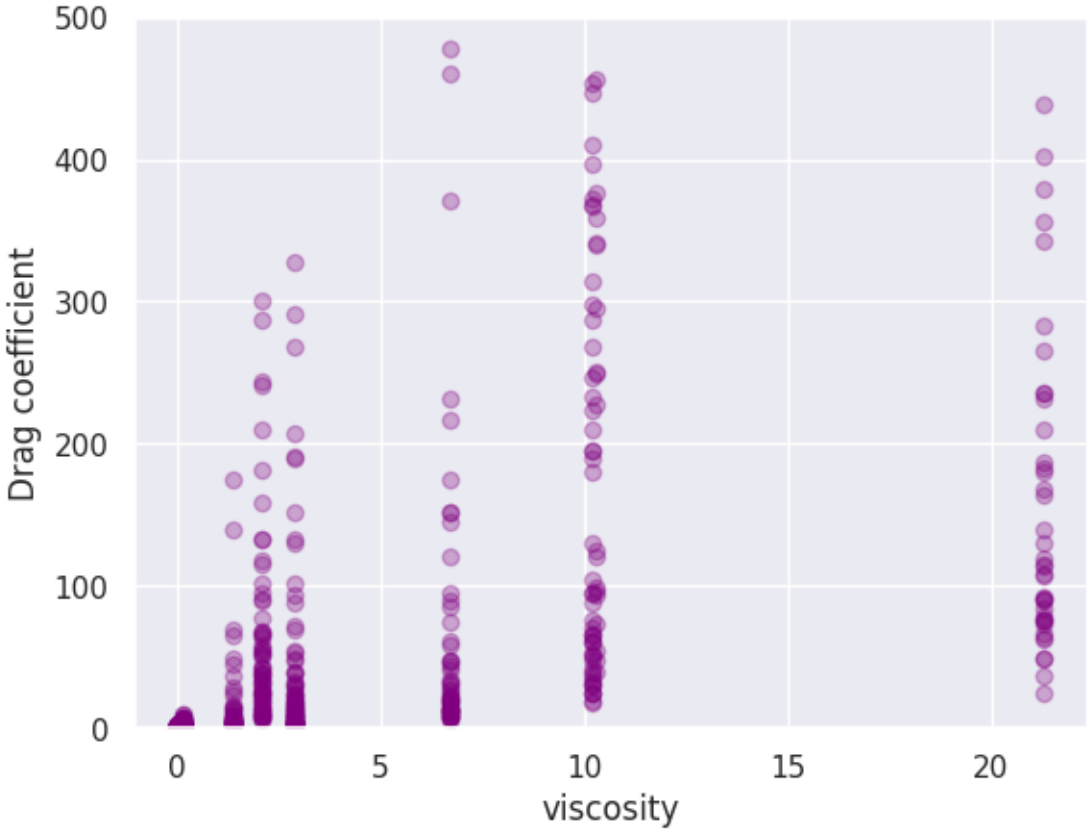
```
# 4)and make a var plot

plt.rc('axes', titlesize=15) #fontsize of the title
univariate = pd.Series(univariate[1])
univariate.index = X_train.columns
univariate.sort_values(ascending=True).plot.bar(figsize=(10, 5), rot=45)
plt.ylabel('p-values')
plt.title('Correlation')
plt.show()
```

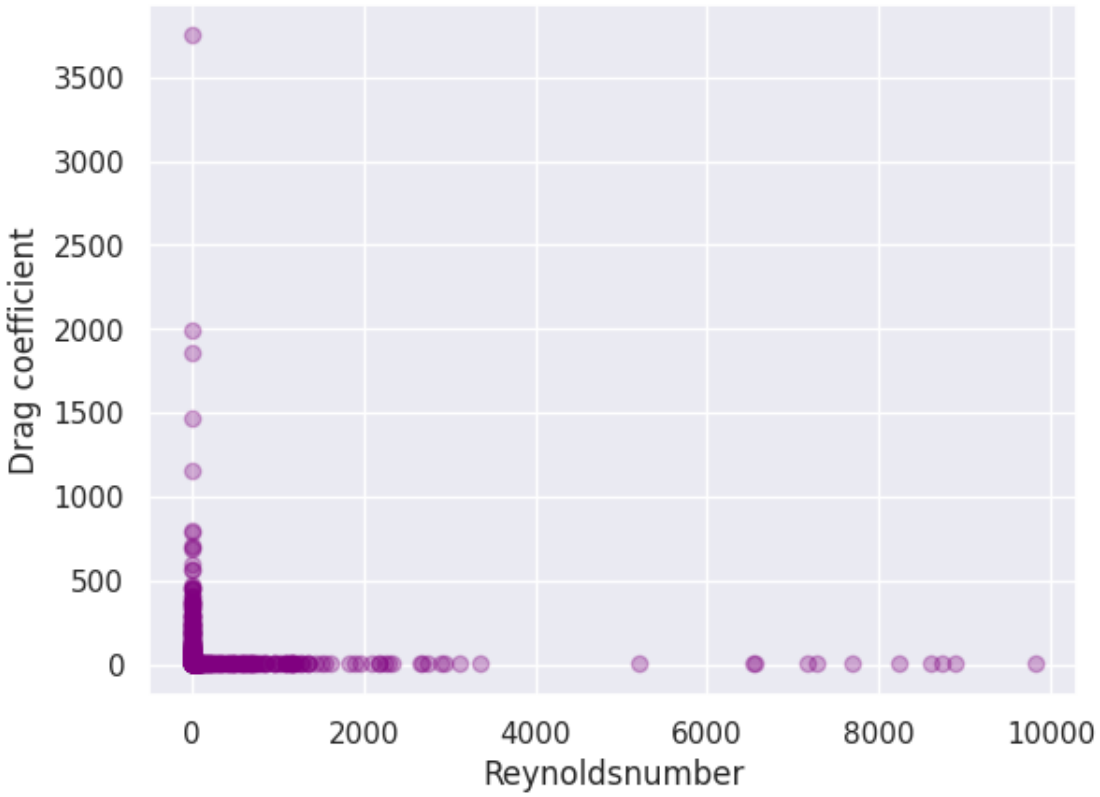


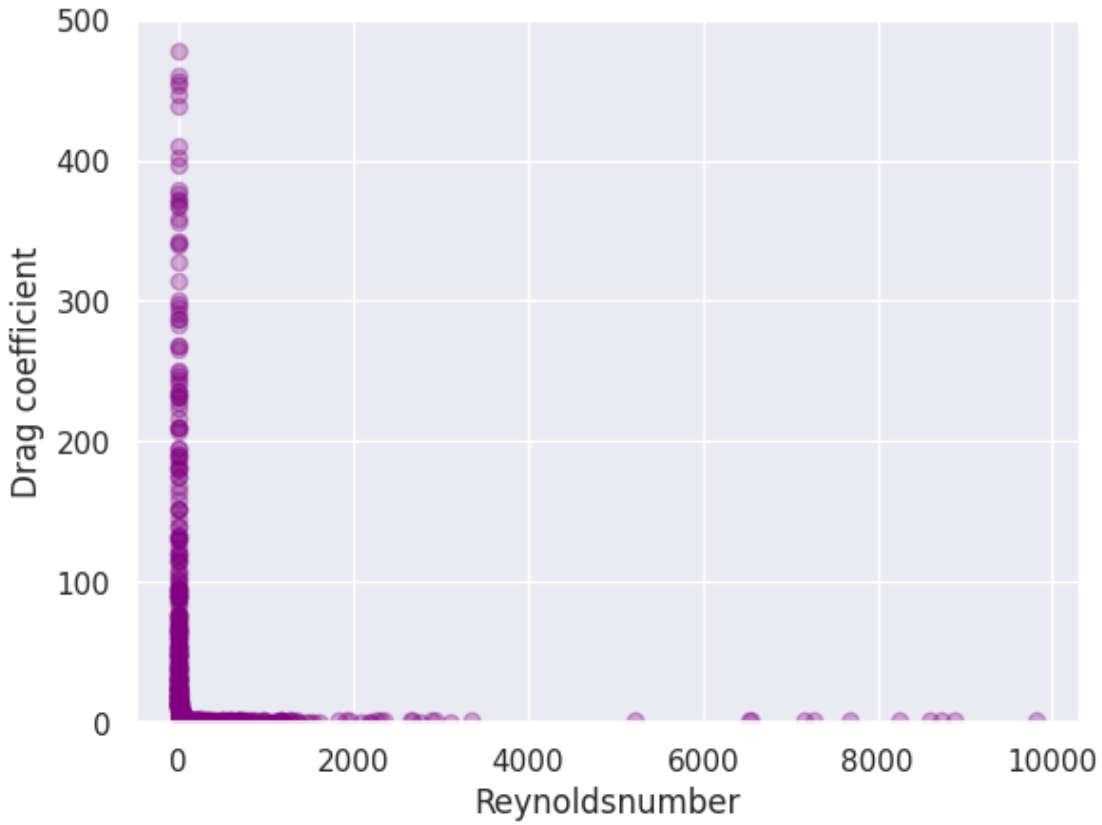
[9]:
```
# variable with small p-value
plt.scatter(X_train["viscosity"], y_train, alpha=0.3, color="purple")
plt.xlabel('viscosity')
plt.ylabel('Drag coefficient')
plt.show()

plt.scatter(X_train["viscosity"], y_train, alpha=0.3, color="purple")
plt.ylim(0, 500)
plt.xlabel('viscosity')
plt.ylabel('Drag coefficient')
plt.show()

plt.scatter(X_train["Reynoldsnumber"], y_train, alpha=0.3, color="purple")
```

```
plt.xlabel('Reynoldsnumber')
plt.ylabel('Drag coefficient')
plt.show()

plt.scatter(X_train["Reynoldsnumber"], y_train, alpha=0.3, color="purple")
plt.ylim(0, 500)
plt.xlabel('Reynoldsnumber')
plt.ylabel('Drag coefficient')
plt.show()
```

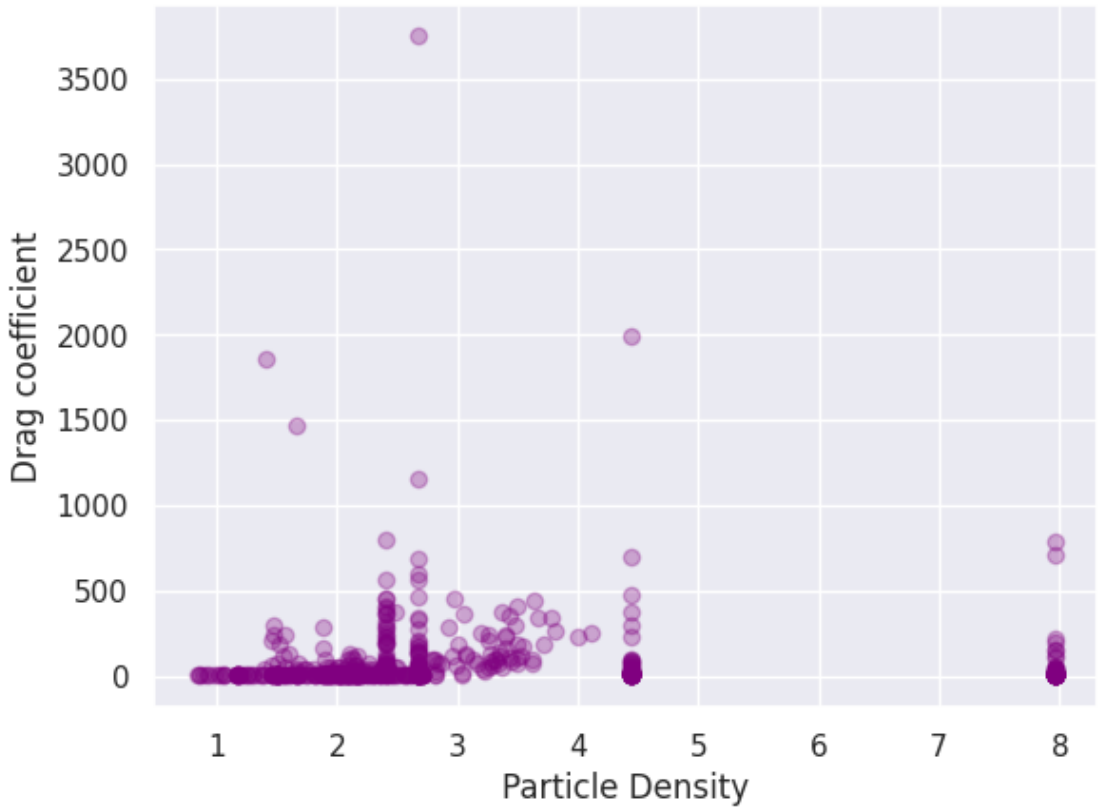



500
400
300
200
100
0
Drag coefficient
0
5
10
15
20
viscosity


Drag coefficient
3500
3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
Reynoldsnumber



[ ]:
```
# variable with bigger p-value

plt.scatter(X_train["Particle_density(g cm-3)"], y_train, alpha=0.3,␣
 ↪color="purple")
plt.xlabel('Particle Density')
plt.ylabel('Drag coefficient')
plt.show()
```



[ ]:
```
sel = SelectPercentile(f_regression, percentile=30).fit(X_train, y_train)

# Display selected feature names

sel.get_feature_names_out()
```

[ ]:
```
array(['viscosity', 'Fluid_density (g cm-3)'], dtype=object)
```

```
!pip install rfpimp
import rfpimp
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

```
Collecting rfpimp
  Downloading rfpimp-1.3.7.tar.gz (10 kB)
  Preparing metadata (setup.py) ... ent already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from rfpimp) (1.25.2)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from rfpimp) (2.0.3)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (from rfpimp) (1.2.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from rfpimp) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->rfpimp)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->rfpimp) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->rfpimp) (2024.1)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->rfpimp)
(1.11.4)
```

```
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->rfpimp)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->rfpimp)
(3.5.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib->rfpimp) (1.16.0)
Building wheels for collected packages: rfpimp
  Building wheel for rfpimp (setup.py) ... p: filename=rfpimp-1.3.7-
py3-none-any.whl size=10650
sha256=b9b0fb3a5f6b4a5f4cfdad623159679bcd91487c0cbc7b205dbf3c69af8b8f9
3
  Stored in directory:
/root/.cache/pip/wheels/6a/12/08/d5bc35127c8d69d39c1f3736a95419ab4763c
c0c80ed65bf41
Successfully built rfpimp
Installing collected packages: rfpimp
Successfully installed rfpimp-1.3.7
```

##Importing Dataset and Splitting

```
data = pd.read_csv('drag_coef.csv')
features = ['phi','Re','Cd,meas']
data_train, data_test = train_test_split(data, test_size=0.20)
data_train = data_train[features]
data_test = data_test[features]

X_train, y_train = data_train.drop('Cd,meas',axis=1),
data_train['Cd,meas']
X_test, y_test = data_test.drop('Cd,meas',axis=1),
data_test['Cd,meas']
```

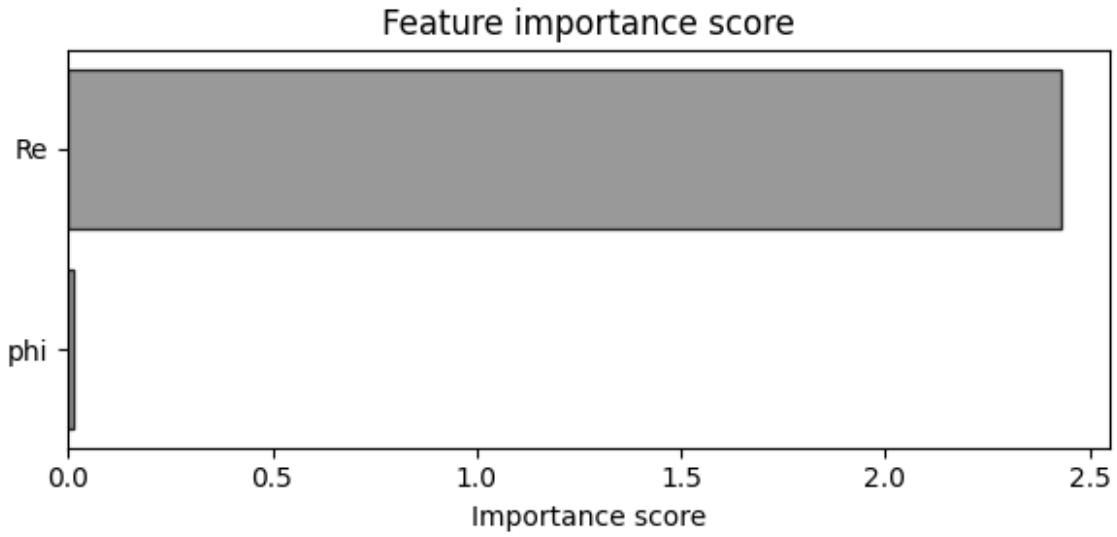
##Training Using Random Forest Regressor

```
randomforest = RandomForestRegressor(n_estimators=100, n_jobs=-1)
randomforest.fit(X_train, y_train)

imp = rfpimp.importances(randomforest, X_test, y_test)
```

##Histogram Plotting of Feature Importance

```
fig, ax = plt.subplots(figsize=(6, 3))
ax.barh(imp.index, imp['Importance'], height=0.8, facecolor='grey',
alpha=0.8, edgecolor='k')
ax.set_xlabel('Importance score')
ax.set_title('Feature importance score')
```

```
ax.set_yticks(imp.index)
ax.set_yticklabels(imp.index)
plt.gca().invert_yaxis()

fig.tight_layout()
```



```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

data = pd.read_csv('drag_coef.csv')
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size=0.3, random_state=0)

from sklearn.feature_selection import SequentialFeatureSelector as SFS

sfs = SFS(
    estimator=LinearRegression(),
    n_features_to_select='auto',
    tol=0.001,
    direction='forward',
    scoring='neg_mean_squared_error',
    cv=3
)

sfs = sfs.fit(X_train, y_train)

sfs.get_feature_names_out()
```

```
array(['Re'], dtype=object)
```

```
X_train_t = sfs.transform(X_train)
X_test_t = sfs.transform(X_test)
print(X_train_t)
```

```
[[4.1930000e-01]
 [8.3770000e+01]
 [5.2941000e+01]
 [5.5540000e+01]
 [8.6335900e+01]
 [8.8425000e+00]
 [1.1640000e-01]
 [3.7540000e-01]
 [2.1794000e+00]
 [3.4079580e+02]
 [1.1492000e+00]
 [4.8236900e+01]
 [4.3092100e+01]
 [6.2780000e-01]
 [7.9000000e-03]
```

```
[1.7636000e+01]
[6.0166700e+01]
[7.2786220e+02]
[6.7509000e+00]
[1.1143780e+02]
[3.6769000e+00]
[2.7060000e-01]
[8.3689000e+00]
[7.1600000e-02]
[4.2790000e+01]
[1.2183000e+00]
[2.0265700e+01]
[1.2152000e+00]
[2.8330500e+01]
[1.4968300e+01]
[1.7366500e+01]
[6.5433000e+00]
[7.5428950e+02]
[8.4300000e-02]
[1.1105200e+01]
[6.7685800e+01]
[6.9034300e+01]
[5.3186500e+01]
[2.9615870e+02]
[1.6723000e+00]
[1.6666420e+02]
[8.4684000e+00]
[1.3479000e+01]
[6.6582700e+01]
[5.9010000e-01]
[1.0246000e+00]
[1.3472093e+03]
[1.1388000e+00]
[1.8290000e-01]
[9.9142500e+01]
[4.3724400e+02]
[5.5049700e+02]
[5.6930000e-01]
[9.8216668e+03]
[9.4943630e+02]
[9.3160000e-01]
[1.5640000e-01]
[9.0197000e+00]
[4.4570000e-01]
[1.4340000e-01]
[1.1168900e+01]
[3.2009440e+02]
[4.9210000e-01]
[5.5213000e+00]
```

```
[2.2571600e+01]
[3.7886000e+00]
[3.8430000e-01]
[6.5349714e+03]
[1.8049590e+02]
[3.7750000e-01]
[5.5700000e-02]
[2.5709000e+00]
[6.5156470e+02]
[1.6676290e+02]
[1.0705000e+00]
[8.1023000e+00]
[8.7369701e+03]
[6.0326260e+02]
[1.8640000e-01]
[4.8340000e-01]
[5.1520000e-01]
[1.0241475e+03]
[4.0621730e+02]
[3.5901900e+01]
[1.5860000e-01]
[2.4384000e+00]
[2.7160600e+01]
[2.2195000e+00]
[2.5123000e+00]
[4.9990000e-01]
[5.2660000e-01]
[2.7560000e-01]
[1.6105880e+03]
[5.9380000e-01]
[2.6260000e-01]
[2.8800000e-01]
[4.5700000e-02]
[9.3443700e+01]
[2.2710000e-01]
[4.5510000e-01]
[3.3720000e+00]
[9.0640000e-01]
[2.0150130e+02]
[5.0211000e+00]
[2.6300000e-02]
[2.2630000e-01]
[6.2991000e+00]
[5.5736140e+02]
[6.3489600e+01]
[1.2399000e+00]
[1.4582810e+02]
[4.7792000e+00]
[6.6723710e+02]
```

```
[9.2212000e+00]
[1.3420590e+03]
[7.7250000e-01]
[4.7285000e+00]
[6.5200000e-01]
[5.4640000e-01]
[3.4471000e+00]
[4.5356000e+00]
[4.6020000e-01]
[5.1610000e-01]
[1.5025693e+03]
[2.6787500e+01]
[3.8490000e-01]
[3.5347950e+02]
[4.5891200e+01]
[1.6352000e+00]
[2.7511845e+03]
[2.7715460e+02]
[3.6606000e+00]
[4.8610000e-01]
[2.8250000e-01]
[4.0013000e+00]
[4.5792400e+01]
[6.6411530e+02]
[2.0871425e+03]
[2.2003130e+02]
[2.4166450e+02]
[1.5813000e+00]
[2.4580000e-01]
[7.4740000e-01]
[2.5307000e+00]
[6.7328700e+02]
[3.1792000e+00]
[3.4097220e+02]
[3.1660000e-01]
[2.0850000e-01]
[5.5810000e-01]
[1.3929300e+01]
[2.9513400e+02]
[3.2820580e+02]
[7.0230000e-01]
[9.0000000e-02]
[1.8325000e+00]
[1.2750000e-01]
[8.3280000e-01]
[4.3941000e+00]
[4.5900940e+02]
[9.0120000e-01]
[5.8881100e+01]
```

```
[1.0084000e+00]
[1.2860348e+03]
[8.8500000e-02]
[2.8257800e+01]
[2.5135000e+00]
[7.8700000e-02]
[2.1821987e+03]
[2.9508000e+00]
[1.1214870e+02]
[1.0821530e+02]
[1.2793000e+00]
[7.0163790e+02]
[2.0179000e+00]
[3.2850000e-01]
[2.5390000e-01]
[3.4170000e-01]
[5.3400000e-02]
[4.5970000e+00]
[1.0290000e-01]
[3.0650000e-01]
[7.1727580e+02]
[4.3691900e+01]
[2.1903578e+03]
[1.2180000e+00]
[1.2056200e+01]
[1.7740000e-01]
[8.2482000e+00]
[9.5427000e+00]
[1.5005000e+00]
[1.3261000e+00]
[6.0002000e+01]
[2.5090000e-01]
[4.4460000e-01]
[1.3360000e-01]
[4.1299000e+01]
[1.8281752e+03]
[3.6890000e-01]
[3.0851100e+01]
[2.1498500e+01]
[1.3609071e+03]
[2.1035000e+00]
[7.0526480e+02]
[2.6948700e+01]
[8.8160000e-01]
[2.7920000e-01]
[4.1316650e+02]
[8.1000000e-02]
[5.5981000e+00]
[6.1051120e+02]
```

```
[7.2600000e-02]
[1.0650000e-01]
[1.7533200e+01]
[1.2200000e+00]
[5.4727000e+00]
[2.6938000e+01]
[2.7034000e+00]
[5.4660000e-01]
[1.3111220e+02]
[9.6654110e+02]
[1.0809714e+03]
[7.5656000e+00]
[5.2417700e+01]
[2.8906000e+00]
[1.5129920e+02]
[9.4800000e-02]
[5.9457000e+00]
[2.3438000e+00]
[1.9527600e+01]
[7.7952250e+02]
[6.5440800e+02]
[4.7600000e-02]
[1.6449400e+01]
[4.0504410e+02]
[1.3643000e+00]
[9.4000000e-02]
[3.6240000e-01]
[1.4935000e+00]
[8.8829889e+03]
[1.4495163e+03]
[4.7310000e-01]
[1.7990000e-01]
[2.0171900e+01]
[2.0383000e+00]
[1.1326000e+00]
[1.3229000e+00]
[2.8560000e-01]
[4.1760000e-01]
[1.1830000e-01]
[2.9870960e+02]
[2.0660990e+02]
[7.9330700e+01]
[1.3743000e+00]
[7.2704604e+03]
[1.1998524e+03]
[4.8140000e-01]
[4.9138700e+01]
[7.6089870e+02]
[6.2283200e+01]
```

```
[3.9169000e+00]
[1.3546000e+00]
[2.8976024e+03]
[6.1540000e-01]
[1.3280000e-01]
[1.1667910e+03]
[1.0300000e-01]
[1.6321000e+00]
[1.2614812e+03]
[2.8048000e+00]
[3.9000000e-01]
[1.1220300e+02]
[1.3571600e+01]
[2.0756470e+02]
[7.3380000e-01]
[6.0691990e+02]
[3.4910000e-01]
[5.7860000e-01]
[2.8000000e-02]
[1.1460000e-01]
[1.3950150e+02]
[4.6719200e+01]
[9.8700000e-02]
[2.8180100e+01]
[6.6220000e-01]
[5.6408000e+00]
[7.6802468e+03]
[9.2513200e+01]
[1.3300000e-01]
[3.9448900e+01]
[4.7980000e-01]
[1.1637832e+03]
[2.3408129e+03]
[3.0669700e+01]
[1.0276000e+00]
[3.2858000e+00]
[8.2798060e+02]
[5.5970000e-01]
[1.7490000e-01]
[1.0664000e+00]
[2.4110000e-01]
[8.5892090e+02]
[8.2960000e-01]
[1.9511450e+02]
[3.7021250e+02]
[8.0610000e-01]
[2.8572100e+01]
[2.5032000e+01]
[8.3230000e-01]
```

```
[5.8728400e+01]
[2.0470000e-01]
[1.2830000e-01]
[2.8660000e+01]
[3.8227300e+01]
[5.6260000e-01]
[1.2772000e+00]
[5.2128367e+03]
[1.0961900e+03]
[7.3680000e-01]
[1.1558816e+03]
[2.7720000e+00]
[2.6150000e-01]
[5.8213120e+02]
[4.8431670e+02]
[1.2878000e+00]
[2.6939000e+00]
[1.5080000e-01]
[1.1118000e+00]
[1.7730000e-01]
[3.1670300e+01]
[6.8697000e+00]
[4.7533930e+02]
[4.5510000e-01]
[4.0180000e-01]
[1.1860000e-01]
[3.6360000e-01]
[1.6197000e+00]
[5.6825000e+00]
[2.3385020e+02]
[1.5701000e+00]
[1.8650000e-01]
[1.4450000e-01]
[3.4281100e+01]
[4.3723000e+00]
[7.5332300e+01]
[7.6860000e-01]
[5.4673000e+00]
[1.1324000e+00]
[2.2030000e-01]
[2.9690000e-01]
[3.0210000e-01]
[1.6941200e+01]
[9.4739000e+00]
[4.0840000e-01]
[4.1246200e+01]
[1.3031000e+00]
[1.2650000e-01]
[1.0960000e-01]
```

```
[5.8889500e+01]
[5.0377700e+01]
[3.6170000e-01]
[2.1315000e+01]
[3.3800000e-01]
[5.5310000e-01]
[2.1083200e+01]
[1.0261080e+03]
[1.9017494e+03]
[8.8383720e+02]
[1.0196100e+02]
[1.0170000e-01]
[6.4700000e-02]
[4.8277000e+00]
[8.4822600e+02]
[7.9300000e-02]
[5.3143400e+01]
[4.4131600e+01]
[2.8410000e-01]
[2.6193000e+00]
[1.2355900e+01]
[2.2240000e-01]
[1.0073000e+00]
[8.2900000e-01]
[6.7750000e-01]
[5.8260000e-01]
[4.4648000e+00]
[1.8542600e+01]
[8.3925000e+00]
[2.3438000e+00]
[2.5294000e+00]
[1.6622000e+00]
[1.8110000e-01]
[3.4096000e+00]
[1.3145000e+00]
[6.3486800e+01]
[1.1153980e+03]
[3.3371600e+01]
[4.0060000e-01]
[2.2941771e+03]
[6.0164700e+02]
[7.0084340e+02]
[1.6340000e-01]
[1.3805110e+02]
[7.5520000e-01]
[7.3250000e-01]
[4.7430090e+02]
[1.6414000e+01]
[2.6780091e+03]
```

```
[7.2287400e+01]
[2.4639000e+00]
[1.1993480e+02]
[6.4080000e-01]
[1.6340800e+01]
[2.5671700e+01]
[1.5143500e+01]
[7.1512940e+02]
[4.3309200e+01]
[1.1197250e+02]
[2.9357822e+03]
[1.8864000e+00]
[1.0729800e+01]
[7.3960000e-01]
[5.1890000e-01]
[7.6823500e+01]
[7.9573100e+01]
[3.4301300e+01]
[7.1609495e+03]
[1.0690100e+01]
[7.8760000e-01]
[3.2610000e-01]
[2.9347500e+01]
[5.0192000e+00]
[1.7814000e+00]
[5.9292270e+02]
[5.0599540e+02]
[9.3300000e-02]
[1.6230000e-01]
[2.2560613e+03]
[7.2600000e-02]
[7.1764700e+01]
[9.0544000e+00]
[4.7003000e+00]
[4.5317000e+02]
[4.7440000e-01]
[6.6908700e+01]
[1.1190000e-01]
[1.6694000e+00]
[4.9100000e+00]
[2.4654700e+01]
[4.3050000e-01]
[9.4900000e-01]
[2.6639400e+01]
[1.6300000e-02]
[3.5130000e+00]
[4.8320000e-01]
[6.1452000e+00]
[3.7390400e+01]
```

```
[1.2156626e+03]
[5.4200000e-02]
[1.4407000e+01]
[2.6501694e+03]
[1.0964600e+01]
[1.5377691e+03]
[9.5120000e-01]
[6.1651000e+00]
[1.0870000e-01]
[4.8940000e-01]
[1.1486400e+01]
[3.4530000e-01]
[1.1544810e+03]
[4.4739000e+00]
[5.4428220e+02]
[8.5262000e+00]
[2.3436600e+01]
[4.3145000e+00]
[1.1494104e+03]
[8.2407970e+03]
[1.5249400e+01]
[2.7690000e-01]
[1.9380000e-01]
[3.8667000e+00]
[6.5489660e+03]
[1.4643600e+01]
[4.7701000e+00]
[1.9810000e-01]
[1.1570000e-01]
[1.1741000e+00]
[1.9566461e+03]
[4.2000000e-01]
[1.1809563e+03]
[4.9067000e+00]
[9.7210000e-01]
[1.0018900e+01]
[4.8959250e+02]
[3.5750000e+00]
[1.2519600e+01]
[8.7120000e-01]
[3.7800000e-02]
[6.5190000e-01]
[7.4510000e-01]
[9.0167200e+01]
[4.4780000e-01]
[2.3881020e+02]
[1.2680188e+03]
[8.5948375e+03]
[2.3958600e+01]
[3.7500000e-02]
```

[6.4000000e-02]
[5.2851900e+01]
[1.2074800e+01]
[2.7130000e-01]
[6.0900000e-02]
[6.3650000e-01]
[9.3251000e+00]
[6.3500000e-02]
[6.8259900e+01]
[1.4832000e+01]
[1.6604800e+01]
[1.6467600e+01]
[1.0916000e+00]
[8.8479000e+00]
[1.9870000e-01]
[2.8187020e+02]
[1.0815541e+03]
[1.1770000e-01]
[1.3150000e-01]
[1.2480000e+00]
[1.9759500e+01]
[4.7361000e+00]
[1.0032430e+02]
[5.0355570e+02]
[3.8990000e-01]
[6.2901000e+00]
[2.2197000e+00]
[1.1030000e-01]
[9.5376950e+02]
[9.6637550e+02]
[5.7229320e+02]
[3.8620000e-01]
[1.2605300e+01]
[1.7560000e-01]
[4.5842600e+01]
[6.0230000e+00]
[4.1248460e+02]
[2.5040000e-01]
[8.8060000e-01]
[2.0342000e+00]
[7.8898000e+01]
[5.3070900e+01]
[6.2430000e-01]
[2.9380000e-01]
[1.7202100e+01]
[1.3620000e-01]
[8.7365000e+00]
[8.3050710e+02]
[7.9400000e-02]

```
[7.4510000e-01]
[4.4274000e+00]
[1.3480000e-01]
[1.0871819e+03]
[2.5142510e+02]
[3.4370000e-01]
[5.1360000e-01]
[2.2285600e+01]
[3.3476233e+03]
[1.5098370e+02]
[1.8043000e+00]
[3.1059824e+03]
[4.5180000e-01]
[4.5000000e-01]
[3.5700000e-02]
[7.9880000e-01]
[5.2910000e+00]
[2.8371000e+00]
[3.0190000e-01]
[1.4360000e-01]
[8.3052000e+00]
[2.1362250e+02]
[1.1682120e+03]
[1.3680000e-01]]
```

Multiple Linear Regression

Importing the libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso,LinearRegression
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.feature_selection import SelectFromModel
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing the dataset

```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
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 = 0)
```

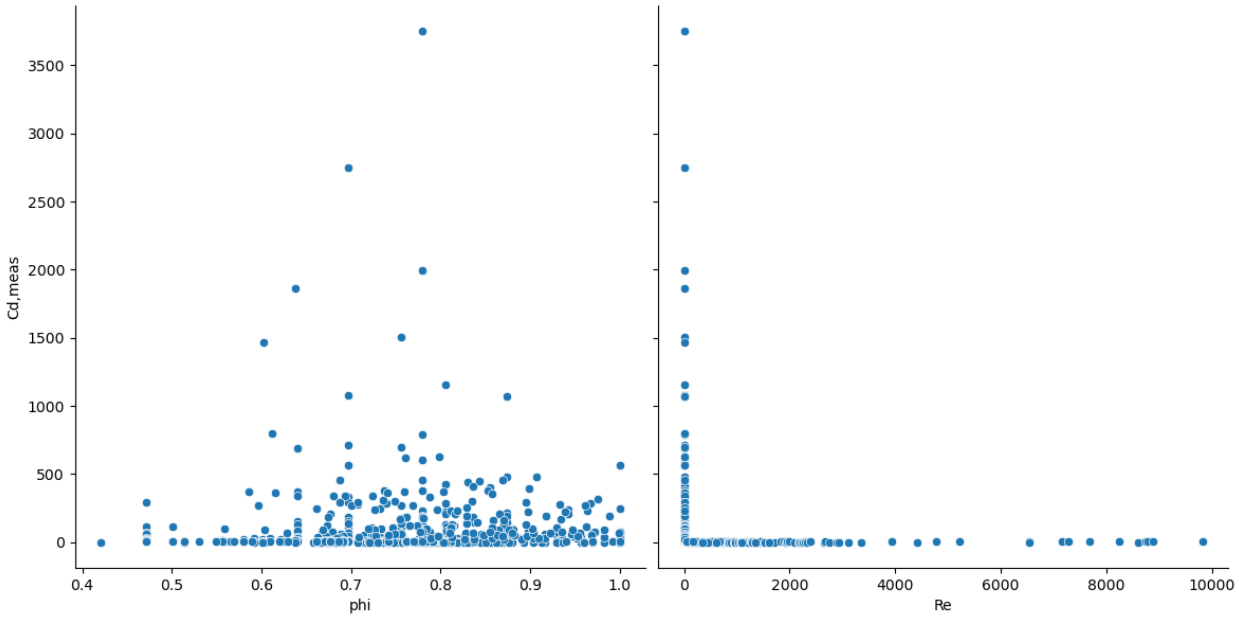
##Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

##Relation Between Input and Target Variable

```
plt.figure(figsize=(16, 8))
sns.pairplot(dataset, x_vars=['phi','Re'], y_vars='Cd,meas', height=6,
aspect=1, kind='scatter')
plt.show()
```

```
<Figure size 1600x800 with 0 Axes>
```



##Applying Lasso Regression

```
# Lasso + Linear Regression
lasso_linear = Lasso(alpha=0.1)
regressor=lasso_linear.fit(X_train, y_train)
sel_ = SelectFromModel(Lasso(alpha=0.001, random_state=10))
sel_.fit((X_train), y_train)
```

```
SelectFromModel(estimator=Lasso(alpha=0.001, random_state=10))
```

```
# Regressor coefficients and intercept
print(f'Coefficient: {regressor.coef_}')
print(f'Intercept: {regressor.intercept_}')

##Selection of Features
sel_.get_support()
```

```
Coefficient: [ -5.5202152  -23.43823414]
Intercept: 80.09389577039282
```

```
array([ True,  True])
```

```
# list with the selected features and the outputs
selected_feat = sel_.get_support()

print('total features: {}'.format((X_train.shape[1])))
print('selected features: {}'.format(np.sum(selected_feat)))
print('features with coefficients shrank to zero: {}'.format(
    np.sum(sel_.estimator_.coef_ == 0)))
```

```
total features: 2
selected features: 2
features with coefficients shrank to zero: 0
```

```
betas = []
penalties = [0, 0.0001, 0.001, 0.001, 0.1, 1]

X_train_df = pd.DataFrame(X_train, columns=['phi','Re'])

for penalty in penalties:

    pipe = Pipeline([
        ("lasso", Lasso(alpha=penalty, random_state=10))
    ])

    pipe.fit(X_train, y_train)

    betas.append(pd.Series(pipe.named_steps["lasso"].coef_))

betas = pd.concat(betas, axis=1)
betas.columns = [f"{penalty}" for penalty in penalties]
betas.index = X_train_df.columns
betas.head()
```

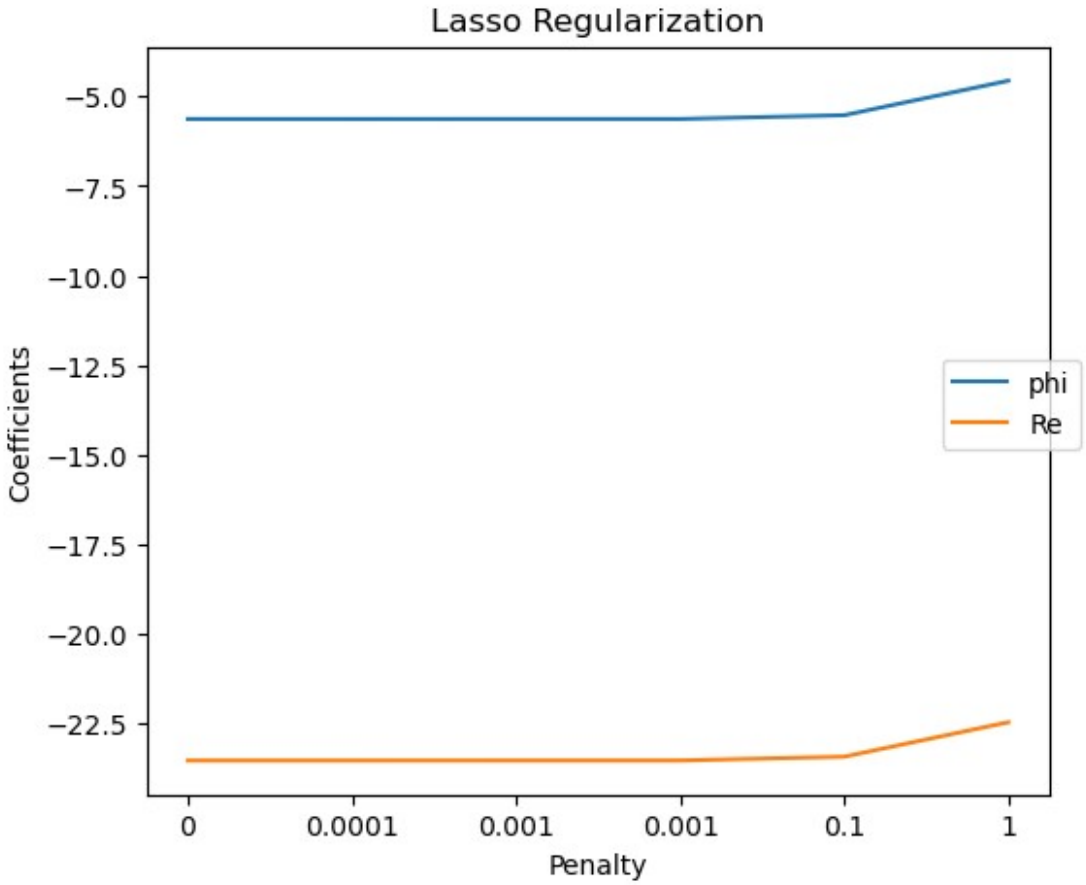
```
c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\sklearn\
base.py:1474: UserWarning: With alpha=0, this algorithm does not
converge well. You are advised to use the LinearRegression estimator
  return fit_method(estimator, *args, **kwargs)
c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\sklearn\
linear_model\_coordinate_descent.py:678: UserWarning: Coordinate
descent with no regularization may lead to unexpected results and is
discouraged.
  model = cd_fast.enet_coordinate_descent(
c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\sklearn\
linear_model\_coordinate_descent.py:678: ConvergenceWarning: Objective
did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation.
Duality gap: 2.011e+07, tolerance: 4.059e+03 Linear regression models
with null weight for the l1 regularization term are more efficiently
fitted using one of the solvers implemented in
sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
```

	0	0.0001	0.001	0.001	0.1	1
phi	-5.627206	-5.627099	-5.626136	-5.626136	-5.520215	-4.557297
Re	-23.545225	-23.545118	-23.544155	-23.544155	-23.438234	-22.475316

```
betas.T.plot(figsize=(6,5), legend=False)
plt.ylabel("Coefficients")
plt.xlabel("Penalty")
plt.title("Lasso Regularization")
```

```
plt.legend(X_train_df.columns,
           bbox_to_anchor = (1.05, 0.6))
```

```
<matplotlib.legend.Legend at 0x1b9462d1fc0>
```



Predicting the Test set results

```
y_pred = lasso_linear.predict(X_test)
y_pred_train = lasso_linear.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[ 8.15e+01  1.49e+00]
 [ 8.66e+01  2.20e+01]
 [ 8.53e+01  1.17e+01]
 [ 1.02e+02  6.23e+01]
 [ 8.20e+01  1.26e+00]
 [ 9.64e+01  1.92e+01]
 [ 8.66e+01  8.22e+01]
 [ 8.08e+01  2.24e+02]
 [ 7.63e+01  1.00e+00]
```

```
[ 7.92e+01  5.26e+01]
[ 9.02e+01  2.96e+02]
[ 8.38e+01  6.92e+01]
[ 5.61e+01  1.45e+00]
[ 8.82e+01  1.07e+02]
[ 7.71e+01  7.48e+01]
[ 8.51e+01  5.42e+00]
[ 1.02e+02  1.36e+01]
[ 7.25e+01  1.03e+00]
[ 8.53e+01  3.84e+00]
[ 9.57e+01  2.69e+02]
[ 8.26e+01  8.61e-01]
[ 8.76e+01  6.20e+00]
[ 1.02e+02  1.16e+02]
[ 1.00e+02  5.61e+00]
[ 8.66e+01  1.26e+01]
[ 8.36e+01  1.44e+02]
[ 8.51e+01  4.95e+00]
[-5.72e+00  1.84e+00]
[ 7.96e+01  1.29e+00]
[ 8.80e+01  2.06e+00]
[ 7.95e+01  8.64e-01]
[ 7.87e+01  1.17e+00]
[ 8.19e+01  7.14e+00]
[ 7.67e+01  2.57e+00]
[ 9.16e+01  1.12e+01]
[ 8.90e+01  3.72e+01]
[ 8.73e+01  2.56e+00]
[ 8.20e+01  7.10e+01]
[ 8.51e+01  3.12e+00]
[ 8.42e+01  2.55e+01]
[ 9.32e+01  5.73e+00]
[ 9.07e+01  3.76e+01]
[ 7.68e+01  1.18e+00]
[ 9.07e+01  4.18e+01]
[ 9.23e+01  1.88e+00]
[ 7.69e+01  1.10e+00]
[ 8.51e+01  1.51e+00]
[ 1.01e+02  5.70e+00]
[ 7.57e+01  7.21e+01]
[ 6.48e+01  1.16e+00]
[ 5.87e+01  1.67e+00]
[ 1.02e+02  6.83e+01]
[ 8.61e+01  6.83e+00]
[ 7.73e+01  1.29e+00]
[ 9.30e+01  4.57e+00]
[ 8.67e+01  3.81e+02]
[ 8.75e+01  3.65e+00]
[ 8.32e+01  1.23e+00]
```

```
[ 7.01e+01  1.19e+00]
[ 8.17e+01  6.83e+01]
[ 8.75e+01  4.64e+00]
[ 8.53e+01  9.92e+00]
[ 9.50e+01  2.76e+01]
[ 8.67e+01  1.02e+00]
[ 8.05e+01  5.94e+01]
[ 9.03e+01  1.66e+00]
[ 8.53e+01  1.71e+00]
[ 7.95e+01  7.77e-01]
[ 8.24e+01  1.31e+00]
[ 8.65e+01  1.05e+01]
[ 8.53e+01  1.49e+01]
[ 9.02e+01  2.80e+02]
[ 8.25e+01  4.83e+01]
[ 8.76e+01  1.37e+00]
[ 8.05e+01  6.77e+01]
[ 8.28e+01  1.10e+02]
[ 8.16e+01  3.52e+00]
[ 8.19e+01  1.00e+01]
[ 8.20e+01  1.07e+03]
[ 8.18e+01  6.55e+00]
[ 9.04e+01  6.39e+00]
[ 9.17e+01  2.09e+02]
[ 8.78e+01  1.50e+03]
[ 6.33e+01  8.60e-01]
[ 9.54e+01  2.37e+01]
[ 8.18e+01  4.89e+00]
[ 8.76e+01  6.19e+02]
[ 7.98e+01  6.27e+01]
[ 8.86e+01  2.61e+00]
[ 8.77e+01  5.43e+01]
[ 8.94e+01  5.55e+01]
[ 8.16e+01  9.17e+01]
[ 8.66e+01  2.56e+01]
[ 9.06e+01  8.60e+00]
[ 8.01e+01  9.80e-01]
[ 7.88e+01  1.82e+01]
[ 7.91e+01  2.79e+02]
[ 8.23e+01  1.26e+00]
[ 8.53e+01  1.31e+02]
[ 8.68e+01  1.88e+01]
[ 8.53e+01  4.28e+02]
[ 9.35e+01  3.95e+01]
[ 7.57e+01  1.69e+01]
[ 8.18e+01  5.57e+00]
[ 7.57e+01  4.17e+01]
[ 9.33e+01  7.50e+00]
[ 8.25e+01  1.07e+01]
```

```
[ 9.09e+01  3.39e+02]
[ 7.80e+01  6.78e+01]
[ 4.99e+01  8.81e-01]
[ 8.50e+01  1.74e+01]
[ 9.09e+01  2.66e+00]
[ 8.86e+01  2.54e+00]
[ 8.19e+01  3.31e+00]
[ 7.72e+01  7.12e+01]
[ 7.57e+01  7.68e+01]
[ 8.47e+01  3.72e+00]
[ 7.38e+01  1.45e+00]
[ 7.83e+01  4.43e+01]
[ 5.45e+01  9.68e-01]
[ 8.19e+01  1.11e+01]
[ 4.91e+01  1.12e+00]
[ 9.02e+01  1.97e+00]
[ 7.97e+01  1.07e+00]
[ 7.89e+01  7.45e+01]
[ 9.35e+01  8.09e+01]
[ 9.34e+01  1.70e+00]
[ 8.32e+01  1.47e+00]
[ 1.02e+02  3.21e+01]
[ 6.19e+01  1.61e+00]
[ 7.71e+01  6.60e-01]
[ 8.48e+01  1.79e+00]
[ 8.52e+01  6.70e+00]
[ 8.19e+01  2.48e+01]
[ 6.80e+01  1.19e+00]
[-4.99e+00  9.35e-01]
[ 9.07e+01  1.83e+02]
[ 8.85e+01  3.64e+02]
[ 1.61e+01  1.60e+00]
[ 8.66e+01  1.10e+01]
[ 8.91e+01  1.64e+00]
[ 8.33e+01  1.15e+01]
[ 8.53e+01  2.20e+02]
[ 8.51e+01  4.15e+00]
[ 9.05e+01  8.61e+00]
[ 8.02e+01  8.28e-01]
[ 1.01e+02  6.73e+00]
[ 8.18e+01  1.46e+00]
[ 8.61e+01  6.53e+00]
[ 7.73e+01  2.16e+00]
[ 4.68e+01  1.18e+00]
[ 8.78e+01  4.36e+01]
[ 8.53e+01  4.13e+01]
[ 8.59e+01  2.20e+00]
[ 7.81e+01  1.33e+00]
[ 7.90e+01  1.72e+02]
```

```
 [ 6.61e+01  9.83e-01]
 [ 8.55e+01  1.28e+00]
 [ 7.76e+01  2.71e+02]
 [ 8.29e+01  7.56e+01]
 [ 8.53e+01  2.09e+00]
 [ 1.02e+02  2.25e+01]
 [ 8.14e+01  1.98e+00]
 [ 8.16e+01  5.34e+01]
 [ 8.66e+01  3.66e+01]
 [ 8.62e+01  3.31e+02]]
```

Evaluating the Model Performance

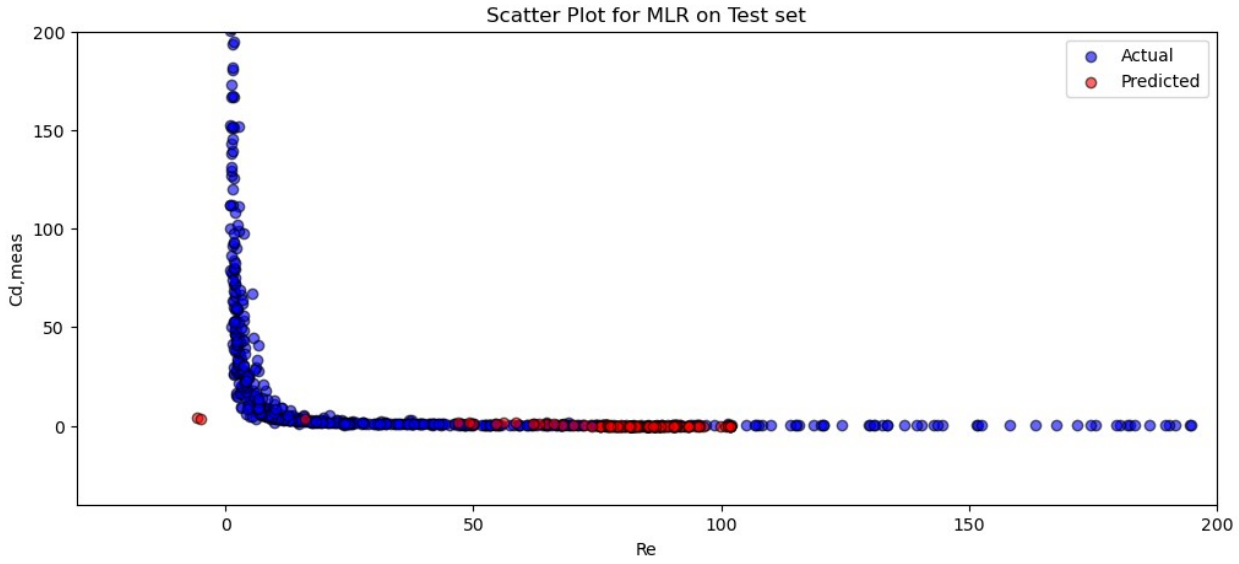
```
from sklearn.metrics import r2_score, mean_squared_error
print('R_2 Score :', r2_score(y_test, y_pred))
print('Mean Squared Error :', mean_squared_error(y_test, y_pred))
print('MSE for Training set:', mean_squared_error(y_train,
y_pred_train))
print('R_2 score for Training set:',r2_score(y_train, y_pred_train))
```

```
R_2 Score : -0.0008807435837168143
Mean Squared Error : 27675.533771170874
MSE for Training set: 60747.311114852986
R_2 score for Training set: 0.009275015063040626
```

##Plotting Results

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6,
label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' , edgecolors='black',
alpha=0.6,label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for MLR on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

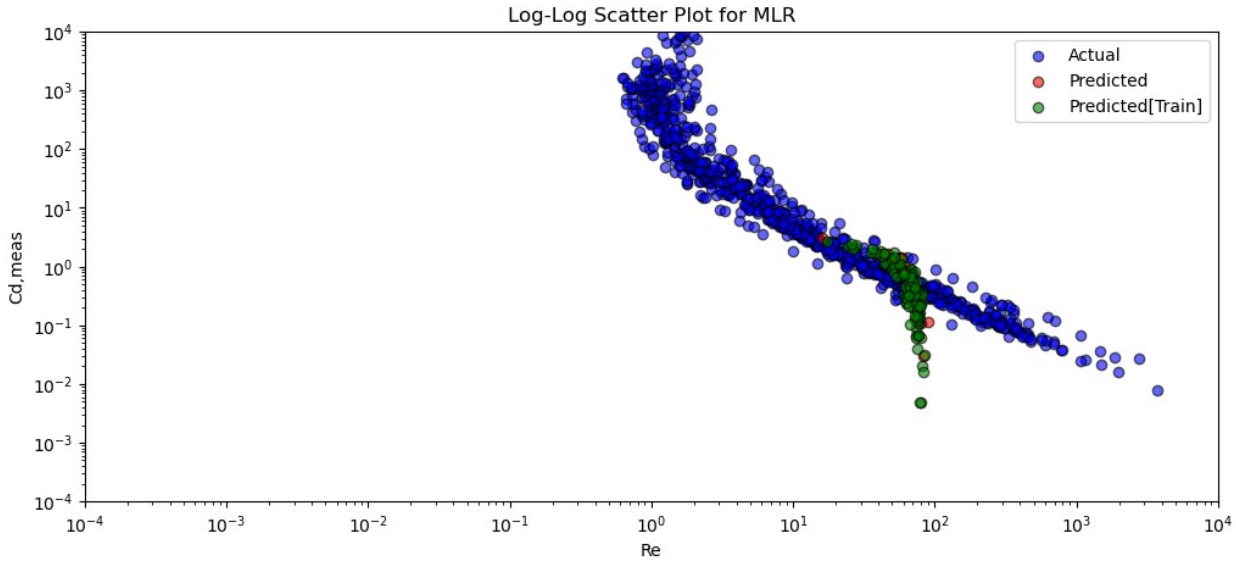
```
<matplotlib.legend.Legend at 0x1b94c2fdf60>
```



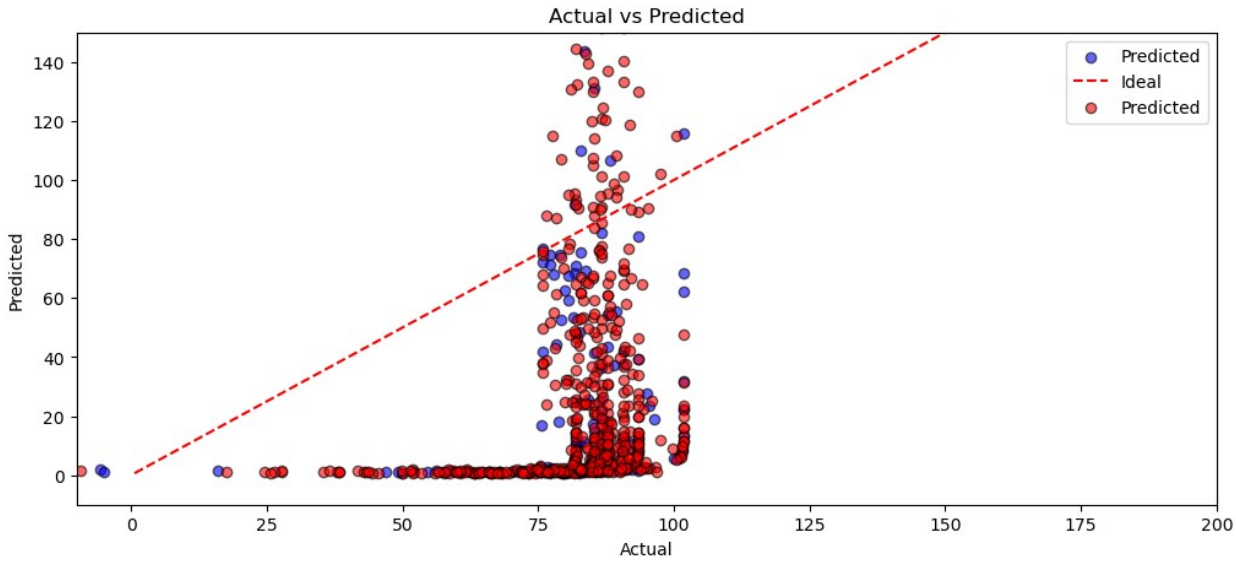

```
print('R_2 Score :', r2_score(y_pred, X_test[:,-1]))
```

```
R_2 Score : -32.535789358252124
```

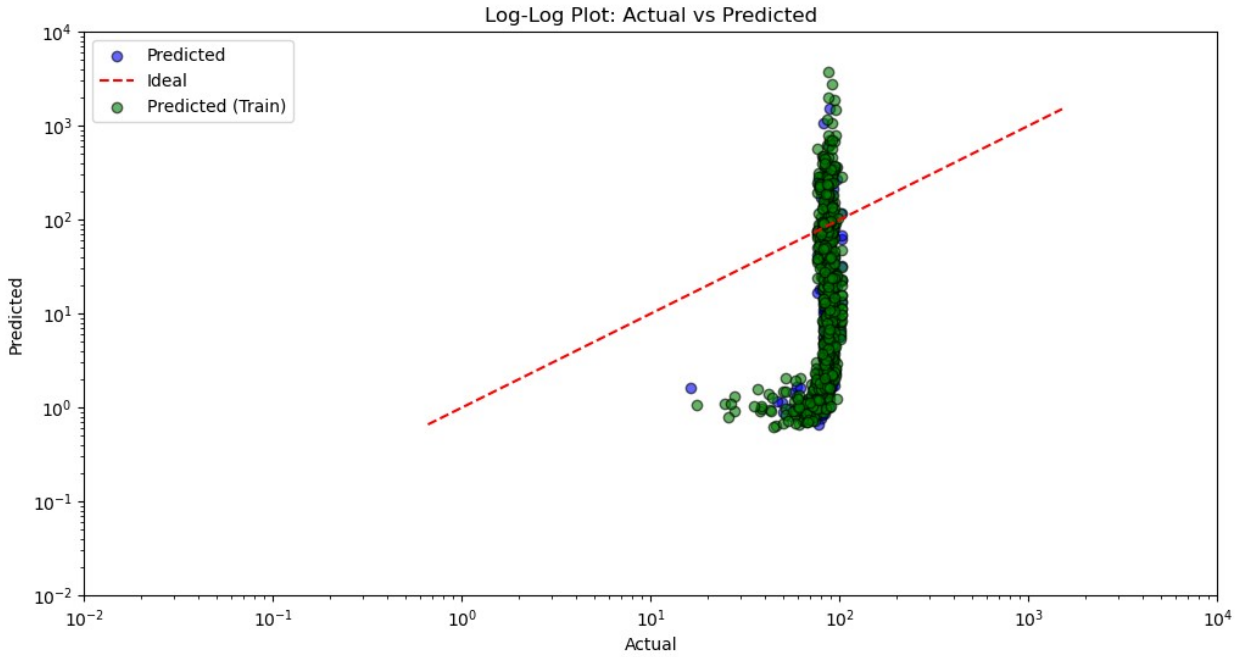
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6,
label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c =
'green' ,edgecolors='black', alpha=0.6, label = 'Predicted[Train]')
plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)
plt.ylim(0.0001,10000)
plt.title('Log-Log Scatter Plot for MLR')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,
label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 200)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```

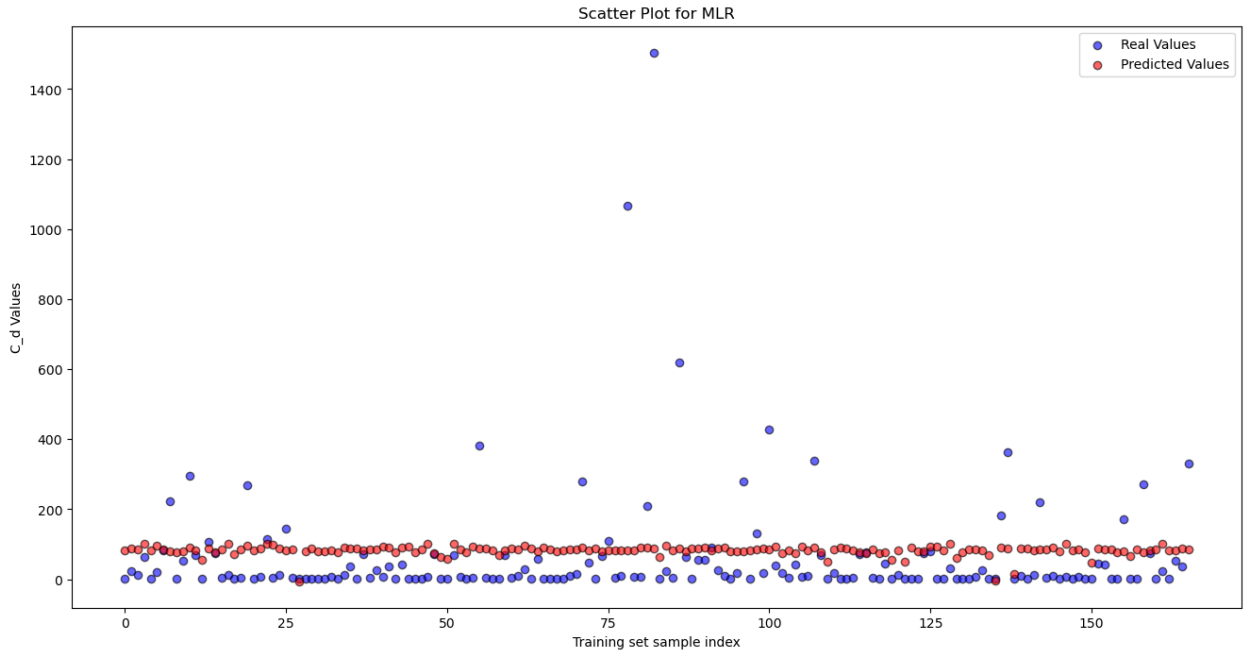


```
plt.figure(figsize=(12, 6))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black',
alpha=0.6, label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black',
alpha=0.6, label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.01,10000)
plt.xlim(0.01,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```

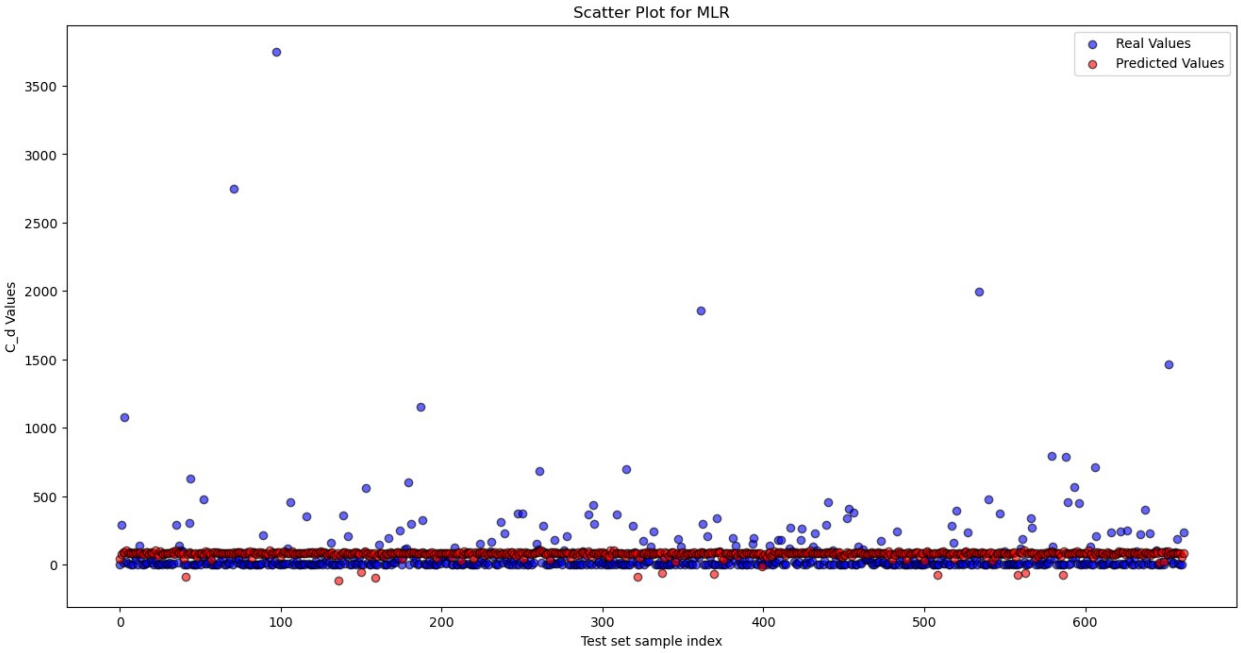



```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for MLR')
plt.xlabel('Training set sample index')
```

```
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_train)), y_train, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for MLR')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```


Scatter Plot for MLR
Real Values
Predicted Values
C_d Values
Test set sample index
3500
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
600

Multiple Linear Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
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 = 0)

print(X_train)
```

```
[[6.91e-01 2.39e+03]
 [8.95e-01 9.81e-02]
 [6.97e-01 4.60e+00]
 ...
 [7.68e-01 2.14e+02]
 [8.29e-01 1.17e+03]
 [7.96e-01 1.37e-01]]
```

```
print(len(X_train))
```

```
662
```

```
print(len(X_test))
```

```
166
```

Training the Multiple Linear Regression model on the Training set

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

##Applying K fold Cross Validation

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor, X = X_train, y =
y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: -4.09 %
Standard Deviation: 14.61 %
```

Predicting the Test set results

```
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[ 8.15e+01  1.49e+00]
 [ 8.67e+01  2.20e+01]
 [ 8.53e+01  1.17e+01]
 [ 1.02e+02  6.23e+01]
 [ 8.20e+01  1.26e+00]
 [ 9.67e+01  1.92e+01]
 [ 8.66e+01  8.22e+01]
 [ 8.07e+01  2.24e+02]
 [ 7.63e+01  1.00e+00]
 [ 7.91e+01  5.26e+01]
 [ 9.03e+01  2.96e+02]
 [ 8.38e+01  6.92e+01]
 [ 5.61e+01  1.45e+00]
 [ 8.83e+01  1.07e+02]
 [ 7.70e+01  7.48e+01]
 [ 8.51e+01  5.42e+00]
 [ 1.02e+02  1.36e+01]
 [ 7.24e+01  1.03e+00]
 [ 8.53e+01  3.84e+00]
 [ 9.59e+01  2.69e+02]
 [ 8.26e+01  8.61e-01]
 [ 8.76e+01  6.20e+00]
 [ 1.02e+02  1.16e+02]
 [ 1.00e+02  5.61e+00]
 [ 8.66e+01  1.26e+01]
 [ 8.36e+01  1.44e+02]
 [ 8.51e+01  4.95e+00]
 [-6.03e+00  1.84e+00]
 [ 7.96e+01  1.29e+00]
 [ 8.80e+01  2.06e+00]
 [ 7.95e+01  8.64e-01]
```

```
[ 7.87e+01  1.17e+00]
[ 8.18e+01  7.14e+00]
[ 7.65e+01  2.57e+00]
[ 9.18e+01  1.12e+01]
[ 8.90e+01  3.72e+01]
[ 8.73e+01  2.56e+00]
[ 8.19e+01  7.10e+01]
[ 8.51e+01  3.12e+00]
[ 8.41e+01  2.55e+01]
[ 9.34e+01  5.73e+00]
[ 9.08e+01  3.76e+01]
[ 7.68e+01  1.18e+00]
[ 9.08e+01  4.18e+01]
[ 9.25e+01  1.88e+00]
[ 7.69e+01  1.10e+00]
[ 8.51e+01  1.51e+00]
[ 1.01e+02  5.70e+00]
[ 7.56e+01  7.21e+01]
[ 6.47e+01  1.16e+00]
[ 5.87e+01  1.67e+00]
[ 1.02e+02  6.83e+01]
[ 8.61e+01  6.83e+00]
[ 7.71e+01  1.29e+00]
[ 9.32e+01  4.57e+00]
[ 8.67e+01  3.81e+02]
[ 8.76e+01  3.65e+00]
[ 8.32e+01  1.23e+00]
[ 7.02e+01  1.19e+00]
[ 8.16e+01  6.83e+01]
[ 8.75e+01  4.64e+00]
[ 8.53e+01  9.92e+00]
[ 9.52e+01  2.76e+01]
[ 8.69e+01  1.02e+00]
[ 8.04e+01  5.94e+01]
[ 9.04e+01  1.66e+00]
[ 8.54e+01  1.71e+00]
[ 7.95e+01  7.77e-01]
[ 8.24e+01  1.31e+00]
[ 8.66e+01  1.05e+01]
[ 8.53e+01  1.49e+01]
[ 9.03e+01  2.80e+02]
[ 8.24e+01  4.83e+01]
[ 8.77e+01  1.37e+00]
[ 8.04e+01  6.77e+01]
[ 8.28e+01  1.10e+02]
[ 8.16e+01  3.52e+00]
[ 8.18e+01  1.00e+01]
[ 8.19e+01  1.07e+03]
[ 8.18e+01  6.55e+00]
```

```
[ 9.05e+01  6.39e+00]
[ 9.18e+01  2.09e+02]
[ 8.79e+01  1.50e+03]
[ 6.32e+01  8.60e-01]
[ 9.56e+01  2.37e+01]
[ 8.17e+01  4.89e+00]
[ 8.76e+01  6.19e+02]
[ 7.97e+01  6.27e+01]
[ 8.87e+01  2.61e+00]
[ 8.78e+01  5.43e+01]
[ 8.95e+01  5.55e+01]
[ 8.15e+01  9.17e+01]
[ 8.67e+01  2.56e+01]
[ 9.07e+01  8.60e+00]
[ 8.00e+01  9.80e-01]
[ 7.87e+01  1.82e+01]
[ 7.89e+01  2.79e+02]
[ 8.23e+01  1.26e+00]
[ 8.53e+01  1.31e+02]
[ 8.69e+01  1.88e+01]
[ 8.53e+01  4.28e+02]
[ 9.37e+01  3.95e+01]
[ 7.55e+01  1.69e+01]
[ 8.17e+01  5.57e+00]
[ 7.56e+01  4.17e+01]
[ 9.35e+01  7.50e+00]
[ 8.24e+01  1.07e+01]
[ 9.10e+01  3.39e+02]
[ 7.78e+01  6.78e+01]
[ 4.97e+01  8.81e-01]
[ 8.50e+01  1.74e+01]
[ 9.11e+01  2.66e+00]
[ 8.87e+01  2.54e+00]
[ 8.18e+01  3.31e+00]
[ 7.71e+01  7.12e+01]
[ 7.56e+01  7.68e+01]
[ 8.47e+01  3.72e+00]
[ 7.36e+01  1.45e+00]
[ 7.81e+01  4.43e+01]
[ 5.44e+01  9.68e-01]
[ 8.18e+01  1.11e+01]
[ 4.90e+01  1.12e+00]
[ 9.03e+01  1.97e+00]
[ 7.98e+01  1.07e+00]
[ 7.88e+01  7.45e+01]
[ 9.37e+01  8.09e+01]
[ 9.35e+01  1.70e+00]
[ 8.31e+01  1.47e+00]
[ 1.02e+02  3.21e+01]
```

```
 [ 6.19e+01  1.61e+00]
 [ 7.72e+01  6.60e-01]
 [ 8.48e+01  1.79e+00]
 [ 8.52e+01  6.70e+00]
 [ 8.19e+01  2.48e+01]
 [ 6.79e+01  1.19e+00]
 [-5.39e+00  9.35e-01]
 [ 9.08e+01  1.83e+02]
 [ 8.86e+01  3.64e+02]
 [ 1.60e+01  1.60e+00]
 [ 8.66e+01  1.10e+01]
 [ 8.92e+01  1.64e+00]
 [ 8.33e+01  1.15e+01]
 [ 8.53e+01  2.20e+02]
 [ 8.51e+01  4.15e+00]
 [ 9.06e+01  8.61e+00]
 [ 8.01e+01  8.28e-01]
 [ 1.02e+02  6.73e+00]
 [ 8.18e+01  1.46e+00]
 [ 8.61e+01  6.53e+00]
 [ 7.72e+01  2.16e+00]
 [ 4.68e+01  1.18e+00]
 [ 8.78e+01  4.36e+01]
 [ 8.53e+01  4.13e+01]
 [ 8.59e+01  2.20e+00]
 [ 7.80e+01  1.33e+00]
 [ 7.89e+01  1.72e+02]
 [ 6.61e+01  9.83e-01]
 [ 8.56e+01  1.28e+00]
 [ 7.75e+01  2.71e+02]
 [ 8.29e+01  7.56e+01]
 [ 8.53e+01  2.09e+00]
 [ 1.02e+02  2.25e+01]
 [ 8.14e+01  1.98e+00]
 [ 8.15e+01  5.34e+01]
 [ 8.67e+01  3.66e+01]
 [ 8.62e+01  3.31e+02]]
```

Evaluating the Model Performance

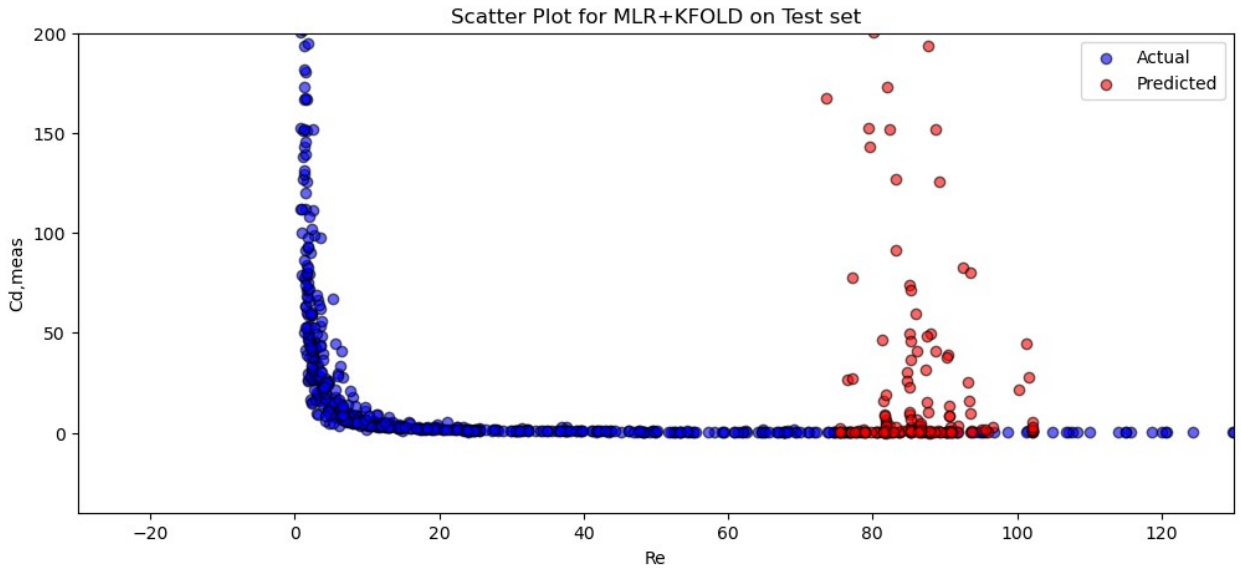
```
from sklearn.metrics import r2_score, mean_squared_error
print("R2 Score: ", r2_score(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("R2 Score  ", r2_score(y_train, y_pred_train))
print("Mean Squared Error  ", mean_squared_error(y_train,
y_pred_train))
```

```
R2 Score:  -0.0009731737028904064
Mean Squared Error:  27678.08957304943
```

```
R2 Score   0.009275364045496892
Mean Squared Error   60747.28971663838
```

```
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  , edgecolors='black',
alpha=0.6,label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' ,edgecolors='black',
alpha=0.6, label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 130)
plt.title('Scatter Plot for MLR+KFOLD on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1ec911d59f0>
```

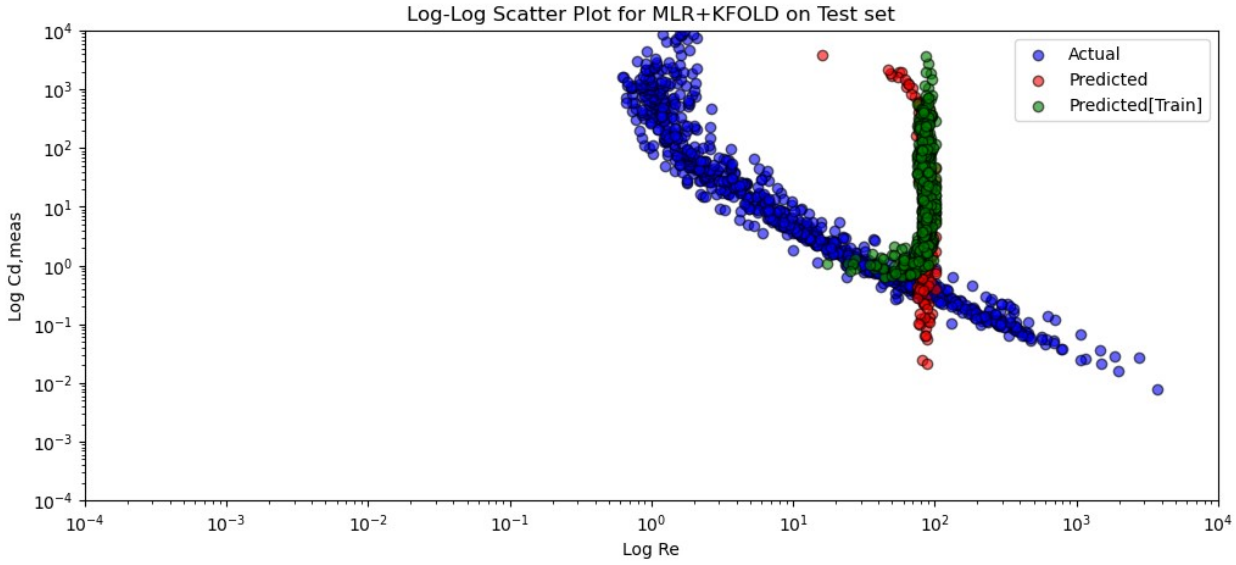



```
print("R2 Score: ", r2_score(y_pred,X_test[:,-1]))
```

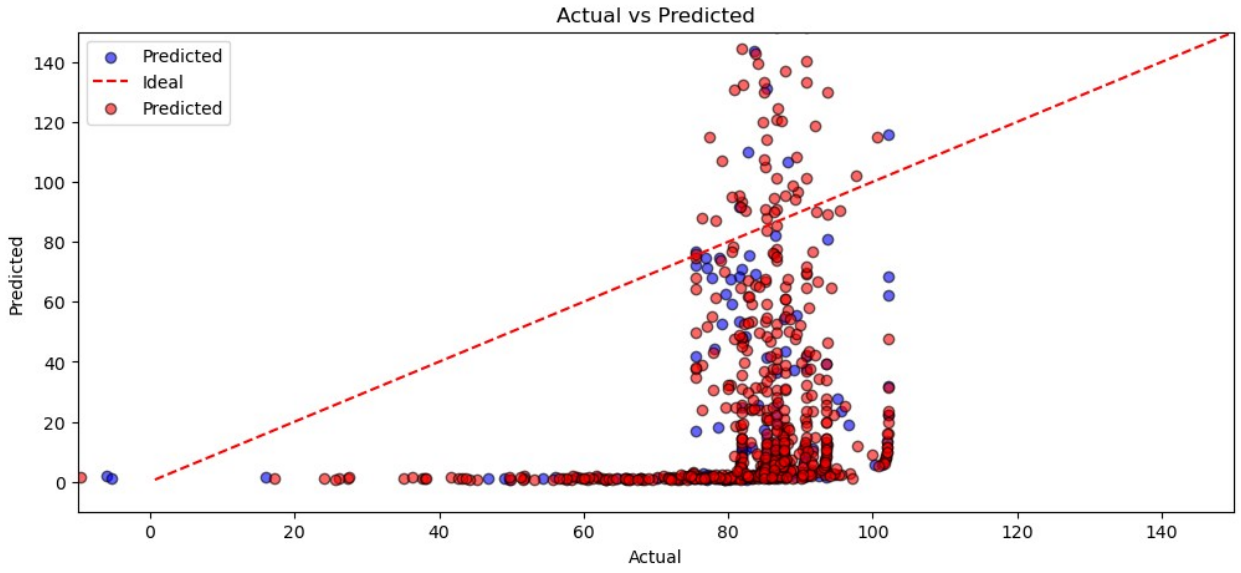
```
R2 Score:  -2543.968429306714
```

```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:,-1], c='blue', edgecolors='black',
alpha=0.6,label='Actual')
plt.scatter(y_pred, X_test[:,-1], c='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.scatter(y_pred_train,y_train, color='green',edgecolors='black',
alpha=0.6, label='Predicted[Train]')
```

```
plt.xscale('log')
plt.yscale('log')

plt.ylim(0.0001, 10000)
plt.xlim(0.0001, 10000)

plt.title('Log-Log Scatter Plot for MLR+KFOLD on Test set')
plt.ylabel('Log Cd,meas')
plt.xlabel('Log Re')
plt.legend()

plt.show()
```

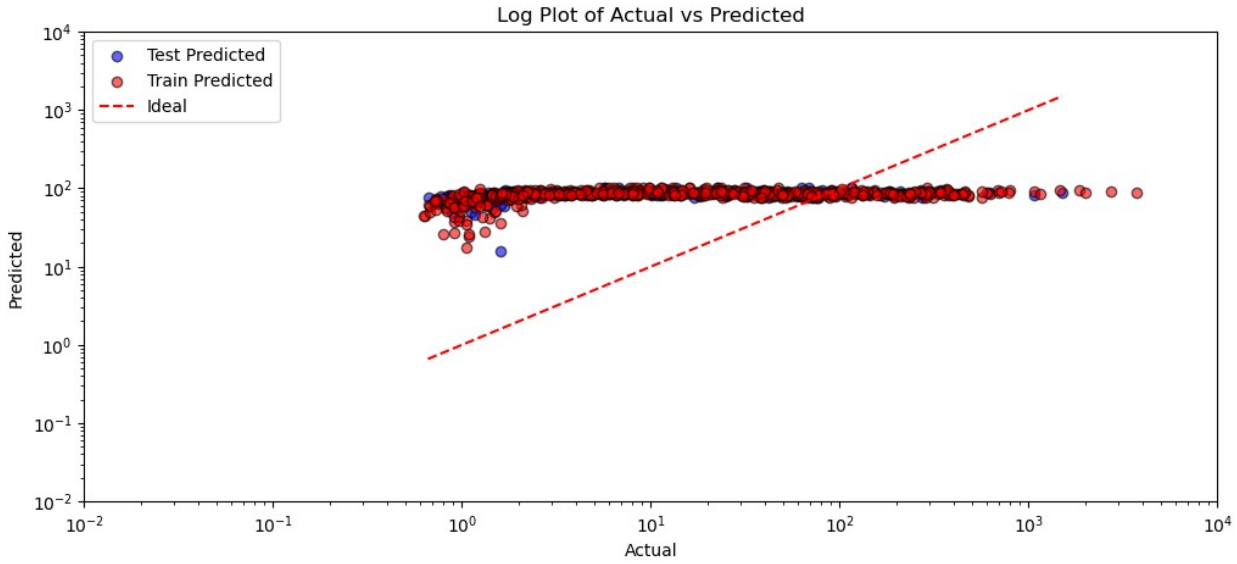



```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,
label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 150)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



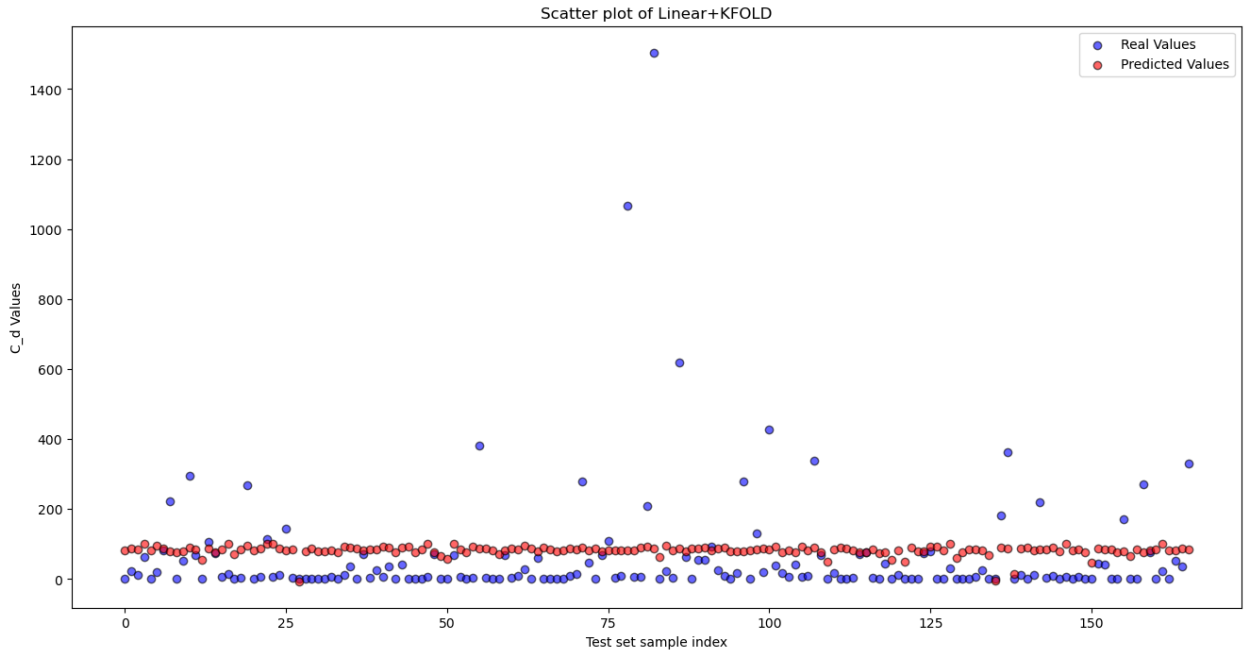

```
plt.figure(figsize=(12, 5))

plt.scatter(y_test, y_pred, color='blue', edgecolors='black',
alpha=0.6,label='Test Predicted')
plt.scatter(y_train, y_pred_train, color='red',edgecolors='black',
alpha=0.6, label='Train Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')

plt.xlim(0.01, 10000)
plt.ylim(0.01, 10000)

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.title('Log Plot of Actual vs Predicted')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1ec9169dff0>
```



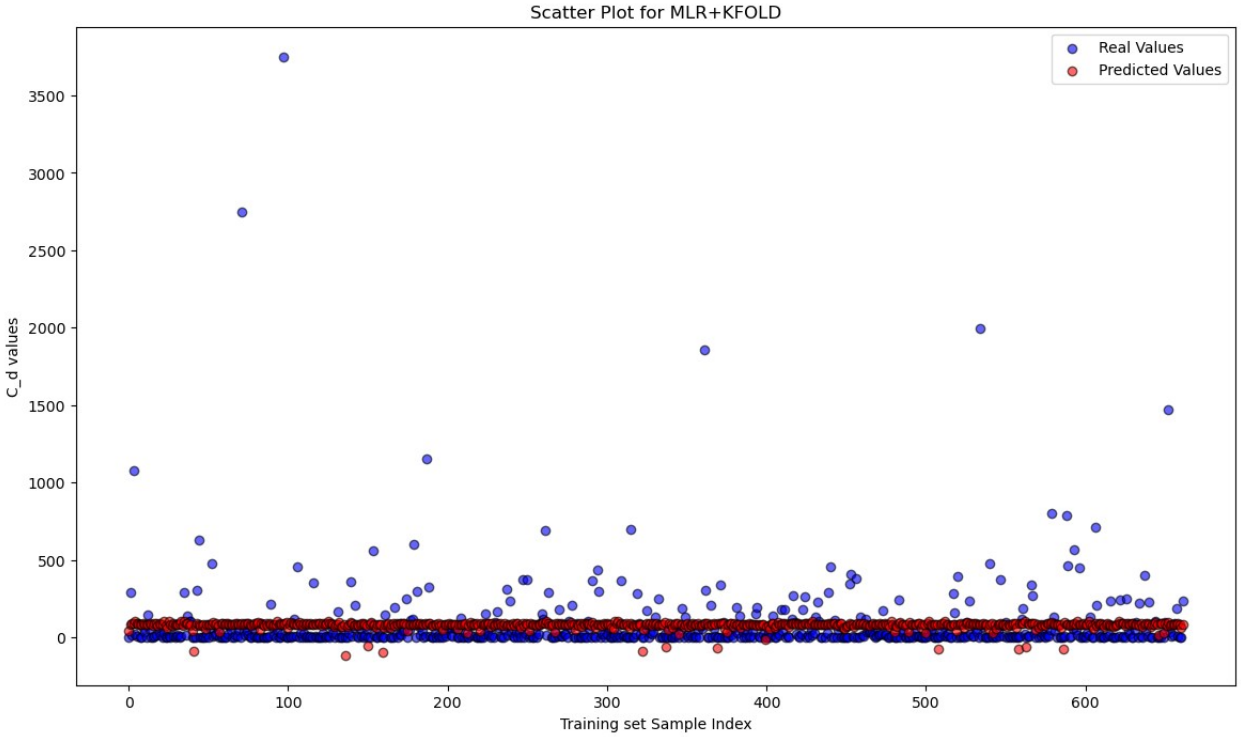
##Plotting the Results on Test set

```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter plot of Linear+KFOLD')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



##Scatter plot for Trainset

```
plt.figure(figsize=(14, 8))
plt.scatter(range(len(y_train)), y_train, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for MLR+KFOLD')
plt.xlabel('Training set Sample Index')
plt.ylabel('C_d values')
plt.legend()
plt.show()
```



Polynomial Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
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 = 0)
```

Training the Polynomial Regression model on the Training set

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 3)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)
```

```
LinearRegression()
```

Predicting the Test set results

```
y_pred = regressor.predict(poly_reg.transform(X_test))
y_pred_train = regressor.predict(poly_reg.transform(X_train))
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[ 6.96e+01  1.49e+00]
 [ 1.01e+02  2.20e+01]
 [ 9.44e+01  1.17e+01]
 [ 4.28e+01  6.23e+01]
 [ 7.31e+01  1.26e+00]
```

```
[ 1.15e+02  1.92e+01]
[ 1.01e+02  8.22e+01]
[ 7.94e+01  2.24e+02]
[ 3.79e+01  1.00e+00]
[ 7.95e+01  5.26e+01]
[ 1.18e+02  2.96e+02]
[ 8.79e+01  6.92e+01]
[-5.69e+01  1.45e+00]
[ 1.09e+02  1.07e+02]
[ 8.66e+01  7.48e+01]
[ 9.35e+01  5.42e+00]
[ 4.20e+01  1.36e+01]
[ 2.13e+01  1.03e+00]
[ 9.32e+01  3.84e+00]
[ 1.19e+02  2.69e+02]
[ 6.89e+01  8.61e-01]
[ 1.06e+02  6.20e+00]
[ 4.29e+01  1.16e+02]
[ 6.80e+01  5.61e+00]
[ 1.01e+02  1.26e+01]
[ 8.69e+01  1.44e+02]
[ 9.34e+01  4.95e+00]
[ 1.98e+01  1.84e+00]
[ 6.48e+01  1.29e+00]
[ 1.06e+02  2.06e+00]
[ 6.40e+01  8.64e-01]
[ 5.37e+01  1.17e+00]
[ 8.08e+01  7.14e+00]
[ 8.28e+01  2.57e+00]
[ 1.22e+02  1.12e+01]
[ 1.13e+02  3.72e+01]
[ 1.04e+02  2.56e+00]
[ 8.15e+01  7.10e+01]
[ 9.17e+01  3.12e+00]
[ 8.93e+01  2.55e+01]
[ 1.22e+02  5.73e+00]
[ 1.19e+02  3.76e+01]
[ 4.39e+01  1.18e+00]
[ 1.20e+02  4.18e+01]
[ 1.11e+02  1.88e+00]
[ 4.25e+01  1.10e+00]
[ 9.11e+01  1.51e+00]
[ 3.55e+01  5.70e+00]
[ 9.66e+01  7.21e+01]
[-1.01e+01  1.16e+00]
[-9.98e+01  1.67e+00]
[ 4.28e+01  6.83e+01]
[ 9.83e+01  6.83e+00]
[ 7.05e+01  1.29e+00]
```

```
[ 1.20e+02  4.57e+00]
[ 1.02e+02  3.81e+02]
[ 1.04e+02  3.65e+00]
[ 8.02e+01  1.23e+00]
[ 7.04e+00  1.19e+00]
[ 8.08e+01  6.83e+01]
[ 1.05e+02  4.64e+00]
[ 9.45e+01  9.92e+00]
[ 1.22e+02  2.76e+01]
[ 7.75e+01  1.02e+00]
[ 7.91e+01  5.94e+01]
[ 1.15e+02  1.66e+00]
[ 9.24e+01  1.71e+00]
[ 5.79e+01  7.77e-01]
[ 7.56e+01  1.31e+00]
[ 1.01e+02  1.05e+01]
[ 9.46e+01  1.49e+01]
[ 1.18e+02  2.80e+02]
[ 8.30e+01  4.83e+01]
[ 9.48e+01  1.37e+00]
[ 7.91e+01  6.77e+01]
[ 8.40e+01  1.10e+02]
[ 7.94e+01  3.52e+00]
[ 8.11e+01  1.00e+01]
[ 8.15e+01  1.07e+03]
[ 8.07e+01  6.55e+00]
[ 1.18e+02  6.39e+00]
[ 1.22e+02  2.09e+02]
[ 1.07e+02  1.50e+03]
[-1.29e+01  8.60e-01]
[ 1.20e+02  2.37e+01]
[ 8.03e+01  4.89e+00]
[ 1.06e+02  6.19e+02]
[ 7.90e+01  6.27e+01]
[ 1.01e+02  2.61e+00]
[ 1.07e+02  5.43e+01]
[ 1.15e+02  5.55e+01]
[ 8.07e+01  9.17e+01]
[ 1.01e+02  2.56e+01]
[ 1.19e+02  8.60e+00]
[ 6.15e+01  9.80e-01]
[ 8.00e+01  1.82e+01]
[ 7.98e+01  2.79e+02]
[ 7.34e+01  1.26e+00]
[ 9.48e+01  1.31e+02]
[ 1.02e+02  1.88e+01]
[ 9.48e+01  4.28e+02]
[ 1.24e+02  3.95e+01]
[ 9.64e+01  1.69e+01]
```

```
[ 8.04e+01  5.57e+00]
[ 9.66e+01  4.17e+01]
[ 1.23e+02  7.50e+00]
[ 8.28e+01  1.07e+01]
[ 1.20e+02  3.39e+02]
[ 8.27e+01  6.78e+01]
[-4.48e+01  8.81e-01]
[ 9.31e+01  1.74e+01]
[ 3.85e+00  2.66e+00]
[ 1.09e+02  2.54e+00]
[ 7.98e+01  3.31e+00]
[ 8.61e+01  7.12e+01]
[ 9.66e+01  7.68e+01]
[ 9.06e+01  3.72e+00]
[ 6.61e+01  1.45e+00]
[ 8.17e+01  4.43e+01]
[-3.34e+01  9.68e-01]
[ 8.11e+01  1.11e+01]
[-4.04e+01  1.12e+00]
[ 1.15e+02  1.97e+00]
[ 5.51e+01  1.07e+00]
[ 8.01e+01  7.45e+01]
[ 1.24e+02  8.09e+01]
[ 1.10e+02  1.70e+00]
[ 8.11e+01  1.47e+00]
[ 4.26e+01  3.21e+01]
[-1.91e+01  1.61e+00]
[ 4.12e+01  6.60e-01]
[ 9.13e+01  1.79e+00]
[ 9.39e+01  6.70e+00]
[ 8.14e+01  2.48e+01]
[ 3.48e+00  1.19e+00]
[-5.13e+00  9.35e-01]
[ 1.20e+02  1.83e+02]
[ 1.11e+02  3.64e+02]
[-4.25e+01  1.60e+00]
[ 1.01e+02  1.10e+01]
[ 1.04e+02  1.64e+00]
[ 8.58e+01  1.15e+01]
[ 9.48e+01  2.20e+02]
[ 9.26e+01  4.15e+00]
[ 1.19e+02  8.61e+00]
[ 6.40e+01  8.28e-01]
[ 3.83e+01  6.73e+00]
[ 7.09e+01  1.46e+00]
[ 9.71e+01  6.53e+00]
[ 7.97e+01  2.16e+00]
[-4.35e+01  1.18e+00]
[ 1.07e+02  4.36e+01]
```

```
 [ 9.47e+01  4.13e+01]
 [ 9.55e+01  2.20e+00]
 [ 5.48e+01  1.33e+00]
 [ 7.99e+01  1.72e+02]
 [-5.82e+00  9.83e-01]
 [ 7.96e+01  1.28e+00]
 [ 8.42e+01  2.71e+02]
 [ 8.43e+01  7.56e+01]
 [ 9.31e+01  2.09e+00]
 [ 4.24e+01  2.25e+01]
 [ 7.65e+01  1.98e+00]
 [ 8.06e+01  5.34e+01]
 [ 1.01e+02  3.66e+01]
 [ 9.93e+01  3.31e+02]]
```

Evaluating the Model Performance

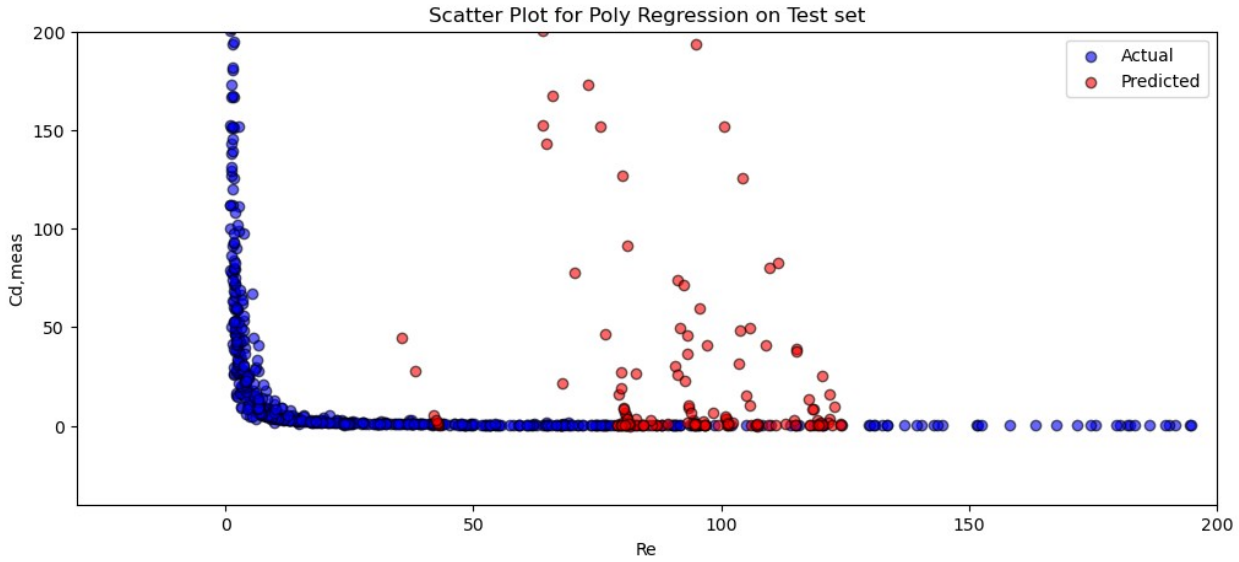
```
from sklearn.metrics import r2_score, mean_squared_error
print('R_2 Score :', r2_score(y_test, y_pred))
print('Mean Squared Error :', mean_squared_error(y_test, y_pred))
print('MSE for Training set:', mean_squared_error(y_train,
y_pred_train))
print('R_2 score for Training set:',r2_score(y_train, y_pred_train))
```

```
R_2 Score : 0.024932545519315563
Mean Squared Error : 26961.765863359975
MSE for Training set: 59503.891747171154
R_2 score for Training set: 0.0295538819249469
```

##Plotting Results

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6,
label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' , edgecolors='black',
alpha=0.6,label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for Poly Regression on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

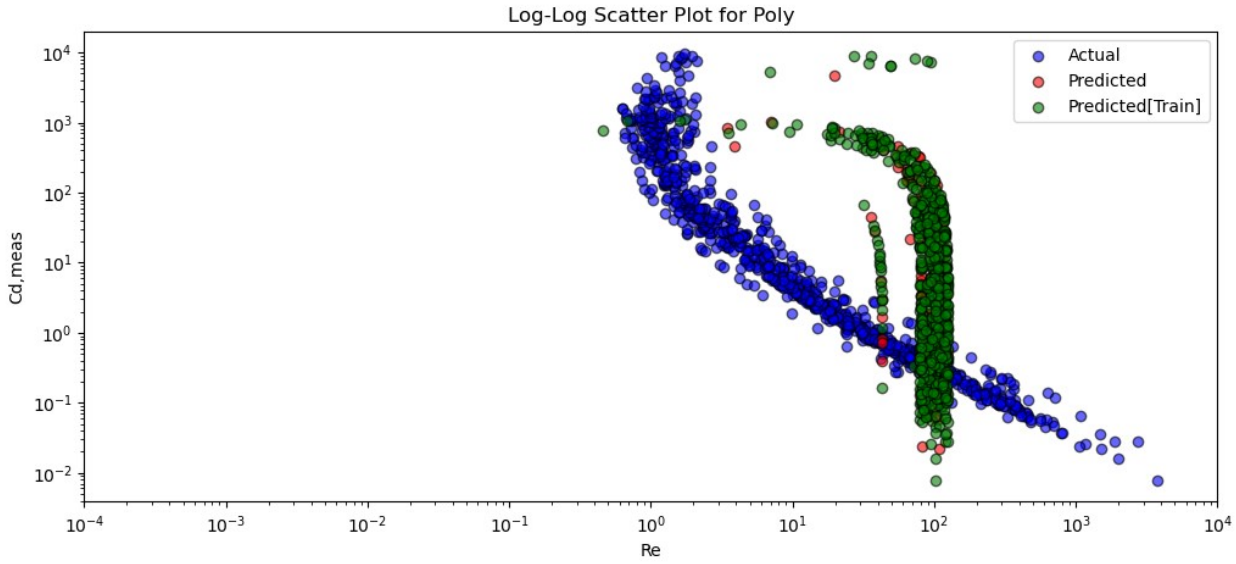
```
<matplotlib.legend.Legend at 0x1d3aa2aafe0>
```



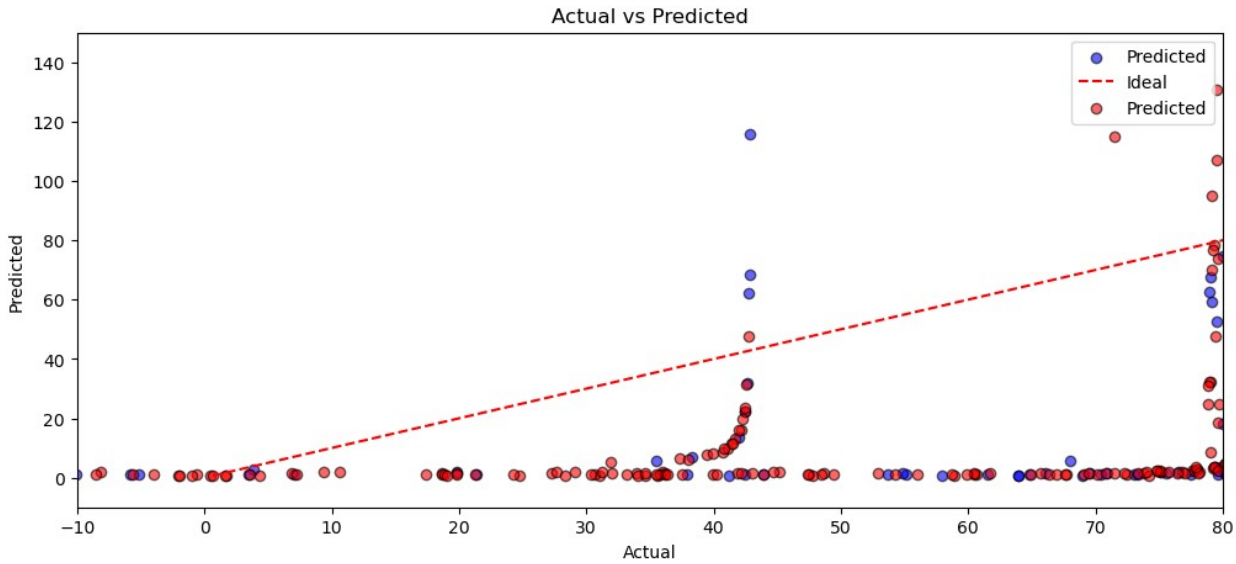

```
print('R_2 Score :', r2_score(y_pred, X_test[:,-1]))
```

```
R_2 Score : -343.4501971581916
```

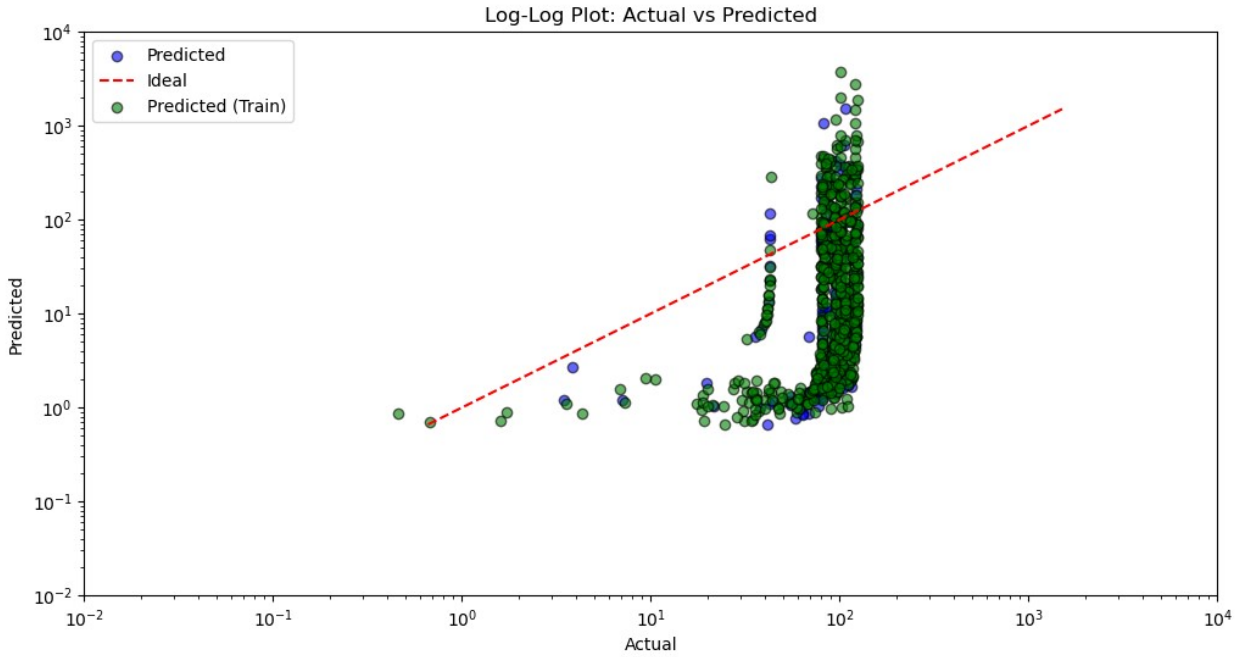
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6,
label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c =
'green' ,edgecolors='black', alpha=0.6, label = 'Predicted[Train]')
plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)
plt.title('Log-Log Scatter Plot for Poly')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,
label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 80)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```

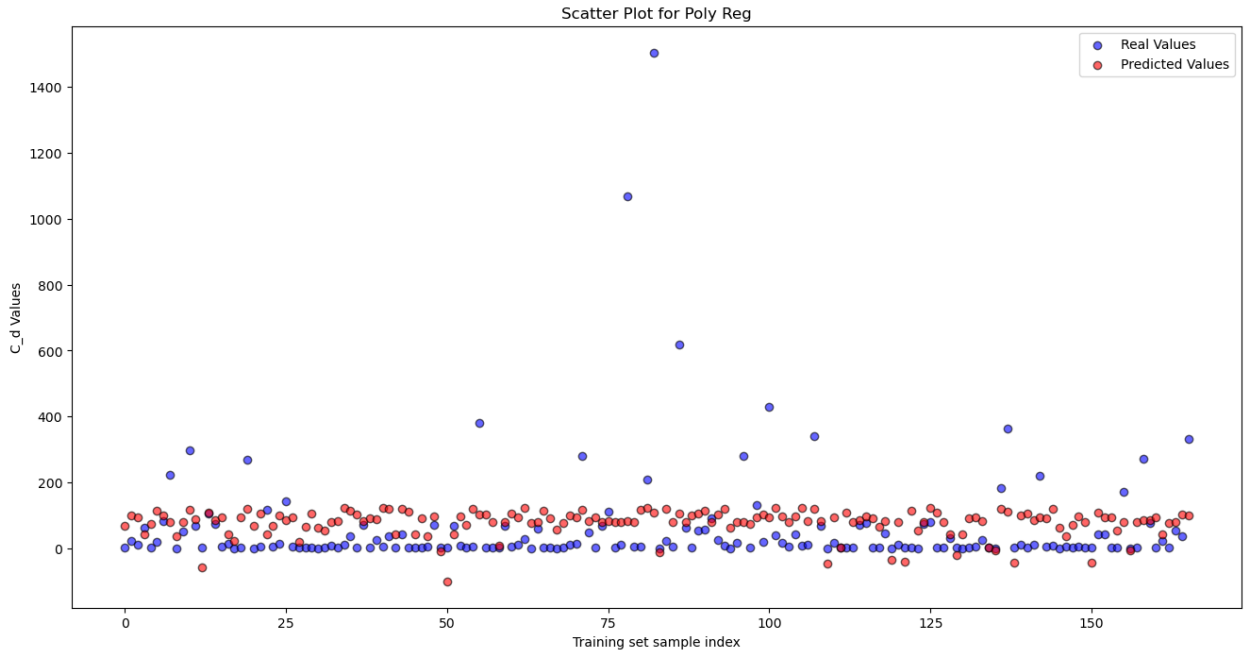


```
plt.figure(figsize=(12, 6))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black',
alpha=0.6, label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black',
alpha=0.6, label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.01,10000)
plt.xlim(0.01,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```

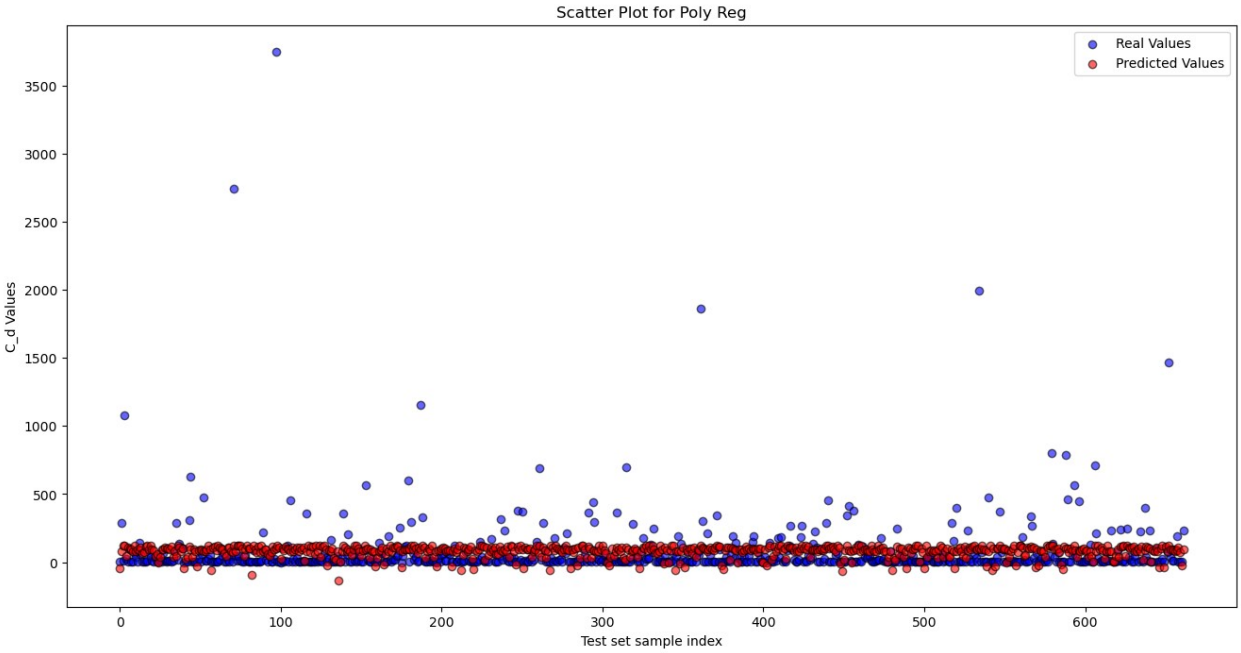



```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for Poly Reg')
plt.xlabel('Training set sample index')
```

```
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_train)), y_train, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for Poly Reg')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```


Scatter Plot for Poly Reg
Real Values
Predicted Values
C_d Values
Test set sample index
0
100
200
300
400
500
600
0
500
1000
1500
2000
2500
3000
3500

Polynomial Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

##Splitting the Dataset

```
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 = 0)
```

Training the Polynomial Regression model on the Training Set

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)

LinearRegression()
```

##Applying KFOLD

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor, X = X_train, y =
y_train, cv = 3)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

Accuracy: 1.00 %
Standard Deviation: 0.60 %
```

##Predicting the test set Results

```
y_pred = regressor.predict(poly_reg.transform(X_test))
y_pred_train = regressor.predict(poly_reg.transform(X_train))
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[ 5.55e+01  1.49e+00]
 [ 1.04e+02  2.20e+01]
 [ 1.02e+02  1.17e+01]
 [ 1.17e+02  6.23e+01]
 [ 6.39e+01  1.26e+00]
 [ 1.13e+02  1.92e+01]
 [ 1.04e+02  8.22e+01]
 [ 9.57e+01  2.24e+02]
 [-3.70e+00  1.00e+00]
 [ 9.28e+01  5.26e+01]
 [ 1.08e+02  2.96e+02]
 [ 1.00e+02  6.92e+01]
 [-3.38e+01  1.45e+00]
 [ 1.06e+02  1.07e+02]
 [ 8.90e+01  7.48e+01]
 [ 1.00e+02  5.42e+00]
 [ 1.16e+02  1.36e+01]
 [-1.75e+01  1.03e+00]
 [ 9.46e+01  3.84e+00]
 [ 1.13e+02  2.69e+02]
 [ 3.14e+01  8.61e-01]
 [ 1.03e+02  6.20e+00]
 [ 1.17e+02  1.16e+02]
 [ 1.11e+02  5.61e+00]
 [ 1.03e+02  1.26e+01]
 [ 1.00e+02  1.44e+02]
 [ 1.00e+02  4.95e+00]
 [ 1.10e+02  1.84e+00]
 [ 6.69e+01  1.29e+00]
 [ 9.48e+01  2.06e+00]
 [ 6.52e+01  8.64e-01]
 [ 3.45e+01  1.17e+00]
 [ 9.63e+01  7.14e+00]
 [ 8.34e+01  2.57e+00]
 [ 1.09e+02  1.12e+01]
 [ 1.07e+02  3.72e+01]
 [ 9.80e+01  2.56e+00]
 [ 9.76e+01  7.10e+01]
 [ 9.16e+01  3.12e+00]
 [ 1.01e+02  2.55e+01]
 [ 1.07e+02  5.73e+00]
 [ 1.08e+02  3.76e+01]
 [ 2.17e+01  1.18e+00]
 [ 1.09e+02  4.18e+01]
```

```
[ 9.16e+01  1.88e+00]
[ 8.07e+00  1.10e+00]
[ 8.65e+01  1.51e+00]
[ 1.06e+02  5.70e+00]
[ 8.62e+01  7.21e+01]
[-4.44e+01  1.16e+00]
[-3.07e+01  1.67e+00]
[ 1.17e+02  6.83e+01]
[ 1.02e+02  6.83e+00]
[ 7.54e+01  1.29e+00]
[ 1.05e+02  4.57e+00]
[ 1.04e+02  3.81e+02]
[ 9.46e+01  3.65e+00]
[ 7.39e+01  1.23e+00]
[-3.68e+01  1.19e+00]
[ 9.71e+01  6.83e+01]
[ 1.02e+02  4.64e+00]
[ 1.02e+02  9.92e+00]
[ 1.12e+02  2.76e+01]
[ 3.96e+01  1.02e+00]
[ 9.52e+01  5.94e+01]
[ 9.96e+01  1.66e+00]
[ 8.73e+01  1.71e+00]
[ 3.92e+01  7.77e-01]
[ 6.83e+01  1.31e+00]
[ 1.03e+02  1.05e+01]
[ 1.02e+02  1.49e+01]
[ 1.08e+02  2.80e+02]
[ 9.84e+01  4.83e+01]
[ 6.51e+01  1.37e+00]
[ 9.51e+01  6.77e+01]
[ 9.89e+01  1.10e+02]
[ 9.38e+01  3.52e+00]
[ 9.68e+01  1.00e+01]
[ 9.76e+01  1.07e+03]
[ 9.61e+01  6.55e+00]
[ 1.05e+02  6.39e+00]
[ 1.10e+02  2.09e+02]
[ 1.06e+02  1.50e+03]
[-3.80e+01  8.60e-01]
[ 1.13e+02  2.37e+01]
[ 9.55e+01  4.89e+00]
[ 1.05e+02  6.19e+02]
[ 9.40e+01  6.27e+01]
[ 7.40e+01  2.61e+00]
[ 1.05e+02  5.43e+01]
[ 1.07e+02  5.55e+01]
[ 9.70e+01  9.17e+01]
[ 1.04e+02  2.56e+01]
```

```
[ 1.07e+02  8.60e+00]
[ 4.64e+01  9.80e-01]
[ 9.20e+01  1.82e+01]
[ 9.27e+01  2.79e+02]
[ 5.82e+01  1.26e+00]
[ 1.03e+02  1.31e+02]
[ 1.04e+02  1.88e+01]
[ 1.03e+02  4.28e+02]
[ 1.11e+02  3.95e+01]
[ 8.58e+01  1.69e+01]
[ 9.56e+01  5.57e+00]
[ 8.61e+01  4.17e+01]
[ 1.09e+02  7.50e+00]
[ 9.77e+01  1.07e+01]
[ 1.09e+02  3.39e+02]
[ 9.06e+01  6.78e+01]
[-3.32e+01  8.81e-01]
[ 1.02e+02  1.74e+01]
[ 2.42e+01  2.66e+00]
[ 9.74e+01  2.54e+00]
[ 9.36e+01  3.31e+00]
[ 8.92e+01  7.12e+01]
[ 8.62e+01  7.68e+01]
[ 9.52e+01  3.72e+00]
[ 5.56e+01  1.45e+00]
[ 9.11e+01  4.43e+01]
[-4.02e+01  9.68e-01]
[ 9.69e+01  1.11e+01]
[-3.08e+01  1.12e+00]
[ 9.98e+01  1.97e+00]
[ 1.64e+01  1.07e+00]
[ 9.23e+01  7.45e+01]
[ 1.11e+02  8.09e+01]
[ 9.31e+01  1.70e+00]
[ 8.08e+01  1.47e+00]
[ 1.17e+02  3.21e+01]
[-4.55e+01  1.61e+00]
[-2.09e+00  6.60e-01]
[ 9.63e+01  1.79e+00]
[ 1.01e+02  6.70e+00]
[ 9.74e+01  2.48e+01]
[-2.37e+01  1.19e+00]
[ 1.22e+02  9.35e-01]
[ 1.09e+02  1.83e+02]
[ 1.07e+02  3.64e+02]
[ 8.18e+01  1.60e+00]
[ 1.03e+02  1.10e+01]
[ 7.98e+01  1.64e+00]
[ 9.91e+01  1.15e+01]
```

```
 [ 1.03e+02  2.20e+02]
 [ 9.72e+01  4.15e+00]
 [ 1.07e+02  8.61e+00]
 [ 5.73e+01  8.28e-01]
 [ 1.10e+02  6.73e+00]
 [ 5.49e+01  1.46e+00]
 [ 9.46e+01  6.53e+00]
 [ 8.44e+01  2.16e+00]
 [-1.99e+01  1.18e+00]
 [ 1.06e+02  4.36e+01]
 [ 1.02e+02  4.13e+01]
 [ 9.05e+01  2.20e+00]
 [ 4.96e+01  1.33e+00]
 [ 9.26e+01  1.72e+02]
 [-4.31e+01  9.83e-01]
 [ 4.19e+01  1.28e+00]
 [ 9.00e+01  2.71e+02]
 [ 9.90e+01  7.56e+01]
 [ 9.26e+01  2.09e+00]
 [ 1.17e+02  2.25e+01]
 [ 8.75e+01  1.98e+00]
 [ 9.69e+01  5.34e+01]
 [ 1.04e+02  3.66e+01]
 [ 1.04e+02  3.31e+02]]
```

##Evaluating the Model Performance

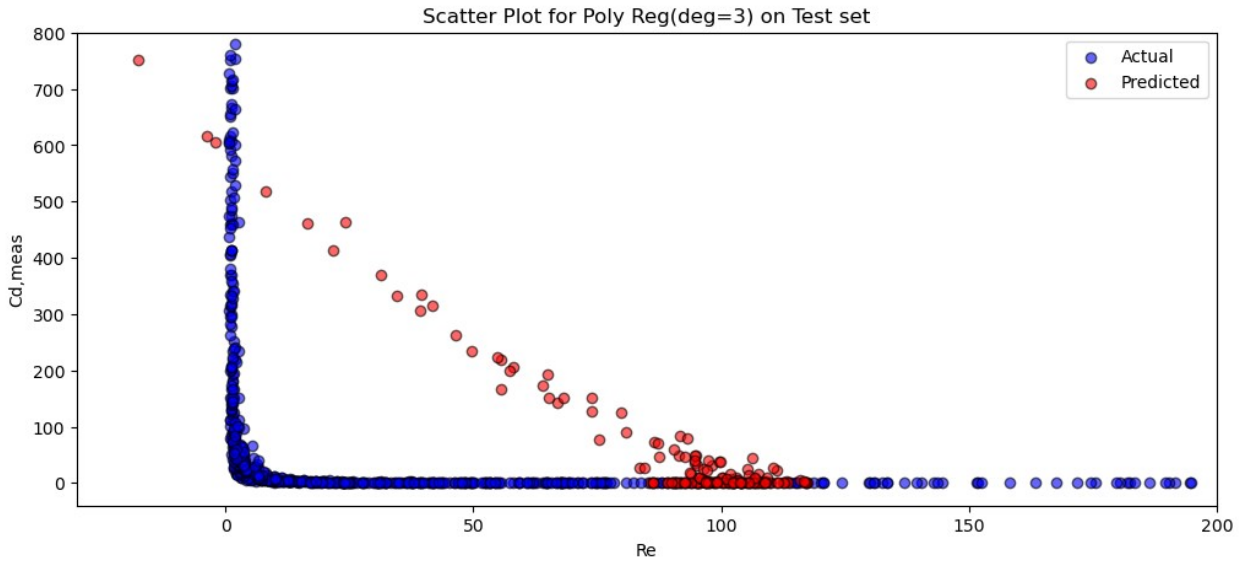
```
from sklearn.metrics import r2_score, mean_squared_error
print("R2 Score: ", r2_score(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_train, y_pred_train))
print("Mean Squared Error:", mean_squared_error(y_train,
y_pred_train))
```

```
R2 Score:  0.026957553642304233
Mean Squared Error:  26905.771998902153
R2 Score: 0.03105021281451692
Mean Squared Error: 59412.142695253126
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6,
label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' ,edgecolors='black',
alpha=0.6, label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 800)
plt.xlim(-30, 200)
```

```
plt.title('Scatter Plot for Poly Reg(deg=3) on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x154113a7010>
```

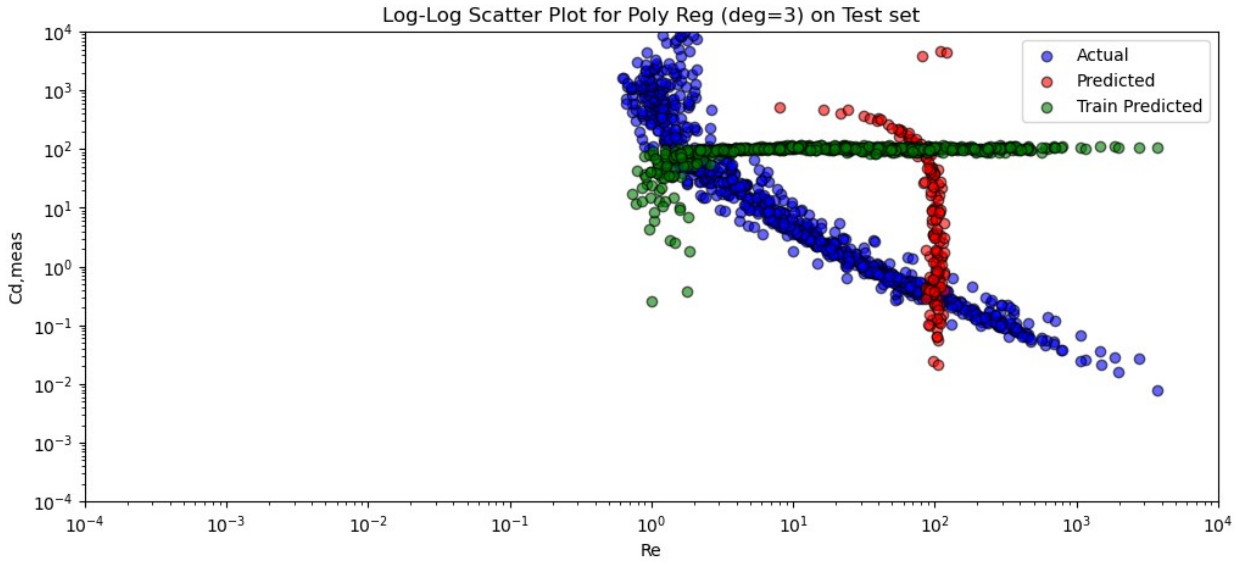



```
print("R2 Score: ", r2_score(y_pred ,X_test[:,-1]))
```

```
R2 Score:  -328.33131102244744
```

```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:,-1], c='blue',edgecolors='black', alpha=0.6,
label='Actual')
plt.scatter(y_pred, X_test[:,-1], c='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.scatter(y_train, y_pred_train, color='green',edgecolors='black',
alpha=0.6, label='Train Predicted')


plt.ylim(0.0001,10000)
plt.xlim(0.0001,10000)

plt.xscale('log')
plt.yscale('log')

plt.title('Log-Log Scatter Plot for Poly Reg (deg=3) on Test set')
plt.xlabel('Re')
plt.ylabel('Cd,meas')
plt.legend()
```

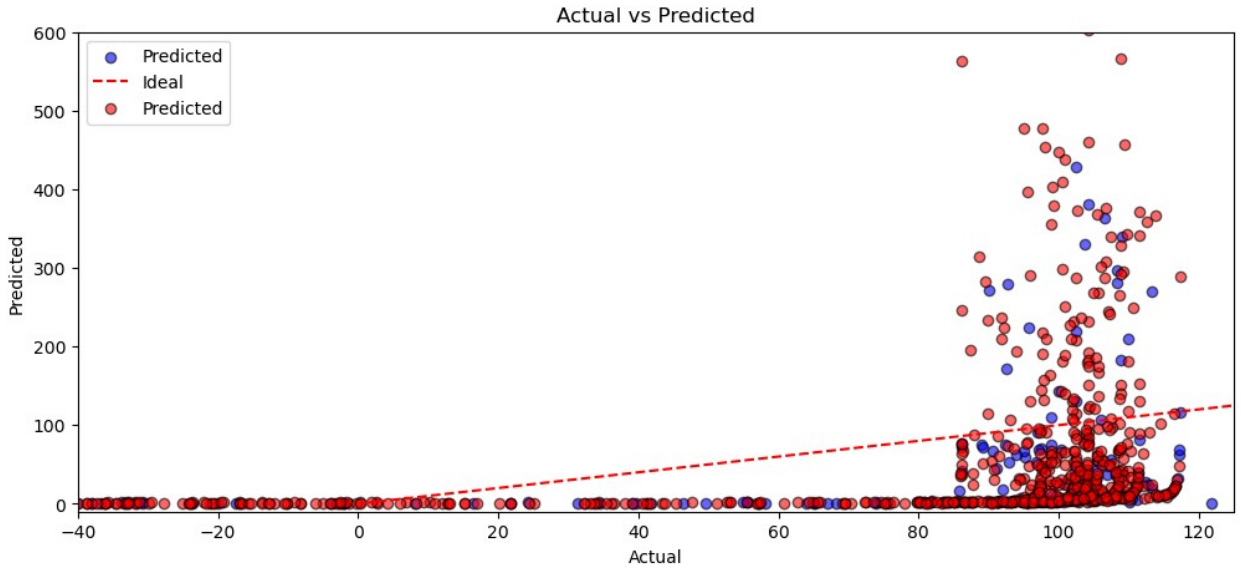
```
plt.show()
```



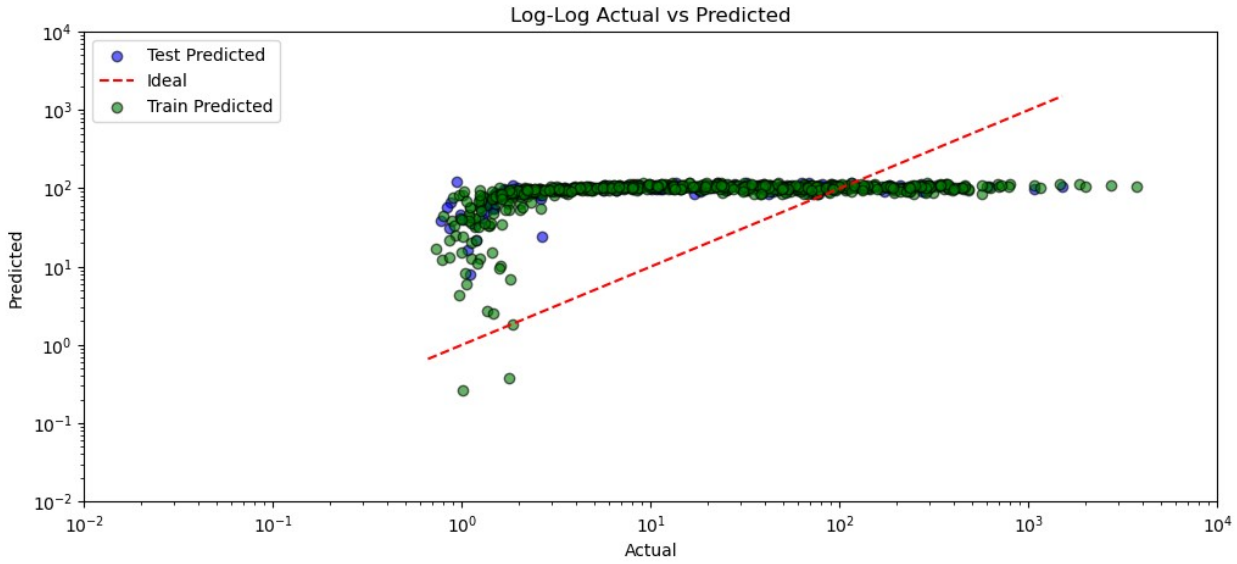

```
print("R2 Score: ", r2_score(y_pred_train ,X_train[:,-1]))
```

```
R2 Score:  -744.5051734278397
```

```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,
label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-40, 125)
plt.ylim(-10, 600)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```

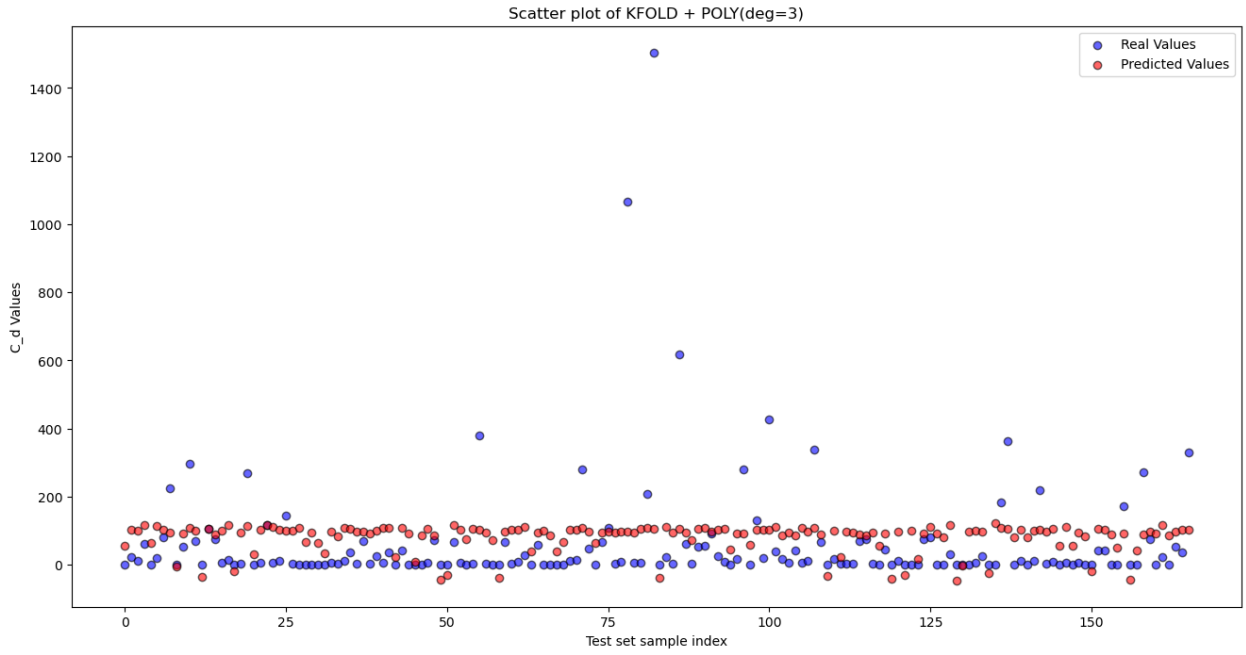



```
plt.figure(figsize=(12, 5))


plt.scatter(y_test, y_pred, color='blue',edgecolors='black',
alpha=0.6, label='Test Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_train, y_pred_train, color='green',edgecolors='black',
alpha=0.6, label='Train Predicted')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0.01, 10000)
plt.ylim(0.01, 10000)

plt.xscale('log')
plt.yscale('log')

plt.title('Log-Log Actual vs Predicted')
plt.legend()
plt.show()
```



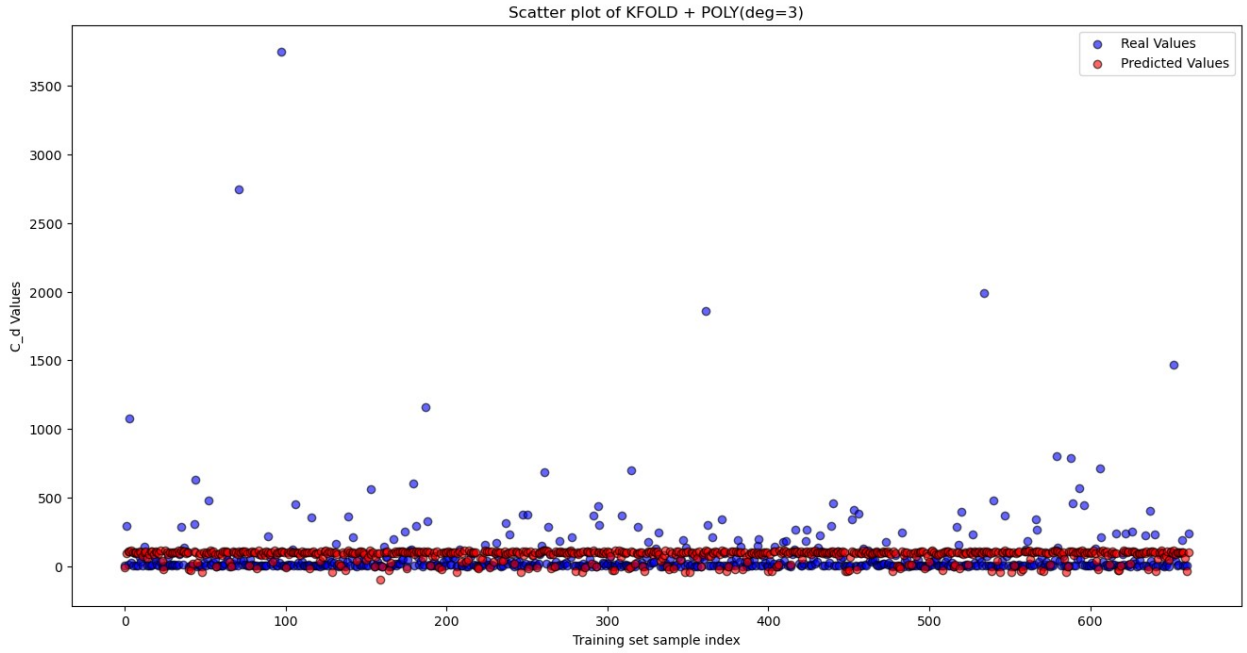
Visualising the Polynomial Regression results

```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter plot of KFOLD + POLY(deg=3)')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



##Scatter plot for Train set

```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_train)), y_train, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter plot of KFOLD + POLY(deg=3)')
plt.xlabel('Training set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



Decision Tree Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
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 = 0)
```

Training the Decision Tree Regression model on the Training set

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

Predicting the Test set results

```
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[1.22e+00 1.49e+00]
 [1.78e+01 2.20e+01]
 [1.15e+01 1.17e+01]
 [4.63e+01 6.23e+01]
 [1.71e+00 1.26e+00]
 [1.96e+01 1.92e+01]
 [1.21e+02 8.22e+01]
 [2.09e+02 2.24e+02]
```

```
[1.22e+00 1.00e+00]
[4.98e+01 5.26e+01]
[3.28e+02 2.96e+02]
[5.91e+01 6.92e+01]
[1.95e+00 1.45e+00]
[5.81e+01 1.07e+02]
[9.04e+01 7.48e+01]
[3.04e+00 5.42e+00]
[1.59e+01 1.36e+01]
[1.57e+00 1.03e+00]
[3.64e+00 3.84e+00]
[3.59e+02 2.69e+02]
[1.22e+00 8.61e-01]
[6.56e+00 6.20e+00]
[1.15e+02 1.16e+02]
[7.69e+00 5.61e+00]
[1.25e+01 1.26e+01]
[1.33e+02 1.44e+02]
[4.67e+00 4.95e+00]
[2.10e+00 1.84e+00]
[9.21e-01 1.29e+00]
[2.75e+00 2.06e+00]
[9.21e-01 8.64e-01]
[1.39e+00 1.17e+00]
[8.01e+00 7.14e+00]
[2.67e+00 2.57e+00]
[1.46e+01 1.12e+01]
[4.95e+01 3.72e+01]
[3.23e+00 2.56e+00]
[7.84e+01 7.10e+01]
[2.73e+00 3.12e+00]
[2.56e+01 2.55e+01]
[5.65e+00 5.73e+00]
[1.84e+01 3.76e+01]
[1.19e+00 1.18e+00]
[3.62e+01 4.18e+01]
[1.77e+00 1.88e+00]
[1.80e+00 1.10e+00]
[2.01e+00 1.51e+00]
[6.38e+00 5.70e+00]
[6.41e+01 7.21e+01]
[2.04e+00 1.16e+00]
[1.24e+00 1.67e+00]
[1.33e+02 6.83e+01]
[7.12e+00 6.83e+00]
[1.50e+00 1.29e+00]
[4.67e+00 4.57e+00]
[3.68e+02 3.81e+02]
[2.75e+00 3.65e+00]
```

```
[1.22e+00 1.23e+00]
[7.11e-01 1.19e+00]
[6.56e+01 6.83e+01]
[5.02e+00 4.64e+00]
[1.78e+01 9.92e+00]
[6.49e+01 2.76e+01]
[2.02e+00 1.02e+00]
[7.00e+01 5.94e+01]
[1.25e+00 1.66e+00]
[2.01e+00 1.71e+00]
[9.88e-01 7.77e-01]
[1.22e+00 1.31e+00]
[1.25e+01 1.05e+01]
[2.44e+01 1.49e+01]
[4.57e+02 2.80e+02]
[4.81e+01 4.83e+01]
[1.77e+00 1.37e+00]
[7.00e+01 6.77e+01]
[1.33e+02 1.10e+02]
[3.46e+00 3.52e+00]
[8.54e+00 1.00e+01]
[1.16e+03 1.07e+03]
[8.01e+00 6.55e+00]
[7.39e+00 6.39e+00]
[3.41e+02 2.09e+02]
[1.99e+03 1.50e+03]
[9.94e-01 8.60e-01]
[2.75e+01 2.37e+01]
[4.93e+00 4.89e+00]
[6.96e+02 6.19e+02]
[7.38e+01 6.27e+01]
[1.62e+00 2.61e+00]
[5.73e+01 5.43e+01]
[5.73e+01 5.55e+01]
[7.60e+01 9.17e+01]
[6.49e+01 2.56e+01]
[8.66e+00 8.60e+00]
[1.11e+00 9.80e-01]
[1.91e+01 1.82e+01]
[2.36e+02 2.79e+02]
[9.68e-01 1.26e+00]
[1.33e+02 1.31e+02]
[2.09e+01 1.88e+01]
[4.60e+02 4.28e+02]
[3.95e+01 3.95e+01]
[1.78e+01 1.69e+01]
[4.30e+00 5.57e+00]
[6.41e+01 4.17e+01]
[6.89e+00 7.50e+00]
```

```
[8.54e+00 1.07e+01]
[4.57e+02 3.39e+02]
[5.52e+01 6.78e+01]
[8.17e-01 8.81e-01]
[1.78e+01 1.74e+01]
[1.24e+00 2.66e+00]
[3.63e+00 2.54e+00]
[3.19e+00 3.31e+00]
[7.60e+01 7.12e+01]
[1.07e+02 7.68e+01]
[3.63e+00 3.72e+00]
[1.06e+00 1.45e+00]
[4.29e+01 4.43e+01]
[1.49e+00 9.68e-01]
[1.15e+01 1.11e+01]
[9.06e-01 1.12e+00]
[1.25e+00 1.97e+00]
[1.13e+00 1.07e+00]
[7.38e+01 7.45e+01]
[4.34e+01 8.09e+01]
[1.97e+00 1.70e+00]
[1.78e+00 1.47e+00]
[2.75e+01 3.21e+01]
[1.50e+00 1.61e+00]
[1.22e+00 6.60e-01]
[3.26e+00 1.79e+00]
[4.18e+00 6.70e+00]
[2.44e+01 2.48e+01]
[1.10e+00 1.19e+00]
[1.06e+00 9.35e-01]
[9.41e+01 1.83e+02]
[2.44e+02 3.64e+02]
[1.62e+00 1.60e+00]
[1.08e+01 1.10e+01]
[1.01e+00 1.64e+00]
[1.15e+01 1.15e+01]
[2.32e+02 2.20e+02]
[3.93e+00 4.15e+00]
[1.02e+01 8.61e+00]
[1.24e+00 8.28e-01]
[6.01e+00 6.73e+00]
[9.68e-01 1.46e+00]
[3.64e+00 6.53e+00]
[2.65e+00 2.16e+00]
[1.95e+00 1.18e+00]
[3.89e+01 4.36e+01]
[4.85e+01 4.13e+01]
[3.50e+00 2.20e+00]
[8.51e-01 1.33e+00]
```

```
 [1.95e+02 1.72e+02]
 [2.04e+00 9.83e-01]
 [1.41e+00 1.28e+00]
 [2.36e+02 2.71e+02]
 [7.67e+01 7.56e+01]
 [2.73e+00 2.09e+00]
 [1.97e+01 2.25e+01]
 [1.83e+00 1.98e+00]
 [4.98e+01 5.34e+01]
 [4.31e+01 3.66e+01]
 [4.60e+02 3.31e+02]]
```

Evaluating the Model Performance

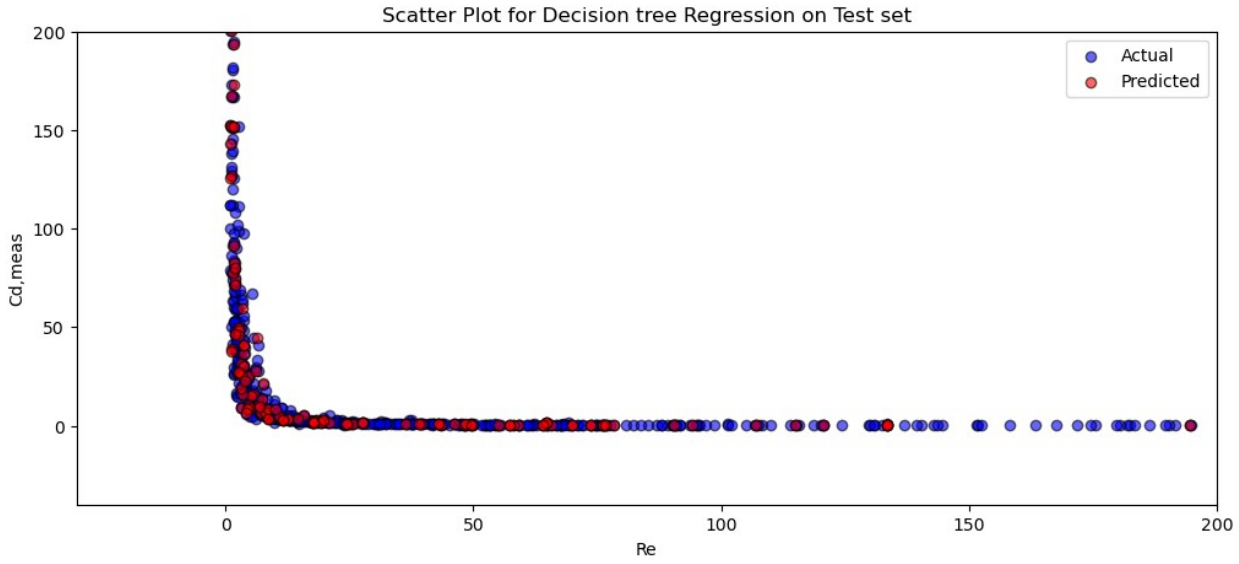
```
from sklearn.metrics import r2_score
print('R_2 Score :', r2_score(y_test, y_pred))
print('R_2 score for Training set:',r2_score(y_train, y_pred_train))
```

```
R_2 Score : 0.9161391752164351
R_2 score for Training set: 0.999999999541169
```

##Plotting Results

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6,
label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' , edgecolors='black',
alpha=0.6,label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for Decision tree Regression on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

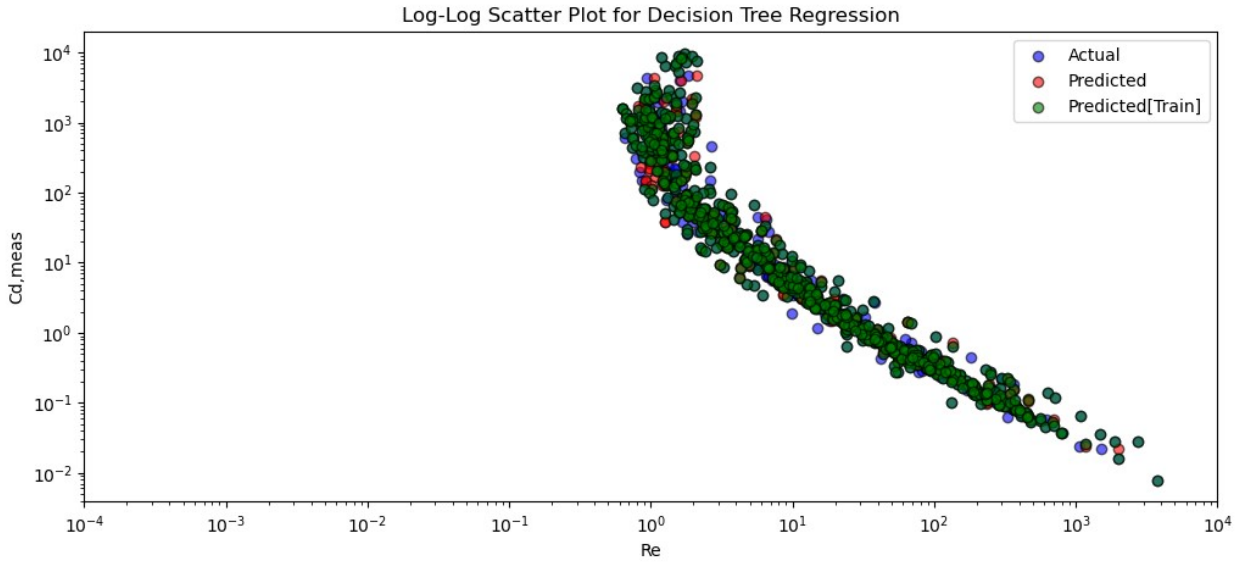
```
<matplotlib.legend.Legend at 0x1bcfff19c00>
```



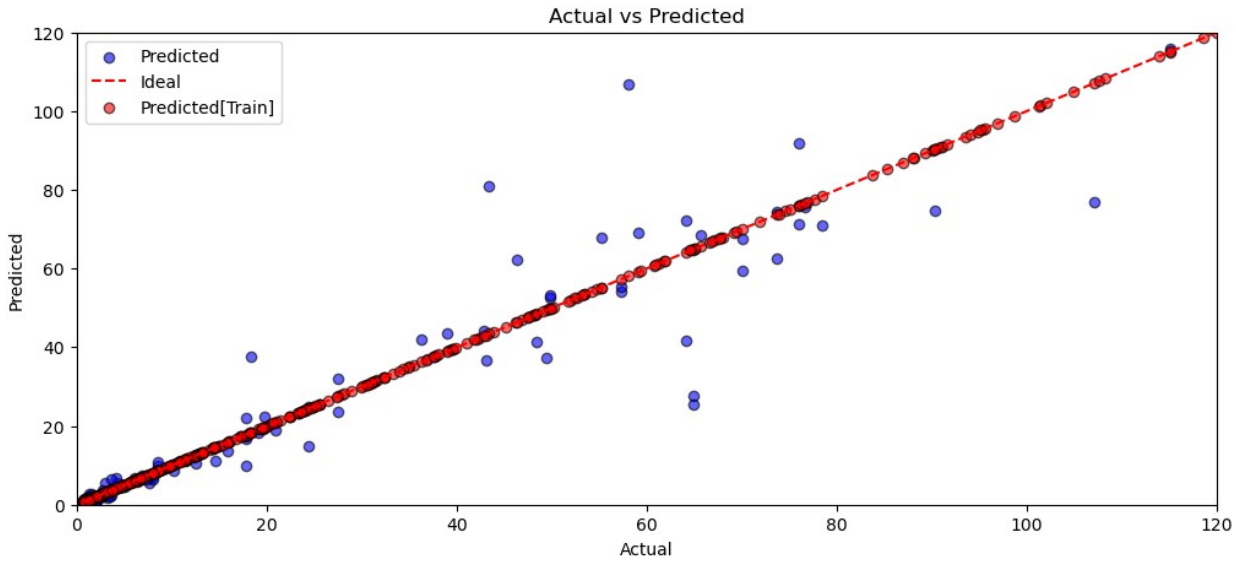

```
print('R_2 Score :', r2_score(y_pred, X_test[:,-1]))
```

```
R_2 Score : -13.511398606150385
```

```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6,
label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black',
alpha=0.6, label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c =
'green' ,edgecolors='black', alpha=0.6, label = 'Predicted[Train]')
plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)
plt.title('Log-Log Scatter Plot for Decision Tree Regression')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
plt.show()
```

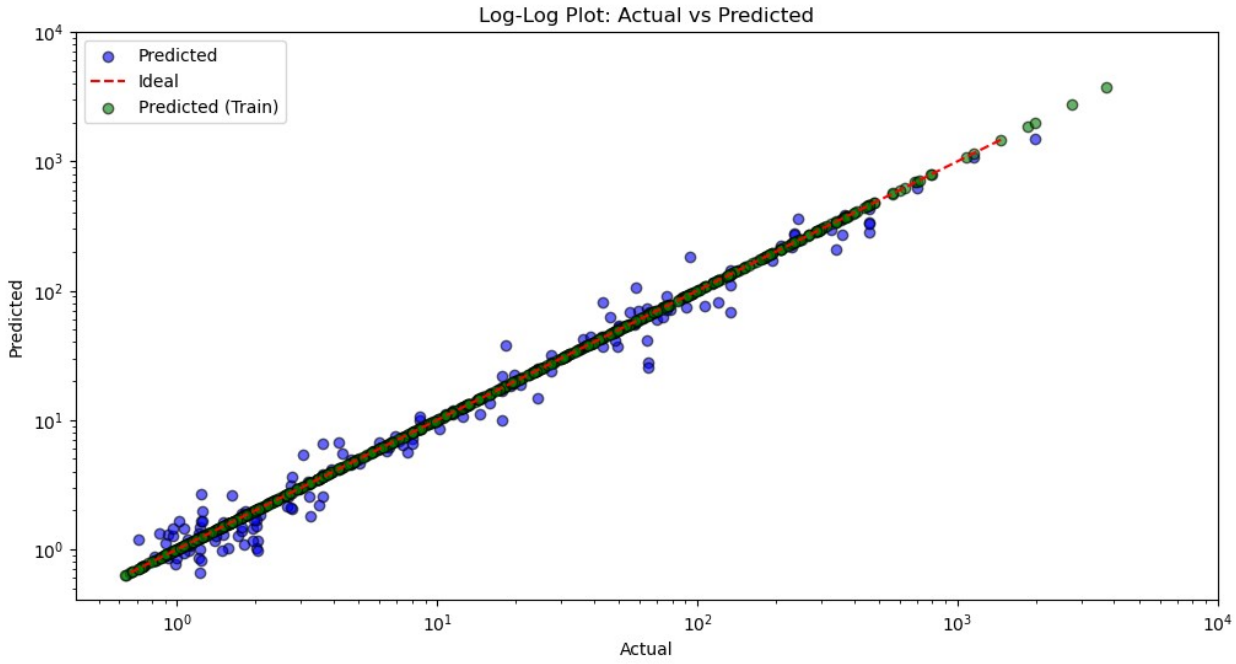



```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,
label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black',
alpha=0.6, label='Predicted[Train]')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0, 120)
plt.ylim(0, 120)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



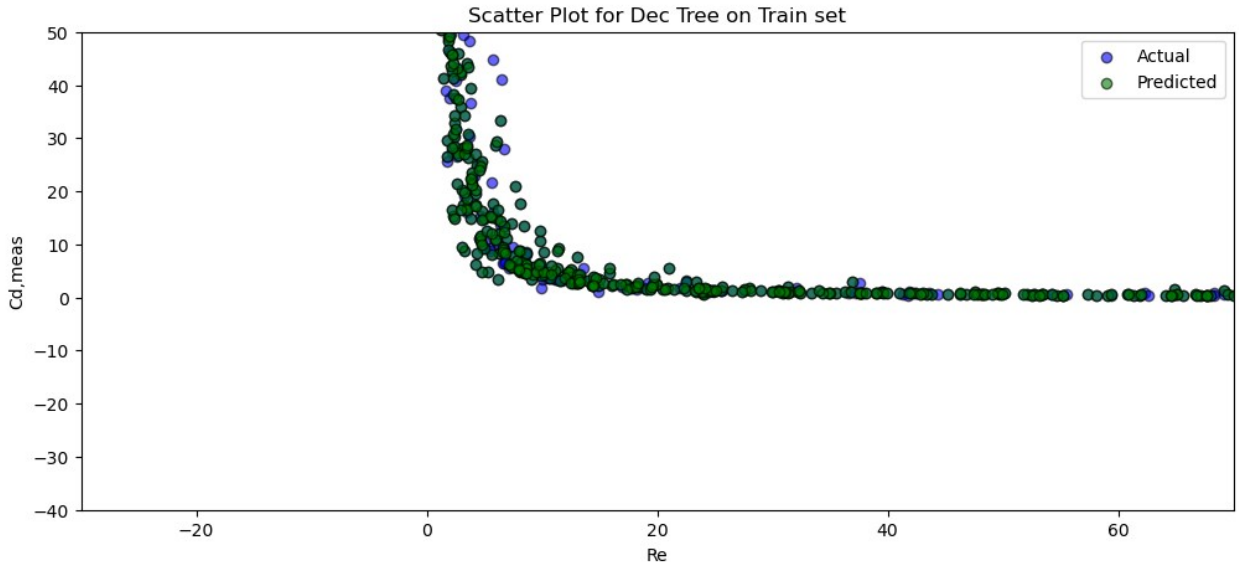
```
plt.figure(figsize=(12, 6))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black',
alpha=0.6, label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black',
alpha=0.6, label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.0,10000)
plt.xlim(0,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```

```
C:\Users\91897\AppData\Local\Temp\ipykernel_16624\3659971482.py:12:
UserWarning: Attempt to set non-positive ylim on a log-scaled axis
will be ignored.
  plt.ylim(0.0,10000)
C:\Users\91897\AppData\Local\Temp\ipykernel_16624\3659971482.py:13:
UserWarning: Attempt to set non-positive xlim on a log-scaled axis
will be ignored.
  plt.xlim(0,10000)
```

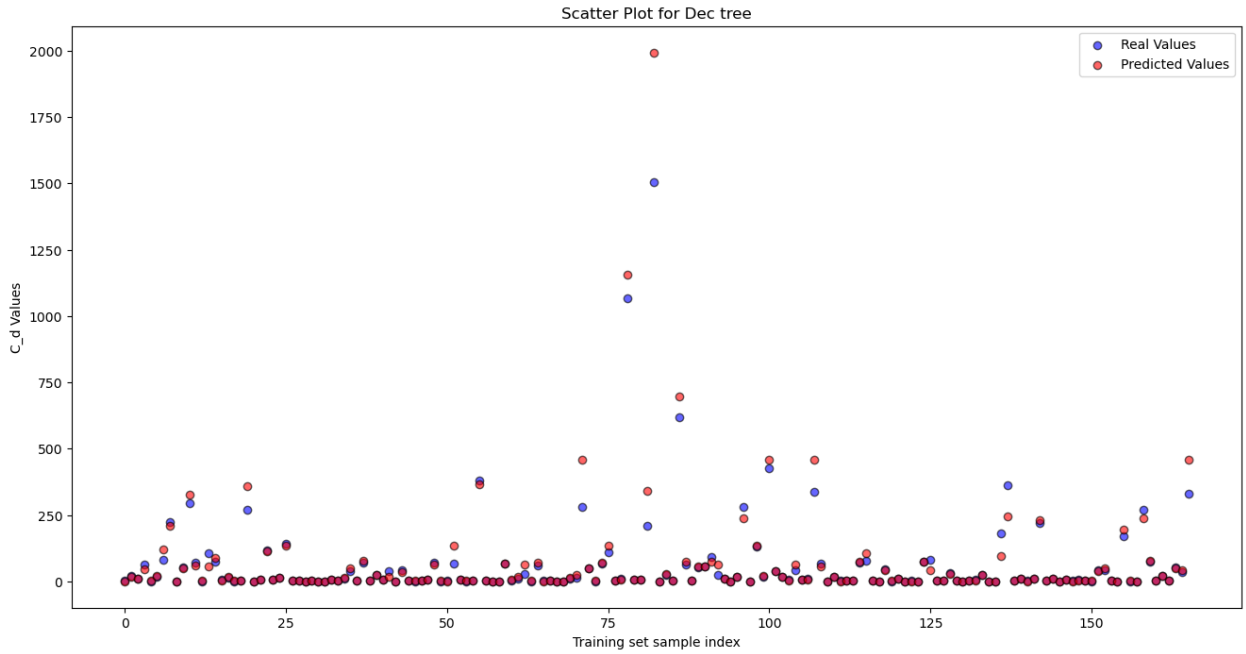



```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6,
label = 'Actual')
plt.scatter( y_pred_train ,X_train[:,-1], c =
'green' ,edgecolors='black', alpha=0.6, label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Ideal')
plt.ylim(-40, 50)
plt.xlim(-30, 70)
plt.title('Scatter Plot for Dec Tree on Train set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1bc805fe020>
```




```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color =
'blue',edgecolors='black', alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color =
'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for Dec tree')
plt.xlabel('Training set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



random-forest-regression

July 15, 2024

1 Random Forest Regression

1.1 Importing the libraries

[29]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1.2 Importing the dataset

[30]:
```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

1.3 Splitting the dataset into the Training set and Test set

[31]:
```
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 = 0)
```

1.4 Training the Random Forest Regression model on the whole dataset

[32]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X_train, y_train)
```

[32]:
```
RandomForestRegressor(n_estimators=10, random_state=0)
```

1.5 Predicting the Test set results

[33]:
```
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))
```

```
[[1.59e+00 1.49e+00]
 [2.13e+01 2.20e+01]
 [1.16e+01 1.17e+01]
 [4.30e+01 6.23e+01]
 [1.45e+00 1.26e+00]
 [2.01e+01 1.92e+01]
 [1.09e+02 8.22e+01]
 [2.07e+02 2.24e+02]
 [1.30e+00 1.00e+00]
 [4.96e+01 5.26e+01]
 [2.87e+02 2.96e+02]
 [6.16e+01 6.92e+01]
 [1.57e+00 1.45e+00]
 [7.61e+01 1.07e+02]
 [7.35e+01 7.48e+01]
 [3.59e+00 5.42e+00]
 [1.31e+01 1.36e+01]
 [1.60e+00 1.03e+00]
 [3.26e+00 3.84e+00]
 [3.22e+02 2.69e+02]
 [1.36e+00 8.61e-01]
 [6.41e+00 6.20e+00]
 [1.05e+02 1.16e+02]
 [7.69e+00 5.61e+00]
 [1.23e+01 1.26e+01]
 [1.40e+02 1.44e+02]
 [4.04e+00 4.95e+00]
 [1.42e+00 1.84e+00]
 [1.20e+00 1.29e+00]
 [2.87e+00 2.06e+00]
 [1.20e+00 8.64e-01]
 [1.33e+00 1.17e+00]
 [7.92e+00 7.14e+00]
 [2.34e+00 2.57e+00]
 [1.34e+01 1.12e+01]
 [5.02e+01 3.72e+01]
 [3.27e+00 2.56e+00]
 [7.49e+01 7.10e+01]
 [2.58e+00 3.12e+00]
 [2.48e+01 2.55e+01]
 [5.69e+00 5.73e+00]
 [1.80e+01 3.76e+01]
 [1.19e+00 1.18e+00]
 [4.66e+01 4.18e+01]
 [1.67e+00 1.88e+00]
 [1.48e+00 1.10e+00]
 [1.82e+00 1.51e+00]
 [5.55e+00 5.70e+00]
```

```
[6.76e+01 7.21e+01]
[1.47e+00 1.16e+00]
[1.90e+00 1.67e+00]
[4.77e+01 6.83e+01]
[6.56e+00 6.83e+00]
[1.54e+00 1.29e+00]
[4.76e+00 4.57e+00]
[3.59e+02 3.81e+02]
[2.62e+00 3.65e+00]
[1.33e+00 1.23e+00]
[9.31e-01 1.19e+00]
[6.93e+01 6.83e+01]
[5.21e+00 4.64e+00]
[2.11e+01 9.92e+00]
[5.26e+01 2.76e+01]
[1.39e+00 1.02e+00]
[7.09e+01 5.94e+01]
[2.70e+00 1.66e+00]
[1.83e+00 1.71e+00]
[1.10e+00 7.77e-01]
[1.40e+00 1.31e+00]
[1.18e+01 1.05e+01]
[2.80e+01 1.49e+01]
[4.64e+02 2.80e+02]
[4.84e+01 4.83e+01]
[1.68e+00 1.37e+00]
[7.03e+01 6.77e+01]
[1.39e+02 1.10e+02]
[3.28e+00 3.52e+00]
[7.92e+00 1.00e+01]
[1.24e+03 1.07e+03]
[6.06e+00 6.55e+00]
[6.84e+00 6.39e+00]
[3.38e+02 2.09e+02]
[1.92e+03 1.50e+03]
[9.05e-01 8.60e-01]
[3.22e+01 2.37e+01]
[4.56e+00 4.89e+00]
[5.59e+02 6.19e+02]
[7.23e+01 6.27e+01]
[1.64e+00 2.61e+00]
[5.93e+01 5.43e+01]
[5.93e+01 5.55e+01]
[7.31e+01 9.17e+01]
[5.41e+01 2.56e+01]
[9.03e+00 8.60e+00]
[1.16e+00 9.80e-01]
[3.08e+01 1.82e+01]
```

[2.20e+02 2.79e+02]
[1.40e+00 1.26e+00]
[1.36e+02 1.31e+02]
[2.12e+01 1.88e+01]
[5.85e+02 4.28e+02]
[3.51e+01 3.95e+01]
[1.83e+01 1.69e+01]
[4.59e+00 5.57e+00]
[6.24e+01 4.17e+01]
[7.57e+00 7.50e+00]
[7.92e+00 1.07e+01]
[4.23e+02 3.39e+02]
[6.57e+01 6.78e+01]
[1.07e+00 8.81e-01]
[2.11e+01 1.74e+01]
[1.72e+00 2.66e+00]
[3.11e+00 2.54e+00]
[3.43e+00 3.31e+00]
[7.81e+01 7.12e+01]
[1.04e+02 7.68e+01]
[2.77e+00 3.72e+00]
[1.18e+00 1.45e+00]
[4.39e+01 4.43e+01]
[1.35e+00 9.68e-01]
[9.76e+00 1.11e+01]
[1.29e+00 1.12e+00]
[2.44e+00 1.97e+00]
[1.27e+00 1.07e+00]
[7.29e+01 7.45e+01]
[6.57e+01 8.09e+01]
[1.71e+00 1.70e+00]
[1.86e+00 1.47e+00]
[3.00e+01 3.21e+01]
[1.20e+00 1.61e+00]
[1.55e+00 6.60e-01]
[2.60e+00 1.79e+00]
[4.68e+00 6.70e+00]
[2.55e+01 2.48e+01]
[1.13e+00 1.19e+00]
[1.02e+00 9.35e-01]
[8.25e+01 1.83e+02]
[2.16e+02 3.64e+02]
[1.56e+00 1.60e+00]
[1.06e+01 1.10e+01]
[1.44e+00 1.64e+00]
[1.13e+01 1.15e+01]
[2.63e+02 2.20e+02]
[4.08e+00 4.15e+00]

```
[9.12e+00 8.61e+00]
[1.16e+00 8.28e-01]
[6.19e+00 6.73e+00]
[1.49e+00 1.46e+00]
[2.90e+00 6.53e+00]
[2.60e+00 2.16e+00]
[1.48e+00 1.18e+00]
[4.33e+01 4.36e+01]
[4.40e+01 4.13e+01]
[2.72e+00 2.20e+00]
[1.10e+00 1.33e+00]
[1.87e+02 1.72e+02]
[1.36e+00 9.83e-01]
[1.37e+00 1.28e+00]
[2.07e+02 2.71e+02]
[6.76e+01 7.56e+01]
[2.81e+00 2.09e+00]
[1.53e+01 2.25e+01]
[1.86e+00 1.98e+00]
[4.88e+01 5.34e+01]
[4.25e+01 3.66e+01]
[5.40e+02 3.31e+02]]
```

1.6 Evaluating the Model Performance

[34]:
```
from sklearn.metrics import r2_score, mean_squared_error
print('R_2 Score :', r2_score(y_test, y_pred))
print('Mean Squared Error :', mean_squared_error(y_test, y_pred))
print('MSE for Training set:', mean_squared_error(y_train, y_pred_train))
print('R_2 score for Training Set:',r2_score(y_train, y_pred_train))
```

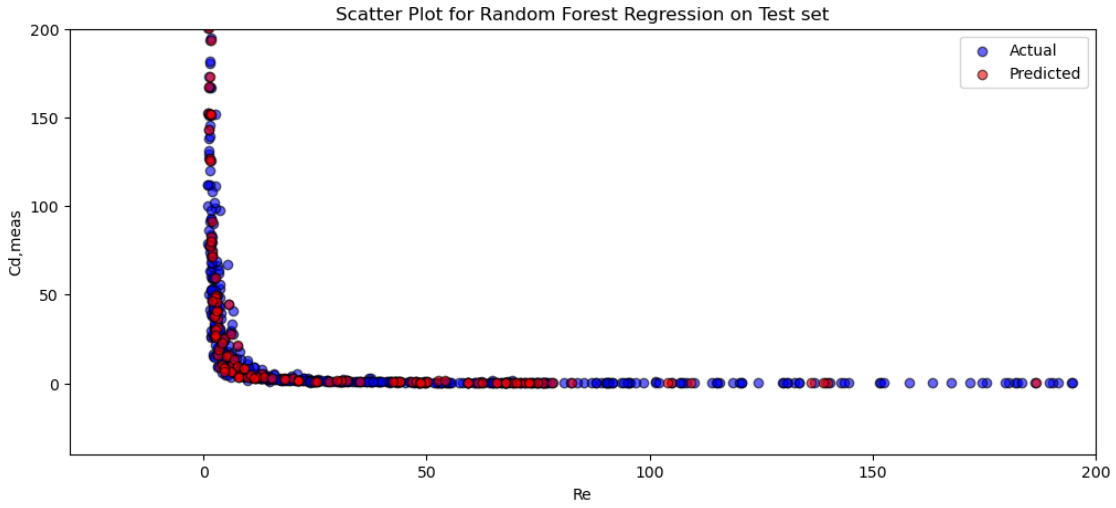
```
R_2 Score : 0.9163039470088328
Mean Squared Error : 2314.294640914941
MSE for Training set: 1332.1837893905595
R_2 score for Training Set: 0.9782734784395334
```

##Plotting Results

[35]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
↪'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' , edgecolors='black', alpha=0.
↪6,label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
↪label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
```

```
plt.title('Scatter Plot for Random Forest Regression on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

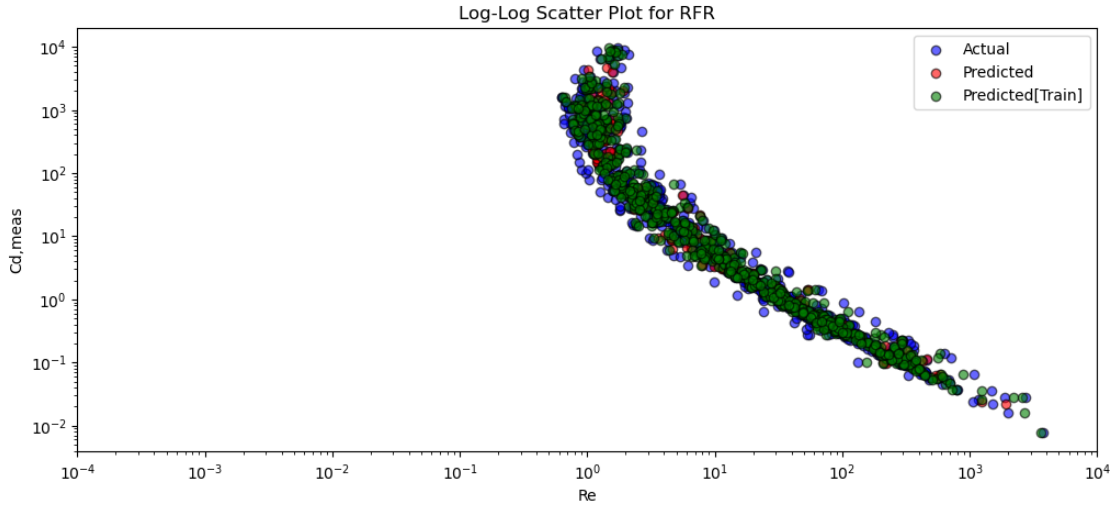
[35]: <matplotlib.legend.Legend at 0x216a191b4c0>



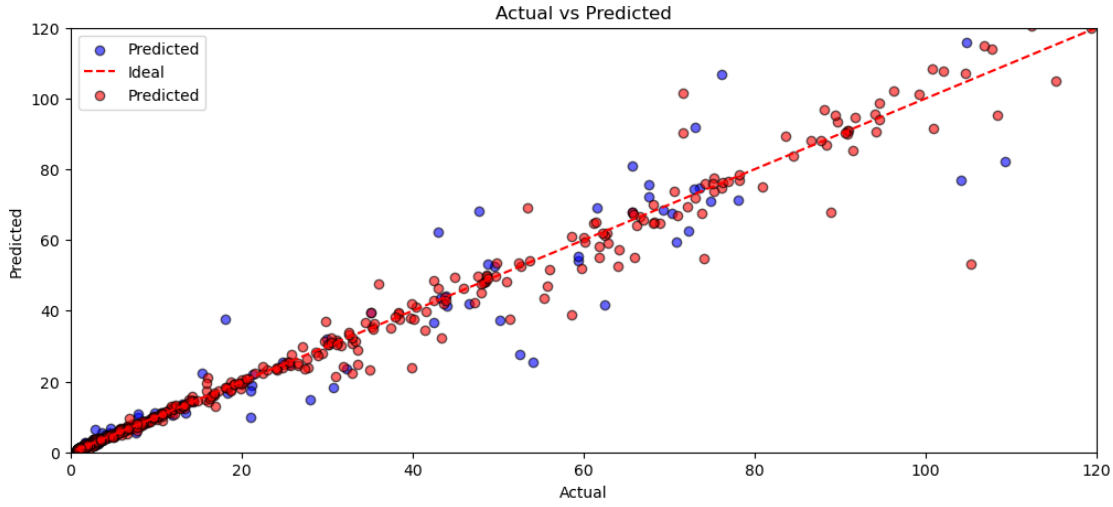
[36]:
```
print('R_2 Score :', r2_score(y_pred, X_test[:,-1]))
```

R_2 Score : -13.816768577668052

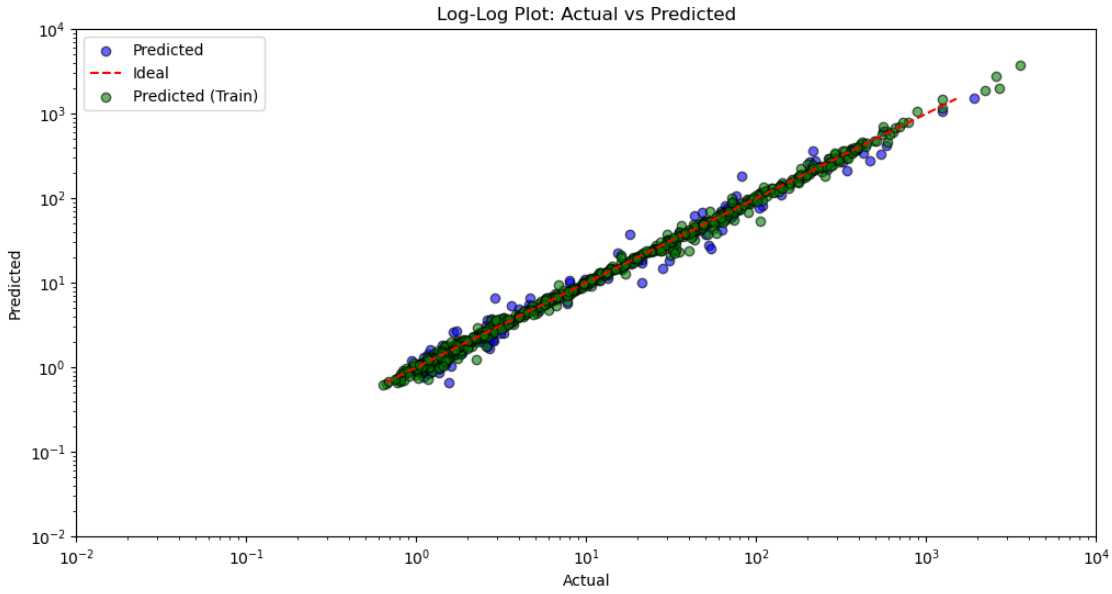
[43]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6, label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c = 'green' ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Predicted[Train]')
plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)
plt.title('Log-Log Scatter Plot for RFR')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
plt.show()
```



[38]:
```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0, 120)
plt.ylim(0, 120)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



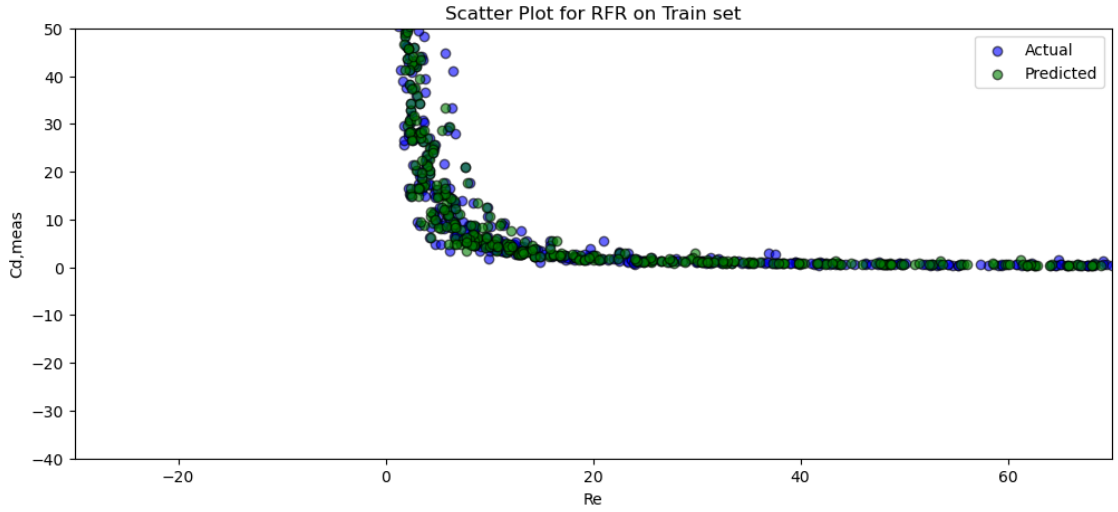
[39]:
```
plt.figure(figsize=(12, 6))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.01,10000)
plt.xlim(0.01,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```



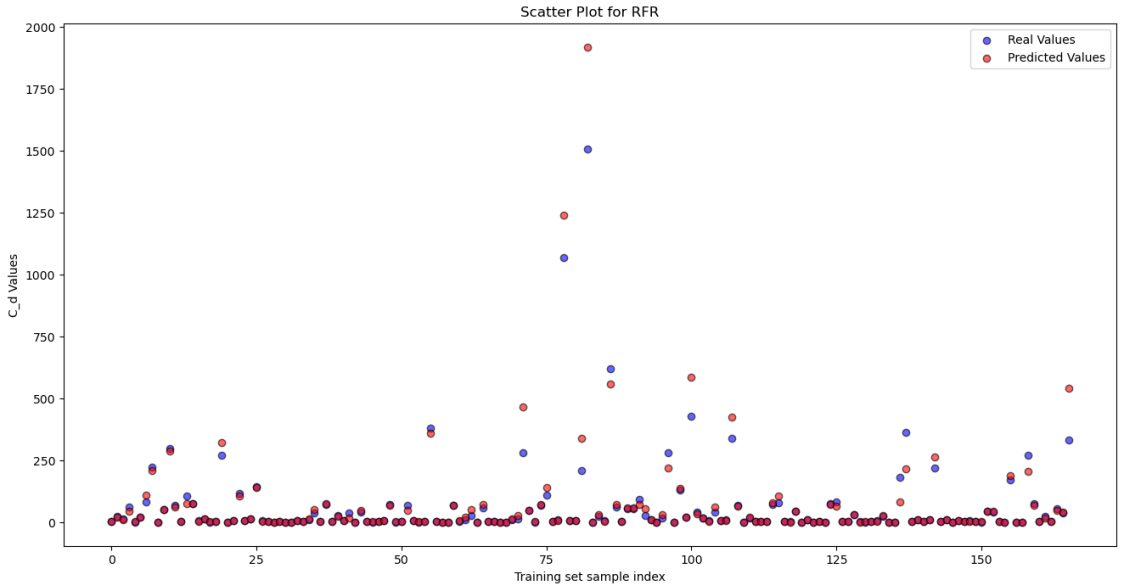
[40]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
```

```
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
 ↪'Actual')
plt.scatter( y_pred_train ,X_train[:,-1], c = 'green' ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.ylim(-40, 50)
plt.xlim(-30, 70)
plt.title('Scatter Plot for RFR on Train set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

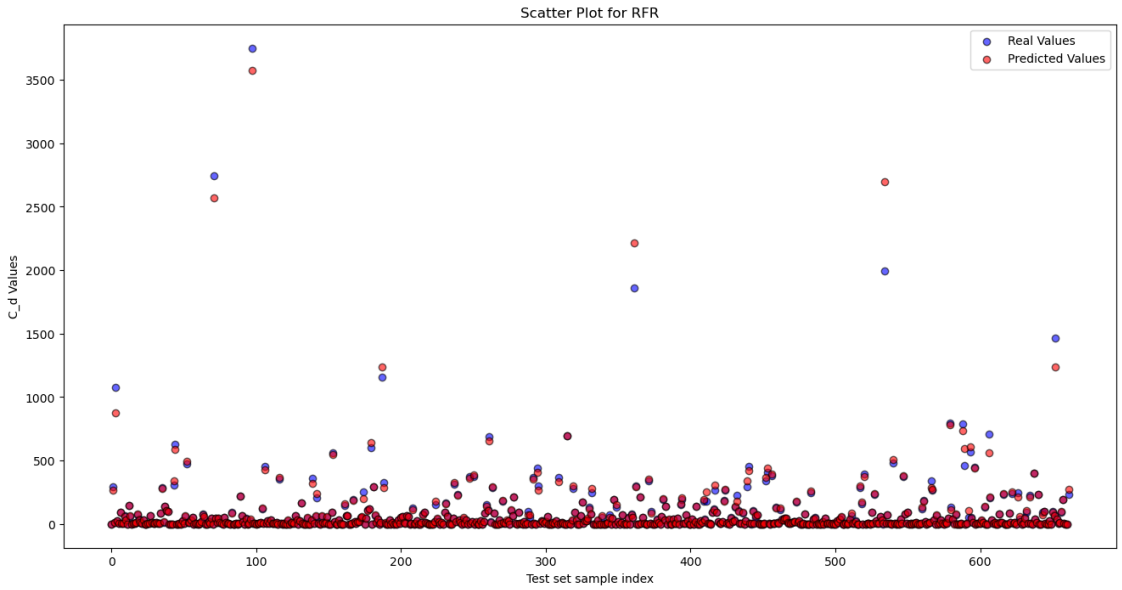
[40]: <matplotlib.legend.Legend at 0x216a37f5120>



[41]:
```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color = 'blue',edgecolors='black',␣
 ↪alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color = 'red',edgecolors='black',␣
 ↪alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for RFR')
plt.xlabel('Training set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



[42]:
```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_train)), y_train, color = 'blue',edgecolors='black',␣
 ↪alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =␣
 ↪'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for RFR')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



svr-gridsearch

July 15, 2024

1 Support Vector Regression (SVR)

1.1 Importing the libraries

[35]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

1.2 Importing the dataset

[36]:
```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

[37]:
```
y = y.reshape(len(y),1)
```

1.3 Splitting the dataset into the Training set and Test set

[38]:
```
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 = 0)
```

##Applying Feature Scaling

[39]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X_train = sc_X.fit_transform(X_train)
y_train = sc_y.fit_transform(y_train)
```

1.4 Training the SVR model on the Training set

[40]:
```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)
```

```
c:\Users\91897\anaconda3\envs\myenv\lib\site-
packages\sklearn\utils\validation.py:1300: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[40]:
```
SVR()
```

##Applying Grid Seach

[41]:
```
from sklearn.model_selection import GridSearchCV
grid_param = { 'C': [0.1, 1, 10, 100, 1000], 'gamma': [1, 0.1, 0.01, 0.001, 0.
 ↪0001], 'kernel': ['rbf','linear']}
model = SVR()
grid_search = GridSearchCV(estimator=model,
                           param_grid=grid_param,
                           scoring='neg_mean_squared_error',
                           cv=5,
                           n_jobs=-1)
# Create the GridSearchCV object
grid_search = GridSearchCV(model, grid_param, refit=True, verbose=2, cv=5)

# Fit the grid search to the data
grid_search.fit(X_train, y_train.ravel())
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
print("Best Parameters:", best_params)
print("Best Model:", best_model)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
```

```
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
```

```
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.0s
```

```
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.01, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.0001, kernel=linear; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.1, kernel=rbf; total time=   0.0s
```

```
[CV] END …C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.1, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.1, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.01, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.01, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.01, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.01, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.01, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.001, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.001, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=100, gamma=0.0001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.0001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.0001, kernel=linear; total time=   0.1s
[CV] END …C=100, gamma=0.0001, kernel=linear; total time=   0.2s
[CV] END …C=100, gamma=0.0001, kernel=linear; total time=   0.2s
[CV] END …C=1000, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1000, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=1000, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1000, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1000, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1000, gamma=1, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=1, kernel=linear; total time=   1.6s
[CV] END …C=1000, gamma=1, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=1, kernel=linear; total time=   2.2s
[CV] END …C=1000, gamma=1, kernel=linear; total time=   1.9s
```

```
[CV] END …C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.1, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=0.1, kernel=linear; total time=   1.6s
[CV] END …C=1000, gamma=0.1, kernel=linear; total time=   2.6s
[CV] END …C=1000, gamma=0.1, kernel=linear; total time=   2.6s
[CV] END …C=1000, gamma=0.1, kernel=linear; total time=   2.1s
[CV] END …C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.01, kernel=linear; total time=   2.5s
[CV] END …C=1000, gamma=0.01, kernel=linear; total time=   1.6s
[CV] END …C=1000, gamma=0.01, kernel=linear; total time=   2.6s
[CV] END …C=1000, gamma=0.01, kernel=linear; total time=   2.3s
[CV] END …C=1000, gamma=0.01, kernel=linear; total time=   1.9s
[CV] END …C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.001, kernel=linear; total time=   2.6s
[CV] END …C=1000, gamma=0.001, kernel=linear; total time=   1.6s
[CV] END …C=1000, gamma=0.001, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=0.001, kernel=linear; total time=   2.2s
[CV] END …C=1000, gamma=0.001, kernel=linear; total time=   2.0s
[CV] END …C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV] END …C=1000, gamma=0.0001, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=0.0001, kernel=linear; total time=   1.7s
[CV] END …C=1000, gamma=0.0001, kernel=linear; total time=   2.4s
[CV] END …C=1000, gamma=0.0001, kernel=linear; total time=   2.3s
[CV] END …C=1000, gamma=0.0001, kernel=linear; total time=   1.9s
Best Parameters: {'C': 100, 'gamma': 1, 'kernel': 'rbf'}
Best Model: SVR(C=100, gamma=1)
```

##Predicting the Test set Results

[42]:
```
y_pred = sc_y.inverse_transform(regressor.predict(sc_X.transform(X_test)).
↪reshape(-1,1))
```

```
y_pred_train = sc_y.inverse_transform(regressor.predict(sc_X.
  ↪transform(X_train)).reshape(-1,1))
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
  ↪reshape(len(y_test),1)),1))
```

```
[[ 4.56e+00  1.49e+00]
 [ 3.50e+01  2.20e+01]
 [ 3.35e+01  1.17e+01]
 [ 3.26e+01  6.23e+01]
 [ 9.62e+00  1.26e+00]
 [ 3.32e+01  1.92e+01]
 [ 3.51e+01  8.22e+01]
 [ 4.55e+01  2.24e+02]
 [-2.16e+01  1.00e+00]
 [ 5.35e+01  5.26e+01]
 [ 3.79e+01  2.96e+02]
 [ 3.43e+01  6.92e+01]
 [ 1.95e+01  1.45e+00]
 [ 3.70e+01  1.07e+02]
 [ 5.70e+01  7.48e+01]
 [ 3.25e+01  5.42e+00]
 [ 3.24e+01  1.36e+01]
 [-2.33e+01  1.03e+00]
 [ 2.89e+01  3.84e+00]
 [ 3.36e+01  2.69e+02]
 [-7.24e+00  8.61e-01]
 [ 3.51e+01  6.20e+00]
 [ 3.26e+01  1.16e+02]
 [ 3.25e+01  5.61e+00]
 [ 3.47e+01  1.26e+01]
 [ 3.48e+01  1.44e+02]
 [ 3.24e+01  4.95e+00]
 [ 2.84e+01  1.84e+00]
 [ 1.72e+01  1.29e+00]
 [ 3.01e+01  2.06e+00]
 [ 1.56e+01  8.64e-01]
 [-7.47e+00  1.17e+00]
 [ 3.89e+01  7.14e+00]
 [ 5.37e+01  2.57e+00]
 [ 3.66e+01  1.12e+01]
 [ 3.76e+01  3.72e+01]
 [ 3.19e+01  2.56e+00]
 [ 3.97e+01  7.10e+01]
 [ 2.70e+01  3.12e+00]
 [ 3.39e+01  2.55e+01]
 [ 3.31e+01  5.73e+00]
```

```
[ 3.74e+01  3.76e+01]
[-1.39e+01  1.18e+00]
[ 3.76e+01  4.18e+01]
[ 2.43e+01  1.88e+00]
[-1.86e+01  1.10e+00]
[ 2.40e+01  1.51e+00]
[ 3.06e+01  5.70e+00]
[ 5.12e+01  7.21e+01]
[-6.41e+00  1.16e+00]
[ 2.13e+01  1.67e+00]
[ 3.26e+01  6.83e+01]
[ 3.37e+01  6.83e+00]
[ 4.42e+01  1.29e+00]
[ 3.19e+01  4.57e+00]
[ 3.53e+01  3.81e+02]
[ 2.99e+01  3.65e+00]
[ 1.58e+01  1.23e+00]
[-1.82e+01  1.19e+00]
[ 4.10e+01  6.83e+01]
[ 3.42e+01  4.64e+00]
[ 3.37e+01  9.92e+00]
[ 3.38e+01  2.76e+01]
[-2.20e+00  1.02e+00]
[ 4.68e+01  5.94e+01]
[ 3.18e+01  1.66e+00]
[ 2.46e+01  1.71e+00]
[-4.89e+00  7.77e-01]
[ 1.23e+01  1.31e+00]
[ 3.46e+01  1.05e+01]
[ 3.38e+01  1.49e+01]
[ 3.79e+01  2.80e+02]
[ 3.75e+01  4.83e+01]
[ 1.12e+01  1.37e+00]
[ 4.71e+01  6.77e+01]
[ 3.66e+01  1.10e+02]
[ 3.73e+01  3.52e+00]
[ 3.92e+01  1.00e+01]
[ 3.97e+01  1.07e+03]
[ 3.88e+01  6.55e+00]
[ 3.57e+01  6.39e+00]
[ 3.71e+01  2.09e+02]
[ 3.67e+01  1.50e+03]
[-1.17e+01  8.60e-01]
[ 3.35e+01  2.37e+01]
[ 3.83e+01  4.89e+00]
[ 3.64e+01  6.19e+02]
[ 5.05e+01  6.27e+01]
[ 1.63e+01  2.61e+00]
```

```
[ 3.65e+01  5.43e+01]
[ 3.78e+01  5.55e+01]
[ 4.12e+01  9.17e+01]
[ 3.50e+01  2.56e+01]
[ 3.65e+01  8.60e+00]
[-8.25e-01  9.80e-01]
[ 5.48e+01  1.82e+01]
[ 5.42e+01  2.79e+02]
[ 6.36e+00  1.26e+00]
[ 3.39e+01  1.31e+02]
[ 3.52e+01  1.88e+01]
[ 3.40e+01  4.28e+02]
[ 3.51e+01  3.95e+01]
[ 5.10e+01  1.69e+01]
[ 3.84e+01  5.57e+00]
[ 5.12e+01  4.17e+01]
[ 3.39e+01  7.50e+00]
[ 3.69e+01  1.07e+01]
[ 3.77e+01  3.39e+02]
[ 5.72e+01  6.78e+01]
[ 1.79e+01  8.81e-01]
[ 3.36e+01  1.74e+01]
[ 1.07e+01  2.66e+00]
[ 3.16e+01  2.54e+00]
[ 3.57e+01  3.31e+00]
[ 5.72e+01  7.12e+01]
[ 5.12e+01  7.68e+01]
[ 2.93e+01  3.72e+00]
[ 3.60e+01  1.45e+00]
[ 5.66e+01  4.43e+01]
[ 1.44e+01  9.68e-01]
[ 3.92e+01  1.11e+01]
[ 2.33e+01  1.12e+00]
[ 3.21e+01  1.97e+00]
[-1.46e+01  1.07e+00]
[ 5.47e+01  7.45e+01]
[ 3.51e+01  8.09e+01]
[ 2.46e+01  1.70e+00]
[ 2.03e+01  1.47e+00]
[ 3.26e+01  3.21e+01]
[ 1.65e+00  1.61e+00]
[-2.11e+01  6.60e-01]
[ 3.00e+01  1.79e+00]
[ 3.30e+01  6.70e+00]
[ 3.95e+01  2.48e+01]
[-2.32e+01  1.19e+00]
[ 2.77e+01  9.35e-01]
[ 3.77e+01  1.83e+02]
```

```
 [ 3.73e+01  3.64e+02]
 [ 2.94e+01  1.60e+00]
 [ 3.46e+01  1.10e+01]
 [ 1.96e+01  1.64e+00]
 [ 3.47e+01  1.15e+01]
 [ 3.40e+01  2.20e+02]
 [ 3.06e+01  4.15e+00]
 [ 3.65e+01  8.61e+00]
 [ 6.37e+00  8.28e-01]
 [ 3.14e+01  6.73e+00]
 [ 4.37e+00  1.46e+00]
 [ 2.92e+01  6.53e+00]
 [ 5.36e+01  2.16e+00]
 [ 2.64e+01  1.18e+00]
 [ 3.65e+01  4.36e+01]
 [ 3.39e+01  4.13e+01]
 [ 2.67e+01  2.20e+00]
 [ 4.90e+00  1.33e+00]
 [ 5.44e+01  1.72e+02]
 [-9.93e+00  9.83e-01]
 [-1.67e+00  1.28e+00]
 [ 5.74e+01  2.71e+02]
 [ 3.63e+01  7.56e+01]
 [ 2.77e+01  2.09e+00]
 [ 3.25e+01  2.25e+01]
 [ 3.13e+01  1.98e+00]
 [ 4.14e+01  5.34e+01]
 [ 3.51e+01  3.66e+01]
 [ 3.47e+01  3.31e+02]]
```

1.5 Evaluating the Model Performance and Regression plot

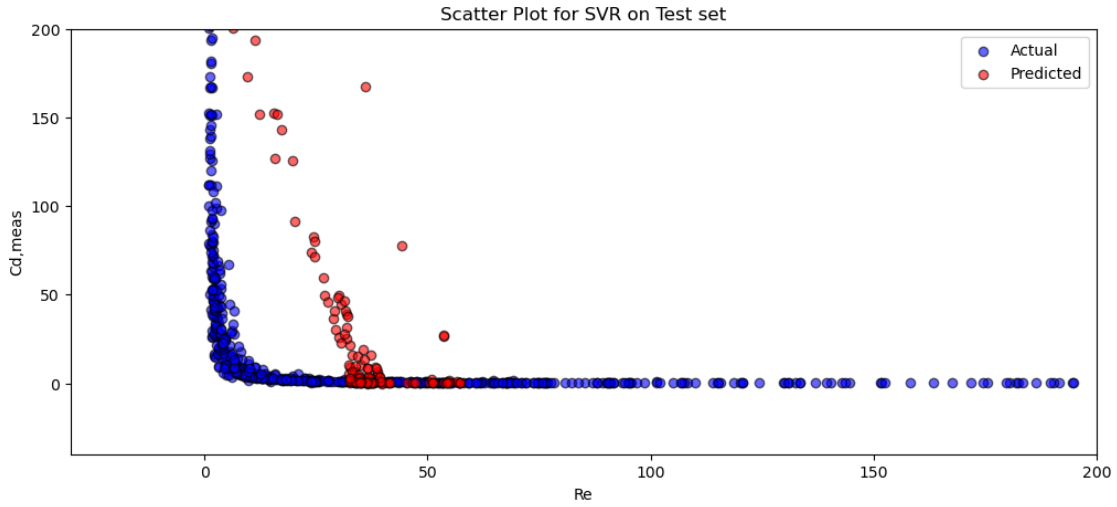
[43]:
```
from sklearn.metrics import r2_score, mean_squared_error
print('R_2 Score :', r2_score(y_test, y_pred))
print('Mean Squared Error :', mean_squared_error(y_test, y_pred))
print('MSE for Training set:', mean_squared_error(y_train, y_pred_train))
print('R_2 score for Training set:',r2_score(y_train, y_pred_train))
```

```
R_2 Score : -0.009176582416843715
Mean Squared Error : 27904.923505420662
MSE for Training set: 1383.1164454472457
R_2 score for Training set: -1382.1164454472457
```

[44]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
 ↪'Actual')
```

```
plt.scatter( y_pred ,X_test[:,-1], c = 'red' , edgecolors='black', alpha=0.
 ↪6,label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for SVR on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

[44]: <matplotlib.legend.Legend at 0x1933febe740>

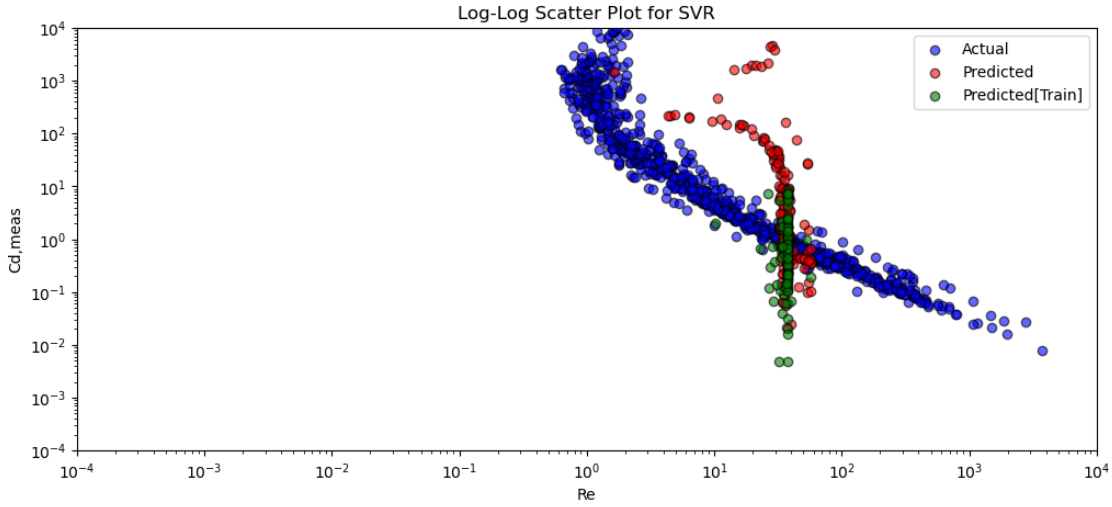


[45]:
```
print('R_2 Score :', r2_score(y_pred, X_test[:,-1]))
```

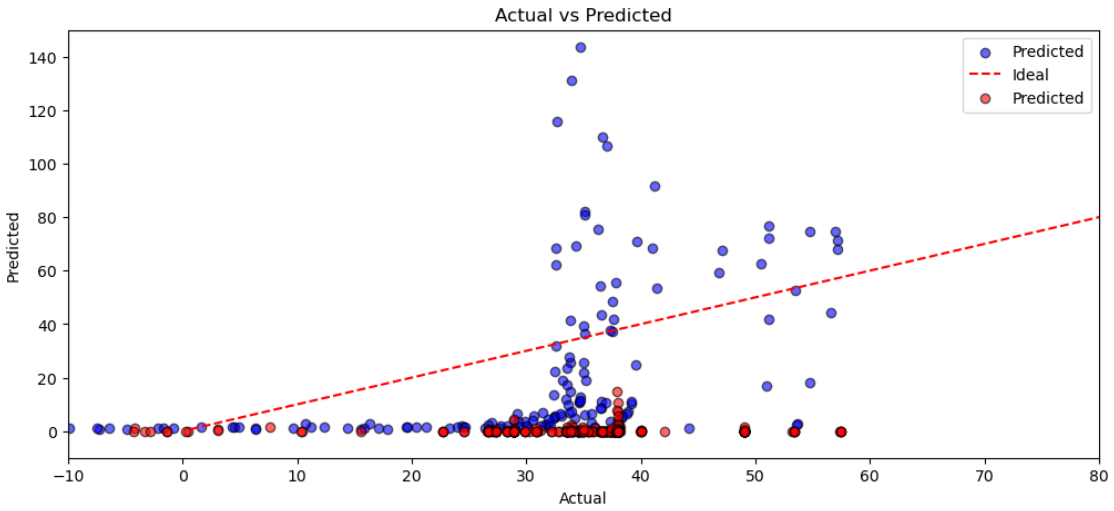
```
R_2 Score : -1734.8272064753535
```

[46]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6, label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c = 'green' ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Predicted[Train]')
plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)
plt.ylim(0.0001,10000)
plt.title('Log-Log Scatter Plot for SVR')
```

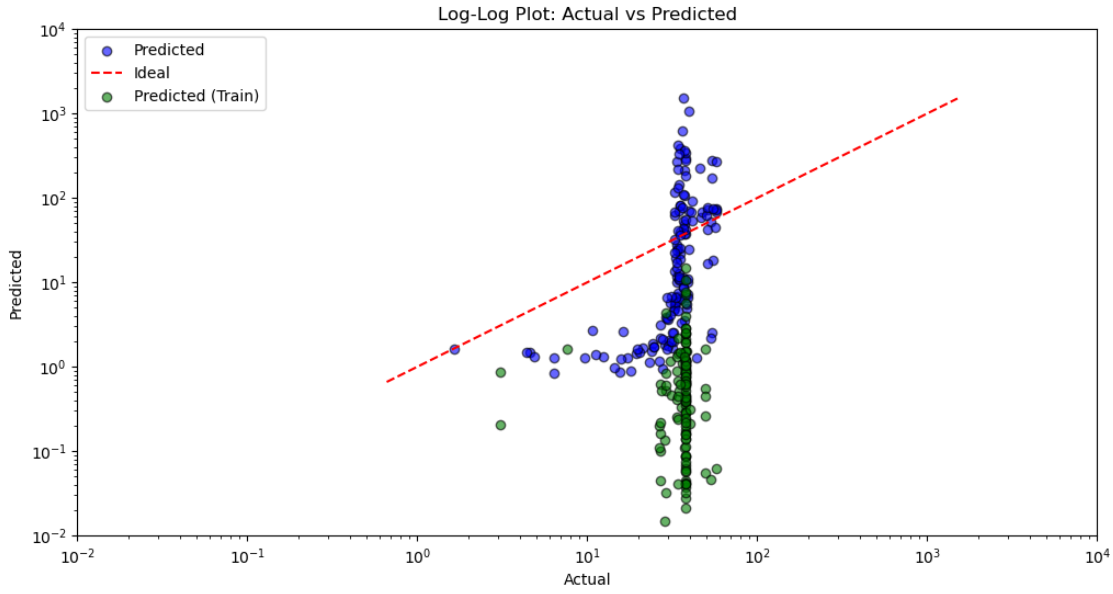
```
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
plt.show()
```



[47]:
```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 80)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```

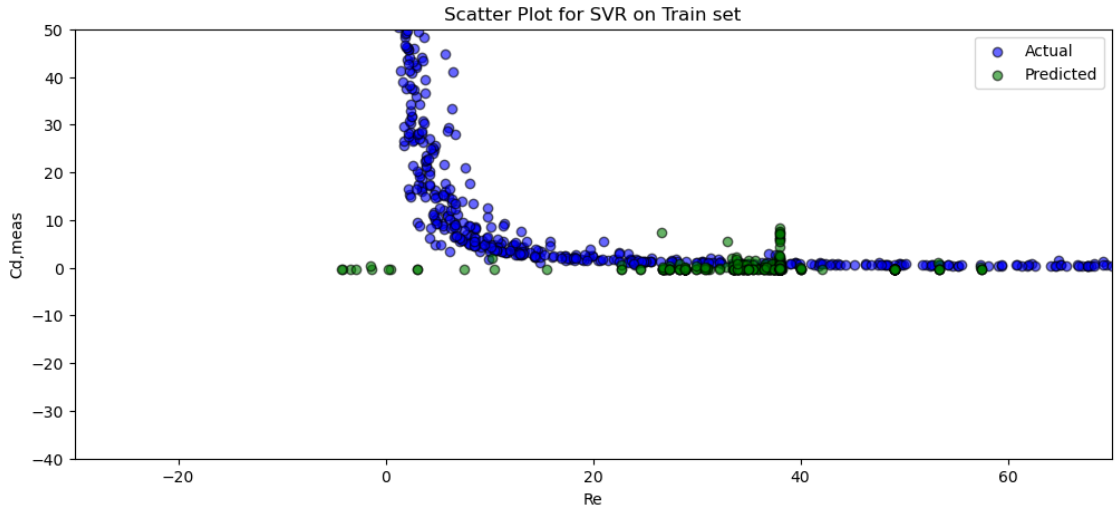


[48]:
```
plt.figure(figsize=(12, 6))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.01,10000)
plt.xlim(0.01,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```



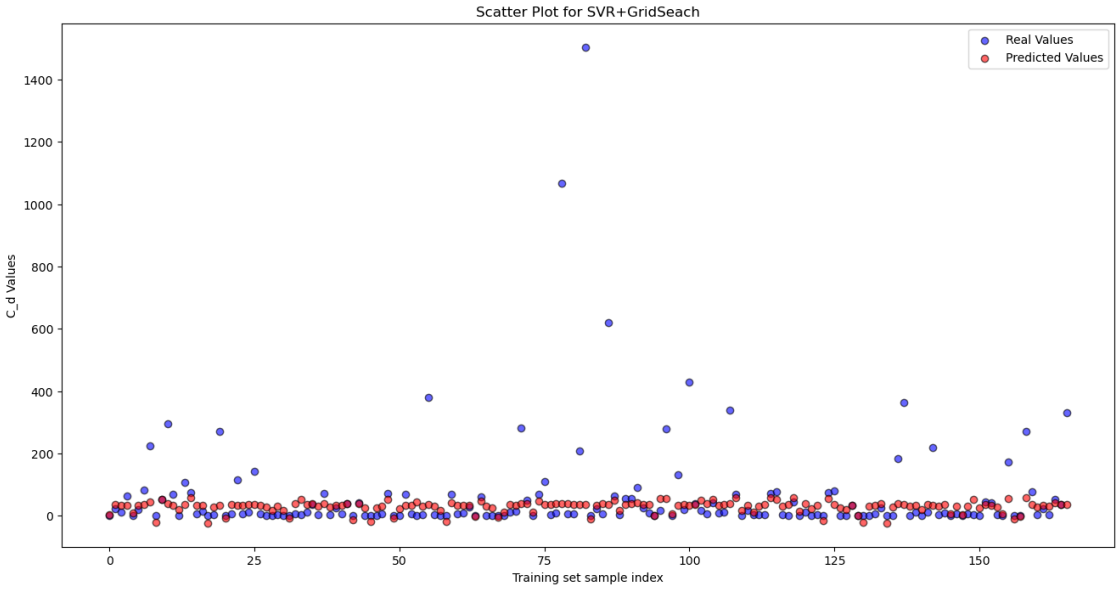
[49]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
↪'Actual')
plt.scatter( y_pred_train ,X_train[:,-1], c = 'green' ,edgecolors='black',␣
↪alpha=0.6, label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
↪label='Ideal')
plt.ylim(-40, 50)
plt.xlim(-30, 70)
plt.title('Scatter Plot for SVR on Train set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

[49]: <matplotlib.legend.Legend at 0x1933fed3c40>



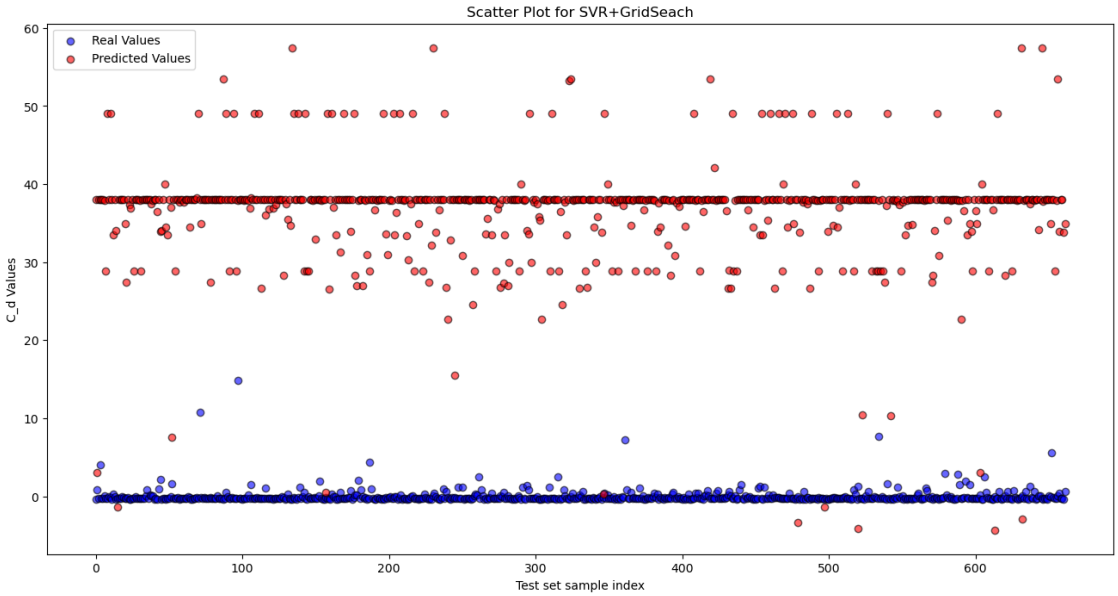
##Scatter Plot for Training set

[50]:
```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_test)), y_test, color = 'blue',edgecolors='black',␣
 ↪alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred)), y_pred , color = 'red',edgecolors='black',␣
 ↪alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for SVR+GridSeach')
plt.xlabel('Training set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



##Scatter plot for Test Set

[51]:
```
plt.figure(figsize=(16,8))
plt.scatter(range(len(y_train)), y_train, color = 'blue',edgecolors='black',␣
 ↪alpha=0.6,label='Real Values')
plt.scatter(range(len(y_pred_train)), y_pred_train , color =␣
 ↪'red',edgecolors='black', alpha=0.6,label='Predicted Values')
plt.title('Scatter Plot for SVR+GridSeach')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



artificial-neural-network

July 15, 2024

1 Artificial Neural Network

1.0.1 Importing the libraries

[113]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
```

[114]:
```
tf.__version__
```

[114]:
```
'2.10.0'
```

1.1 Part 1 - Data Preprocessing

1.1.1 Importing the dataset

[115]:
```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

1.1.2 Splitting the dataset into the Training set and Test set

[116]:
```
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 = 0)
```

1.2 Part 2 - Building the ANN

1.2.1 Initializing the ANN

[117]:
```
ann = tf.keras.models.Sequential()
```

1.2.2 Adding the input layer and the first hidden layer

[118]:
```
ann.add(tf.keras.layers.Dense(units=2, activation='relu'))
```

1.2.3 Adding the second hidden layer

[119]:
```
ann.add(tf.keras.layers.Dense(units=4, activation='relu'))
```

1.2.4 Adding the output layer

[120]:
```
ann.add(tf.keras.layers.Dense(units=1))
```

1.3 Part 3 - Training the ANN

1.3.1 Compiling the ANN

[121]:
```
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

1.3.2 Training the ANN model on the Training set

[122]:
```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
21/21 [==============================] - 1s 1ms/step - loss: 230528.5312
Epoch 2/100
21/21 [==============================] - 0s 1ms/step - loss: 194050.1406
Epoch 3/100
21/21 [==============================] - 0s 1ms/step - loss: 166020.6719
Epoch 4/100
21/21 [==============================] - 0s 1ms/step - loss: 142652.0312
Epoch 5/100
21/21 [==============================] - 0s 1ms/step - loss: 126514.8281
Epoch 6/100
21/21 [==============================] - 0s 1ms/step - loss: 113857.9844
Epoch 7/100
21/21 [==============================] - 0s 1ms/step - loss: 102487.1562
Epoch 8/100
21/21 [==============================] - 0s 1ms/step - loss: 94517.6484
Epoch 9/100
21/21 [==============================] - 0s 1ms/step - loss: 88400.4922
Epoch 10/100
21/21 [==============================] - 0s 1ms/step - loss: 83770.0469
Epoch 11/100
21/21 [==============================] - 0s 1ms/step - loss: 79997.1797
Epoch 12/100
21/21 [==============================] - 0s 1ms/step - loss: 77129.6172
Epoch 13/100
21/21 [==============================] - 0s 1ms/step - loss: 74819.7969
Epoch 14/100
21/21 [==============================] - 0s 1ms/step - loss: 72897.5234
Epoch 15/100
```

```
21/21 [==============================] - 0s 1ms/step - loss: 71598.7500
Epoch 16/100
21/21 [==============================] - 0s 1ms/step - loss: 70502.1797
Epoch 17/100
21/21 [==============================] - 0s 1ms/step - loss: 69657.8203
Epoch 18/100
21/21 [==============================] - 0s 1ms/step - loss: 69078.2109
Epoch 19/100
21/21 [==============================] - 0s 1ms/step - loss: 68568.8125
Epoch 20/100
21/21 [==============================] - 0s 1ms/step - loss: 68224.1250
Epoch 21/100
21/21 [==============================] - 0s 1ms/step - loss: 67942.2656
Epoch 22/100
21/21 [==============================] - 0s 1ms/step - loss: 67755.6406
Epoch 23/100
21/21 [==============================] - 0s 1ms/step - loss: 67585.3203
Epoch 24/100
21/21 [==============================] - 0s 1ms/step - loss: 67472.2266
Epoch 25/100
21/21 [==============================] - 0s 1ms/step - loss: 67379.9219
Epoch 26/100
21/21 [==============================] - 0s 1ms/step - loss: 67308.2578
Epoch 27/100
21/21 [==============================] - 0s 1ms/step - loss: 67198.2812
Epoch 28/100
21/21 [==============================] - 0s 1ms/step - loss: 67140.9375
Epoch 29/100
21/21 [==============================] - 0s 1ms/step - loss: 67099.6172
Epoch 30/100
21/21 [==============================] - 0s 1ms/step - loss: 67052.3516
Epoch 31/100
21/21 [==============================] - 0s 1ms/step - loss: 67003.8750
Epoch 32/100
21/21 [==============================] - 0s 1ms/step - loss: 66951.1094
Epoch 33/100
21/21 [==============================] - 0s 1ms/step - loss: 66897.6406
Epoch 34/100
21/21 [==============================] - 0s 1ms/step - loss: 66838.9219
Epoch 35/100
21/21 [==============================] - 0s 1ms/step - loss: 66777.8516
Epoch 36/100
21/21 [==============================] - 0s 1ms/step - loss: 66719.6719
Epoch 37/100
21/21 [==============================] - 0s 1ms/step - loss: 66656.3359
Epoch 38/100
21/21 [==============================] - 0s 1ms/step - loss: 66591.3984
Epoch 39/100
```

```
21/21 [==============================] - 0s 1ms/step - loss: 66524.6875
Epoch 40/100
21/21 [==============================] - 0s 1ms/step - loss: 66453.0391
Epoch 41/100
21/21 [==============================] - 0s 1ms/step - loss: 66379.9297
Epoch 42/100
21/21 [==============================] - 0s 1ms/step - loss: 66309.2734
Epoch 43/100
21/21 [==============================] - 0s 1ms/step - loss: 66225.9297
Epoch 44/100
21/21 [==============================] - 0s 1ms/step - loss: 66146.9609
Epoch 45/100
21/21 [==============================] - 0s 1ms/step - loss: 66062.0078
Epoch 46/100
21/21 [==============================] - 0s 1ms/step - loss: 65975.9297
Epoch 47/100
21/21 [==============================] - 0s 1ms/step - loss: 65886.7500
Epoch 48/100
21/21 [==============================] - 0s 1ms/step - loss: 65798.0391
Epoch 49/100
21/21 [==============================] - 0s 1ms/step - loss: 65703.2422
Epoch 50/100
21/21 [==============================] - 0s 1ms/step - loss: 65609.0156
Epoch 51/100
21/21 [==============================] - 0s 1ms/step - loss: 65511.9414
Epoch 52/100
21/21 [==============================] - 0s 1ms/step - loss: 65414.1445
Epoch 53/100
21/21 [==============================] - 0s 1ms/step - loss: 65314.1875
Epoch 54/100
21/21 [==============================] - 0s 4ms/step - loss: 65209.4219
Epoch 55/100
21/21 [==============================] - 0s 1ms/step - loss: 65111.6680
Epoch 56/100
21/21 [==============================] - 0s 2ms/step - loss: 65001.5664
Epoch 57/100
21/21 [==============================] - 0s 1ms/step - loss: 64899.0625
Epoch 58/100
21/21 [==============================] - 0s 1ms/step - loss: 64793.6680
Epoch 59/100
21/21 [==============================] - 0s 1ms/step - loss: 64683.6250
Epoch 60/100
21/21 [==============================] - 0s 1ms/step - loss: 64566.8477
Epoch 61/100
21/21 [==============================] - 0s 1ms/step - loss: 64456.3125
Epoch 62/100
21/21 [==============================] - 0s 1ms/step - loss: 64335.9219
Epoch 63/100
```

```
21/21 [==============================] - 0s 1ms/step - loss: 64218.6641
Epoch 64/100
21/21 [==============================] - 0s 1ms/step - loss: 64106.3203
Epoch 65/100
21/21 [==============================] - 0s 1ms/step - loss: 63981.6445
Epoch 66/100
21/21 [==============================] - 0s 1ms/step - loss: 63861.3906
Epoch 67/100
21/21 [==============================] - 0s 1ms/step - loss: 63745.6562
Epoch 68/100
21/21 [==============================] - 0s 1ms/step - loss: 63616.5195
Epoch 69/100
21/21 [==============================] - 0s 1ms/step - loss: 63492.4102
Epoch 70/100
21/21 [==============================] - 0s 1ms/step - loss: 63365.8203
Epoch 71/100
21/21 [==============================] - 0s 1ms/step - loss: 63237.7578
Epoch 72/100
21/21 [==============================] - 0s 1ms/step - loss: 63109.0391
Epoch 73/100
21/21 [==============================] - 0s 1ms/step - loss: 62982.2070
Epoch 74/100
21/21 [==============================] - 0s 1ms/step - loss: 62854.5312
Epoch 75/100
21/21 [==============================] - 0s 1ms/step - loss: 62715.9766
Epoch 76/100
21/21 [==============================] - 0s 1ms/step - loss: 62590.6172
Epoch 77/100
21/21 [==============================] - 0s 1ms/step - loss: 62448.5000
Epoch 78/100
21/21 [==============================] - 0s 1ms/step - loss: 62317.1367
Epoch 79/100
21/21 [==============================] - 0s 1ms/step - loss: 62184.4844
Epoch 80/100
21/21 [==============================] - 0s 1ms/step - loss: 62045.5000
Epoch 81/100
21/21 [==============================] - 0s 1ms/step - loss: 61901.1016
Epoch 82/100
21/21 [==============================] - 0s 1ms/step - loss: 61769.2695
Epoch 83/100
21/21 [==============================] - 0s 1ms/step - loss: 61616.1016
Epoch 84/100
21/21 [==============================] - 0s 1ms/step - loss: 61476.3438
Epoch 85/100
21/21 [==============================] - 0s 1ms/step - loss: 61335.8789
Epoch 86/100
21/21 [==============================] - 0s 1ms/step - loss: 61202.5938
Epoch 87/100
```

```
21/21 [==============================] - 0s 1ms/step - loss: 61049.6250
Epoch 88/100
21/21 [==============================] - 0s 1ms/step - loss: 60909.8555
Epoch 89/100
21/21 [==============================] - 0s 1ms/step - loss: 60765.0703
Epoch 90/100
21/21 [==============================] - 0s 1ms/step - loss: 60622.2773
Epoch 91/100
21/21 [==============================] - 0s 1ms/step - loss: 60479.5547
Epoch 92/100
21/21 [==============================] - 0s 1ms/step - loss: 60331.0625
Epoch 93/100
21/21 [==============================] - 0s 1ms/step - loss: 60190.7812
Epoch 94/100
21/21 [==============================] - 0s 1ms/step - loss: 60048.3672
Epoch 95/100
21/21 [==============================] - 0s 1ms/step - loss: 59903.8203
Epoch 96/100
21/21 [==============================] - 0s 1ms/step - loss: 59748.5664
Epoch 97/100
21/21 [==============================] - 0s 1ms/step - loss: 59607.1250
Epoch 98/100
21/21 [==============================] - 0s 1ms/step - loss: 59466.1016
Epoch 99/100
21/21 [==============================] - 0s 1ms/step - loss: 59317.1953
Epoch 100/100
21/21 [==============================] - 0s 1ms/step - loss: 59171.2148
```

[122]: `<keras.callbacks.History at 0x22a4da12830>`

1.3.3 Predicting the results of the Test set

[123]:
```
y_pred = ann.predict(X_test)
y_pred_train = ann.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
  ↪reshape(len(y_test),1)),1))
```

```
6/6 [==============================] - 0s 1ms/step
21/21 [==============================] - 0s 1ms/step
[[7.10e+00 1.49e+00]
 [5.08e+01 2.20e+01]
 [3.89e+01 1.17e+01]
 [5.02e+01 6.23e+01]
 [7.37e+00 1.26e+00]
 [3.59e+01 1.92e+01]
 [6.67e+01 8.22e+01]
```

```
[7.25e+01 2.24e+02]
[4.69e+00 1.00e+00]
[6.92e+01 5.26e+01]
[6.48e+01 2.96e+02]
[6.66e+01 6.92e+01]
[3.35e+00 1.45e+00]
[6.41e+01 1.07e+02]
[7.30e+01 7.48e+01]
[8.36e+00 5.42e+00]
[8.38e+00 1.36e+01]
[3.87e+00 1.03e+00]
[8.19e+00 3.84e+00]
[6.14e+01 2.69e+02]
[6.18e+00 8.61e-01]
[8.35e+00 6.20e+00]
[5.46e+01 1.16e+02]
[8.29e+00 5.61e+00]
[3.39e+01 1.26e+01]
[6.95e+01 1.44e+02]
[8.35e+00 4.95e+00]
[3.22e+00 1.84e+00]
[7.55e+00 1.29e+00]
[8.11e+00 2.06e+00]
[7.49e+00 8.64e-01]
[6.41e+00 1.17e+00]
[1.73e+01 7.14e+00]
[8.25e+00 2.57e+00]
[3.12e+01 1.12e+01]
[5.94e+01 3.72e+01]
[8.22e+00 2.56e+00]
[6.89e+01 7.10e+01]
[8.11e+00 3.12e+00]
[5.91e+01 2.55e+01]
[8.32e+00 5.73e+00]
[3.96e+01 3.76e+01]
[5.92e+00 1.18e+00]
[5.62e+01 4.18e+01]
[7.91e+00 1.88e+00]
[5.29e+00 1.10e+00]
[7.97e+00 1.51e+00]
[8.15e+00 5.70e+00]
[7.34e+01 7.21e+01]
[3.38e+00 1.16e+00]
[3.34e+00 1.67e+00]
[5.11e+01 6.83e+01]
[8.37e+00 6.83e+00]
[7.94e+00 1.29e+00]
[8.26e+00 4.57e+00]
```

```
[6.86e+01 3.81e+02]
[8.12e+00 3.65e+00]
[7.65e+00 1.23e+00]
[3.39e+00 1.19e+00]
[6.88e+01 6.83e+01]
[8.32e+00 4.64e+00]
[5.17e+01 9.92e+00]
[4.92e+01 2.76e+01]
[6.39e+00 1.02e+00]
[6.93e+01 5.94e+01]
[8.18e+00 1.66e+00]
[7.98e+00 1.71e+00]
[6.57e+00 7.77e-01]
[7.50e+00 1.31e+00]
[3.20e+01 1.05e+01]
[5.89e+01 1.49e+01]
[6.58e+01 2.80e+02]
[6.66e+01 4.83e+01]
[7.24e+00 1.37e+00]
[6.97e+01 6.77e+01]
[7.00e+01 1.10e+02]
[8.31e+00 3.52e+00]
[3.75e+01 1.00e+01]
[7.26e+01 1.07e+03]
[8.37e+00 6.55e+00]
[8.33e+00 6.39e+00]
[6.45e+01 2.09e+02]
[6.84e+01 1.50e+03]
[3.38e+00 8.60e-01]
[4.82e+01 2.37e+01]
[8.36e+00 4.89e+00]
[6.83e+01 6.19e+02]
[7.02e+01 6.27e+01]
[7.50e+00 2.61e+00]
[6.29e+01 5.43e+01]
[6.15e+01 5.55e+01]
[6.97e+01 9.17e+01]
[5.53e+01 2.56e+01]
[8.36e+00 8.60e+00]
[6.82e+00 9.80e-01]
[6.02e+01 1.82e+01]
[7.40e+01 2.79e+02]
[7.17e+00 1.26e+00]
[6.83e+01 1.31e+02]
[5.21e+01 1.88e+01]
[6.98e+01 4.28e+02]
[5.34e+01 3.95e+01]
[5.80e+01 1.69e+01]
```

```
[8.36e+00 5.57e+00]
[7.31e+01 4.17e+01]
[8.36e+00 7.50e+00]
[3.63e+01 1.07e+01]
[6.54e+01 3.39e+02]
[7.16e+01 6.78e+01]
[3.36e+00 8.81e-01]
[5.15e+01 1.74e+01]
[5.62e+00 2.66e+00]
[8.17e+00 2.54e+00]
[8.30e+00 3.31e+00]
[7.28e+01 7.12e+01]
[7.47e+01 7.68e+01]
[8.23e+00 3.72e+00]
[7.40e+00 1.45e+00]
[6.87e+01 4.43e+01]
[3.36e+00 9.68e-01]
[3.89e+01 1.11e+01]
[3.35e+00 1.12e+00]
[8.19e+00 1.97e+00]
[5.62e+00 1.07e+00]
[7.12e+01 7.45e+01]
[5.92e+01 8.09e+01]
[7.93e+00 1.70e+00]
[7.86e+00 1.47e+00]
[4.12e+01 3.21e+01]
[3.37e+00 1.61e+00]
[4.76e+00 6.60e-01]
[8.26e+00 1.79e+00]
[8.37e+00 6.70e+00]
[6.11e+01 2.48e+01]
[3.40e+00 1.19e+00]
[3.24e+00 9.35e-01]
[6.20e+01 1.83e+02]
[6.63e+01 3.64e+02]
[3.26e+00 1.60e+00]
[2.49e+01 1.10e+01]
[7.65e+00 1.64e+00]
[4.04e+01 1.15e+01]
[6.92e+01 2.20e+02]
[8.27e+00 4.15e+00]
[8.36e+00 8.61e+00]
[7.20e+00 8.28e-01]
[8.25e+00 6.73e+00]
[7.07e+00 1.46e+00]
[8.16e+00 6.53e+00]
[8.24e+00 2.16e+00]
[3.33e+00 1.18e+00]
```

```
[6.08e+01 4.36e+01]
[6.32e+01 4.13e+01]
[8.05e+00 2.20e+00]
[7.00e+00 1.33e+00]
[7.35e+01 1.72e+02]
[3.38e+00 9.83e-01]
[6.51e+00 1.28e+00]
[7.50e+01 2.71e+02]
[6.84e+01 7.56e+01]
[8.13e+00 2.09e+00]
[2.74e+01 2.25e+01]
[8.13e+00 1.98e+00]
[6.71e+01 5.34e+01]
[6.10e+01 3.66e+01]
[6.92e+01 3.31e+02]]
```

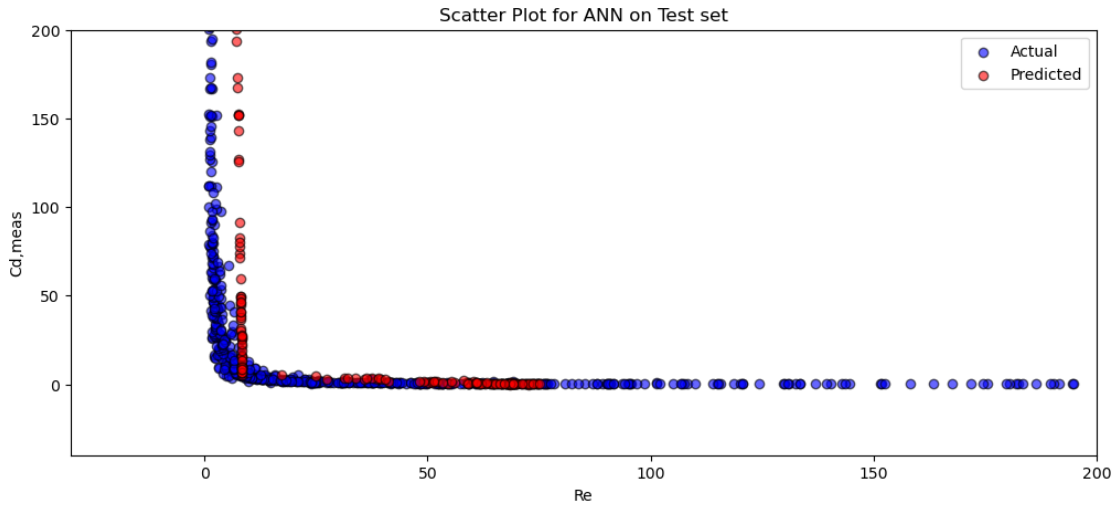
##Model Evaluation

[124]:
```
from sklearn.metrics import r2_score
print('R2-score (training set): {:.3f}'.format(r2_score(y_train, y_pred_train)))
print('R2-score (test set): {:.3f}'.format(r2_score(y_test, y_pred)))
```

```
R2-score (training set): 0.036
R2-score (test set): 0.091
```

[125]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
 ↪'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' ,edgecolors='black', alpha=0.6,␣
 ↪label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for ANN on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

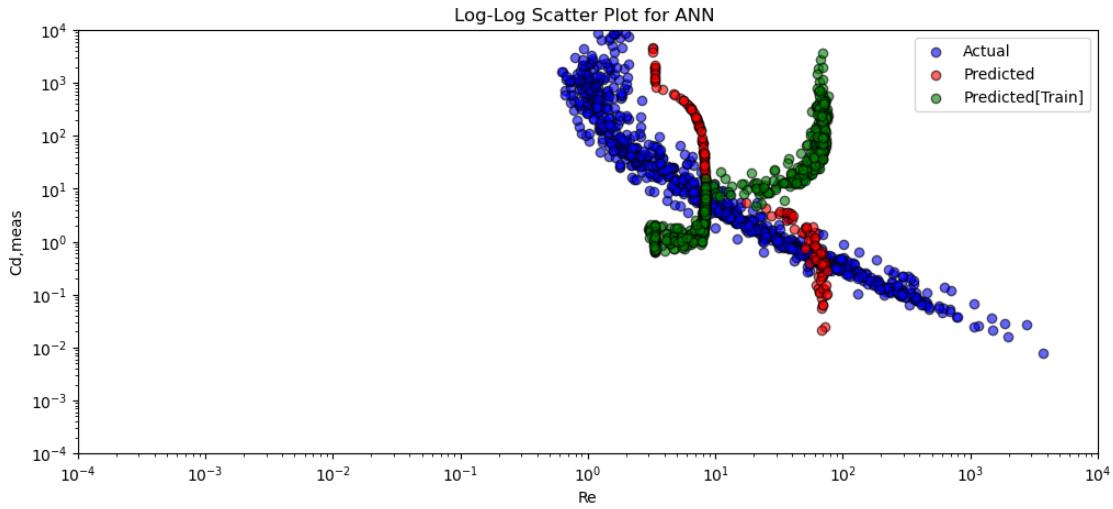
[125]:
```
<matplotlib.legend.Legend at 0x22a4da12b60>
```



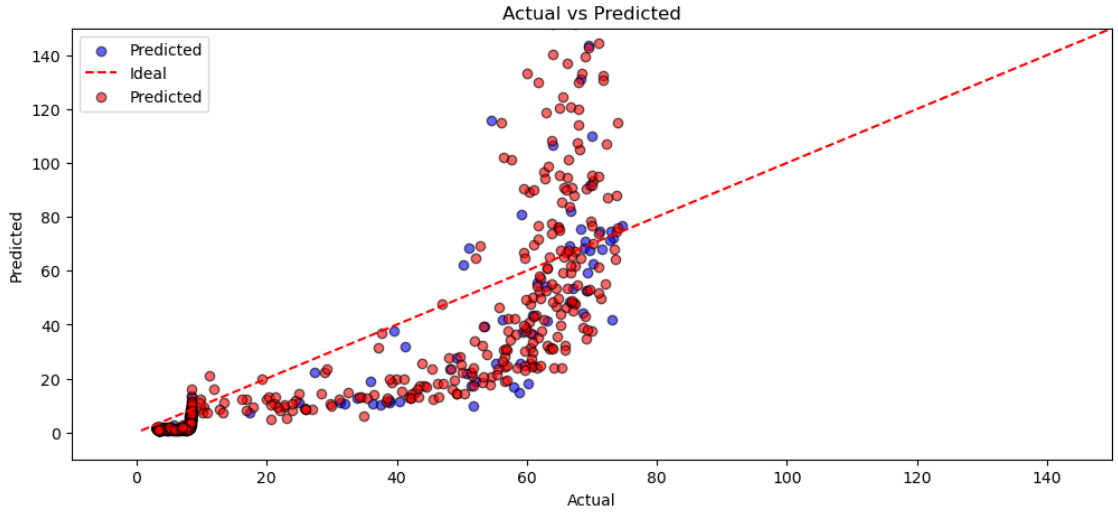
[126]:
```
print('R2-score (test set): {:.3f}'.format(r2_score(y_pred ,X_test[:,-1])))
```

```
R2-score (test set): -711.532
```

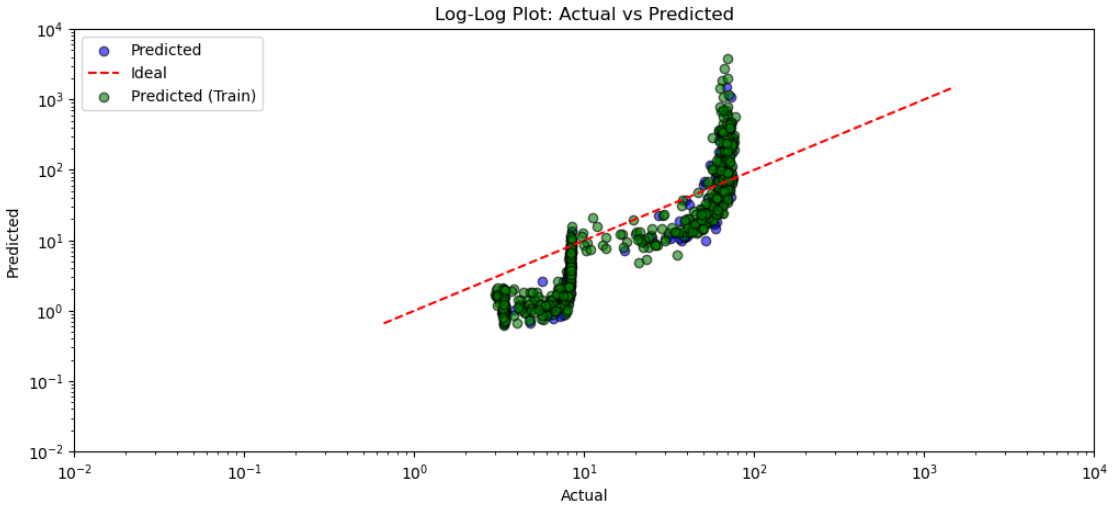
[127]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6, label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red', edgecolors='black', alpha=0.
 ↪6,label='Predicted')
plt.scatter(y_pred_train,y_train, color='green',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted[Train]')

plt.xscale('log')
plt.yscale('log')

plt.ylim(0.0001,10000)
plt.xlim(0.0001,10000)

plt.title('Log-Log Scatter Plot for ANN ')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()

plt.show()
```



[128]:
```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 150)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



[129]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.ylim(0.01,10000)
plt.xlim(0.01,10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```

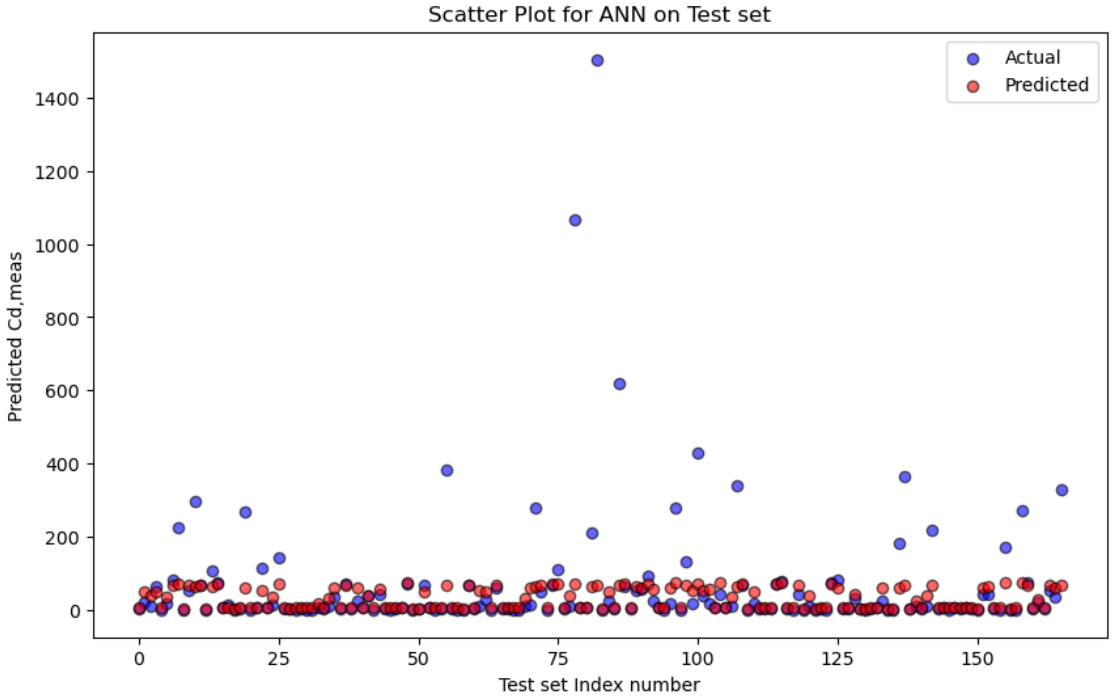


[130]:
```
plt.figure(figsize=(16, 8))

# Plot Actual vs Predicted values
plt.figure(figsize=(10, 6))
```

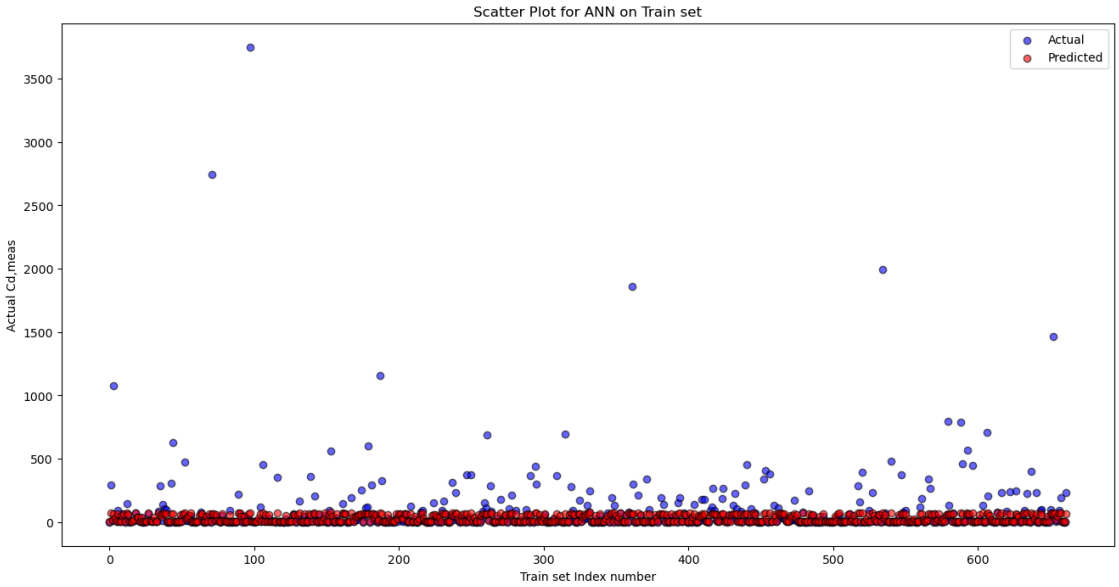
```
plt.scatter(range(len(y_test)), y_test , c = 'blue'  ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Actual')
plt.scatter(range(len(y_pred)), y_pred , c = 'red' ,edgecolors='black', alpha=0.
 ↪6, label = 'Predicted')
plt.title('Scatter Plot for ANN on Test set')
plt.xlabel('Test set Index number')
plt.ylabel('Predicted Cd,meas')
plt.legend()
```

[130]: <matplotlib.legend.Legend at 0x22a4f422410>

<Figure size 1600x800 with 0 Axes>



[131]:
```
plt.figure(figsize=(16, 8))
plt.scatter(range(len(y_train)), y_train , c = 'blue'  ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Actual')
plt.scatter(range(len(y_pred_train)), y_pred_train , c = 'red'␣
 ↪,edgecolors='black', alpha=0.6, label = 'Predicted')
plt.title('Scatter Plot for ANN on Train set')
plt.ylabel('Actual Cd,meas')
plt.xlabel('Train set Index number')
plt.legend()
plt.show()
```


Scatter Plot for ANN on Train set
Actual
Predicted
Actual Cd,meas
Train set Index number
0
500
1000
1500
2000
2500
3000
3500
0
100
200
300
400
500
600

bpfnn-gridcv

July 15, 2024

##Importing Libraries

[3]:
```
%pip install scikeras[tensorflow]
%pip install scikeras[tensorflow] pandas scikit-learn matplotlib tensorflow
%pip install --upgrade tensorflow
import keras
%pip3 install keras


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from scikeras.wrappers import KerasRegressor
```

```
Requirement already satisfied: scikeras[tensorflow] in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (0.13.0)Note: you may need
to restart the kernel to use updated packages.

Requirement already satisfied: keras>=3.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
scikeras[tensorflow]) (3.4.1)
Requirement already satisfied: scikit-learn>=1.4.2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
scikeras[tensorflow]) (1.4.2)
Requirement already satisfied: tensorflow>=2.16.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
scikeras[tensorflow]) (2.17.0)
Requirement already satisfied: absl-py in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (2.1.0)
Requirement already satisfied: numpy in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
```

```
keras>=3.2.0->scikeras[tensorflow]) (1.26.4)
Requirement already satisfied: rich in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (13.7.1)
Requirement already satisfied: namex in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (0.0.8)
Requirement already satisfied: h5py in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (3.11.0)
Requirement already satisfied: optree in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (0.12.1)
Requirement already satisfied: ml-dtypes in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (0.4.0)
Requirement already satisfied: packaging in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (23.2)
Requirement already satisfied: scipy>=1.6.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-
learn>=1.4.2->scikeras[tensorflow]) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-
learn>=1.4.2->scikeras[tensorflow]) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-
learn>=1.4.2->scikeras[tensorflow]) (3.5.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorflow>=2.16.1->scikeras[tensorflow]) (2.17.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
```

```
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (3.3.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.32.2)
Requirement already satisfied: setuptools in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (69.5.1)
Requirement already satisfied: six>=1.12.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (1.48.2)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.17.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (0.31.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->scikeras[tensorflow]) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->scikeras[tensorflow]) (2.15.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
astunparse>=1.6.0->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (0.43.0)
Requirement already satisfied: mdurl~=0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from markdown-it-
py>=2.2.0->rich->keras>=3.2.0->scikeras[tensorflow]) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-
```

```
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2024.6.2)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-
intel==2.17.0->tensorflow>=2.16.1->scikeras[tensorflow]) (2.1.3)
Requirement already satisfied: pandas in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (2.2.2)
Requirement already satisfied: scikit-learn in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (1.4.2)
Requirement already satisfied: matplotlib in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (3.8.4)
Requirement already satisfied: tensorflow in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (2.17.0)
Requirement already satisfied: scikeras[tensorflow] in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (0.13.0)
Requirement already satisfied: keras>=3.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
scikeras[tensorflow]) (3.4.1)
Requirement already satisfied: numpy>=1.22.4 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
```

```
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.6.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-learn)
(1.13.1)
Requirement already satisfied: joblib>=1.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-learn)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from scikit-learn)
(3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: tensorflow-intel==2.17.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
```

```
intel==2.17.0->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.3.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.32.2)
Requirement already satisfied: setuptools in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (69.5.1)
Requirement already satisfied: six>=1.12.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.48.2)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.17.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.31.0)
Requirement already satisfied: rich in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (13.7.1)
Requirement already satisfied: namex in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (0.0.8)
Requirement already satisfied: optree in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->scikeras[tensorflow]) (0.12.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
```

```
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
astunparse>=1.6.0->tensorflow-intel==2.17.0->tensorflow) (0.43.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2024.6.2)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.0.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->scikeras[tensorflow]) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->scikeras[tensorflow]) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from markdown-it-
py>=2.2.0->rich->keras>=3.2.0->scikeras[tensorflow]) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow)
(2.1.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: tensorflow in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (2.17.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
```

```
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.3.0)
Requirement already satisfied: packaging in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (23.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.32.2)
Requirement already satisfied: setuptools in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (69.5.1)
Requirement already satisfied: six>=1.12.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.48.2)
```

```
Requirement already satisfied: tensorboard<2.18,>=2.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
astunparse>=1.6.0->tensorflow-intel==2.17.0->tensorflow) (0.43.0)
Requirement already satisfied: rich in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (13.7.1)
Requirement already satisfied: namex in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2024.6.2)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow)
```

```
(2.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from
rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in
c:\users\91897\anaconda3\envs\myenv\lib\site-packages (from markdown-it-
py>=2.2.0->rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[3], line 4
      2 get_ipython().run_line_magic('pip', 'install scikeras[tensorflow] panda
 ↪scikit-learn matplotlib tensorflow')
      3 get_ipython().run_line_magic('pip', 'install --upgrade tensorflow')
----> 4 import keras
      5 get_ipython().run_line_magic('pip3', 'install keras')
      8 import pandas as pd

File c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\__init__.py:4
      1 import os
      3 # DO NOT EDIT. Generated by api_gen.sh
----> 4 from keras.api import DTypePolicy
      5 from keras.api import FloatDTypePolicy
      6 from keras.api import Function

File c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\api\__init__.p
 ↪8
      1 """DO NOT EDIT.
      2
      3 This file was autogenerated. Do not edit it by hand,
      4 since your modifications would be overwritten.
      5 """
----> 8 from keras.api import activations
      9 from keras.api import applications
     10 from keras.api import backend

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\api\activations\__i
 ↪py:7
      1 """DO NOT EDIT.
      2
      3 This file was autogenerated. Do not edit it by hand,
      4 since your modifications would be overwritten.
      5 """
```

```
----> 7 from keras.src.activations import deserialize
      8 from keras.src.activations import get
      9 from keras.src.activations import serialize

File c:\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\__init__.py:
 ↪1
----> 1 from keras.src import activations
      2 from keras.src import applications
      3 from keras.src import backend

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\activations\__init__.
 ↪py:3
      1 import types
----> 3 from keras.src.activations.activations import elu
      4 from keras.src.activations.activations import exponential
      5 from keras.src.activations.activations import gelu

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\activations\activations.
 ↪py:1
----> 1 from keras.src import backend
      2 from keras.src import ops
      3 from keras.src.api_export import keras_export

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\backend\__init__.
 ↪py:9
      3 if backend() == "torch":
      4     # When using the torch backend,
      5     # torch needs to be imported first, otherwise it will segfault
      6     # upon import.
      7     import torch
----> 9 from keras.src.backend.common.dtypes import result_type
     10 from keras.src.backend.common.keras_tensor import KerasTensor
     11 from keras.src.backend.common.keras_tensor import any_symbolic_tensors

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\backend\common\__init__.
 ↪py:2
      1 from keras.src.backend.common import backend_utils
----> 2 from keras.src.backend.common.dtypes import result_type
      3 from keras.src.backend.common.variables import AutocastScope
      4 from keras.src.backend.common.variables import KerasVariable

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\backend\common\dtypes.
 ↪py:5
      3 from keras.src.api_export import keras_export
```

```
      4 from keras.src.backend import config
----> 5 from keras.src.backend.common.variables import standardize_dtype
      7 BOOL_TYPES = ("bool",)
      8 INT_TYPES = (
      9     "uint8",
     10     "uint16",
   (…)
     16     "int64",
     17 )

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\backend\common\variables.
 ↪py:10
      8 from keras.src.backend.common.stateless_scope import get_stateless_scop
      9 from keras.src.backend.common.stateless_scope import in_stateless_scope
---> 10 from keras.src.utils.module_utils import tensorflow as tf
     11 from keras.src.utils.naming import auto_name
     14 class KerasVariable:

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\utils\__init__.
 ↪py:1
----> 1 from keras.src.utils.audio_dataset_utils import␣
 ↪audio_dataset_from_directory
      2 from keras.src.utils.dataset_utils import split_dataset
      3 from keras.src.utils.file_utils import get_file

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\utils\audio_dataset_utils.
 ↪py:4
      1 import numpy as np
      3 from keras.src.api_export import keras_export
----> 4 from keras.src.utils import dataset_utils
      5 from keras.src.utils.module_utils import tensorflow as tf
      6 from keras.src.utils.module_utils import tensorflow_io as tfio

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\utils\dataset_utils.
 ↪py:9
      5 from multiprocessing.pool import ThreadPool
      7 import numpy as np
----> 9 from keras.src import tree
     10 from keras.src.api_export import keras_export
     11 from keras.src.utils import io_utils

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\tree\__init__.py:
 ↪1
----> 1 from keras.src.tree.tree_api import assert_same_structure
```

```
      2 from keras.src.tree.tree_api import flatten
      3 from keras.src.tree.tree_api import is_nested

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\tree\tree_api.p
 ↪6
      3 from keras.src.utils.module_utils import optree
      5 if optree.available:
----> 6     from keras.src.tree import optree_impl as tree_impl
      7 elif dmtree.available:
      8     from keras.src.tree import dmtree_impl as tree_impl

File c:
 ↪\Users\91897\anaconda3\envs\myenv\lib\site-packages\keras\src\tree\optree_imp
 ↪py:17
     15 # Register backend-specific node classes
     16 if backend() == "tensorflow":
---> 17     from tensorflow.python.trackable.data_structures import ListWrapper
     19     optree.register_pytree_node(
     20         ListWrapper,
     21         lambda x: (x, None),
     22         lambda metadata, children: ListWrapper(list(children)),
     23         namespace="keras",
     24     )
     27 def is_nested(structure):

ModuleNotFoundError: No module named 'tensorflow.python.trackable'
```

##Importing Dataset and Splitting

[ ]:
```
# Load dataset
file_path = 'drag_coef.csv'
data = pd.read_csv(file_path)

X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split the data into 80% training and 20% test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

##Applying BPFNN and GridsearchCV for Optimization

[ ]:
```
# Define the BPFNN model
def create_model(optimizer='adam', neurons=5):
    model = Sequential()
    # Specify the input shape in the first layer
    model.add(Dense(neurons, input_dim=X_train.shape[1], activation='relu'))
```

```
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer=optimizer)
    return model

# Wrap the model using KerasRegressor
model = KerasRegressor(model=create_model, verbose=0,neurons=5)

# Define the grid search parameters
param_grid = {
    'batch_size': [10, 20],
    'epochs': [50, 100],
    'optimizer': ['SGD', 'Adam'],
    'neurons': [5, 10]
}

# Perform GridSearchCV to find the best parameters
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3,n_jobs=-1)
grid_result = grid.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-
packages/joblib/externals/loky/backend/fork_exec.py:38: RuntimeWarning:
os.fork() was called. os.fork() is incompatible with multithreaded code, and JAX
is multithreaded, so this will likely lead to a deadlock.
  pid = os.fork()
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

##Tuning the Hyperparameters

[ ]:
```
# Print the best parameters and best score
print(f"Best: {grid_result.best_score_} using {grid_result.best_params_}")

# Train the model using the best parameters
best_model = grid_result.best_estimator_.model_
best_model.compile(loss='mean_squared_error', optimizer=grid_result.
 ↪best_params_['optimizer'])
history = best_model.fit(X_train, y_train, validation_split=0.2,␣
 ↪epochs=grid_result.best_params_['epochs'], batch_size=grid_result.
 ↪best_params_['batch_size'], verbose=0)
```

```
Best: 0.020839630388728898 using {'batch_size': 10, 'epochs': 100, 'neurons':
10, 'optimizer': 'Adam'}
```

##Predictions on the Test set

[ ]:
```
# Predict using the test set
import numpy as np
y_pred = best_model.predict(X_test)
y_pred_train=best_model.predict(X_train)
np.set_printoptions(precision=3)
print(y_pred)
```

```
8/8             0s 9ms/step
19/19             0s 1ms/step
[[ -8.829]
 [ 86.671]
 [ 85.654]
 [  6.574]
 [ 90.323]
 [ 87.   ]
 [ 98.391]
 [ 88.432]
 [109.327]
 [ 85.565]
 [ 76.066]
 [ 98.301]
 [109.151]
 [ 80.416]
 [ 86.15 ]
 [ 80.049]
 [ 75.955]
 [109.902]
 [ 75.185]
 [  4.237]
 [ 60.404]
 [ 95.042]
 [ 80.454]
 [ 86.679]
 [ 80.383]
 [ 93.565]
 [ 87.809]
 [ 37.93 ]
 [102.665]
 [ 82.626]
 [ 80.371]
 [102.827]
 [ 98.898]
 [ 95.273]
 [ 83.798]
 [  2.416]
 [ 65.882]
 [ 81.515]
```

```
[ 94.859]
[ 95.254]
[ 23.792]
[ 99.063]
[ 85.114]
[ 48.007]
[ 77.94 ]
[ 95.121]
[ 91.937]
[ 79.873]
[ 41.149]
[ 84.91 ]
[ 85.498]
[ 76.321]
[ 82.467]
[110.226]
[ 14.077]
[  9.112]
[ 84.285]
[ 44.83 ]
[  0.84 ]
[ 82.551]
[ 60.652]
[ 73.864]
[ 81.821]
[ 88.429]
[ 82.072]
[ 80.274]
[ 89.944]
[ 87.613]
[ 71.322]
[ 76.884]
[ 91.292]
[ 95.272]
[107.152]
[ 80.286]
[ 94.752]
[ 77.727]
[ 80.489]
[ 71.456]
[ 85.3  ]
[ 86.005]
[ 88.549]
[ 11.169]
[-10.975]
[ 80.249]
[ 79.742]
[ 69.453]
```

[80.378]
[94.458]
[95.182]
[85.828]
[17.686]
[87.982]
[104.328]
[98.033]
[81.887]
[92.16]
[91.795]
[107.707]
[35.29]
[93.06]
[100.937]
[77.388]
[72.477]
[89.947]
[86.17]
[95.195]
[75.983]
[109.821]
[-1.244]
[83.693]
[88.405]
[72.056]
[93.876]
[69.597]
[83.373]
[100.726]
[74.365]
[85.348]
[84.723]
[75.449]
[88.392]
[86.164]
[79.686]
[86.922]
[90.912]
[76.156]
[96.851]
[33.21]
[92.285]
[87.846]
[67.425]
[87.058]
[83.247]
[54.043]

```
[ 72.339]
[ 93.246]
[ 80.601]
[ 88.433]
[ 79.711]
[ 76.26 ]
[ 34.262]
[ -4.856]
[ 79.932]
[ 86.852]
[ 81.951]
[ 87.759]
[ 15.343]
[ 85.328]
[ 28.501]
[ 83.481]
[ 86.037]
[ 93.014]
[ 78.595]
[ 88.426]
[ 82.628]
[ 85.499]
[ 14.686]
[ 93.796]
[ 86.109]
[ 73.839]
[ 91.42 ]
[-10.474]
[ 89.589]
[ 89.682]
[ 80.597]
[ 76.797]
[ 16.994]
[ 73.706]
[101.534]
[ 92.877]
[ -2.23 ]
[101.438]
[ 71.823]
[ 93.993]
[ 97.723]
[ 74.435]
[ 89.824]
[ 60.136]
[ 84.141]
[ 84.231]
[ 81.091]
[ 87.924]
```

```
[ 98.627]
[  4.182]
[ 79.564]
[ 64.896]
[ 95.028]
[ 59.29 ]
[ 70.566]
[-11.713]
[ 71.465]
[ 88.355]
[  4.504]
[ 80.486]
[ 99.833]
[ 93.082]
[ 95.024]
[ -2.293]
[ 90.831]
[ 96.29 ]
[ -0.964]
[ 81.657]
[ -1.697]
[ 85.735]
[  1.286]
[ 80.071]
[ 90.563]
[ 80.848]
[ 88.196]
[ 92.021]
[ 90.334]
[ 84.24 ]
[ 86.74 ]
[ 92.633]
[ 98.708]
[ 95.254]
[ -2.033]
[ 98.994]
[ 82.362]
[ 81.151]
[ 95.109]
[ 82.029]
[  1.608]
[ 87.695]
[ 92.975]
[ 98.922]
[ 88.428]
[ 90.244]
[ 90.305]
[ 84.349]
```

```
[ 69.895]
[ 88.929]
[ 86.42 ]
[  9.79 ]
[ 85.078]
[ 98.665]
[  1.667]
[ 94.816]
[ 98.602]
[ 56.573]
[ 86.131]
[  3.844]
[ 75.139]
[ 80.167]
[ 78.97 ]
[ 95.226]
[ 83.622]
[104.055]
[ 84.806]]
```

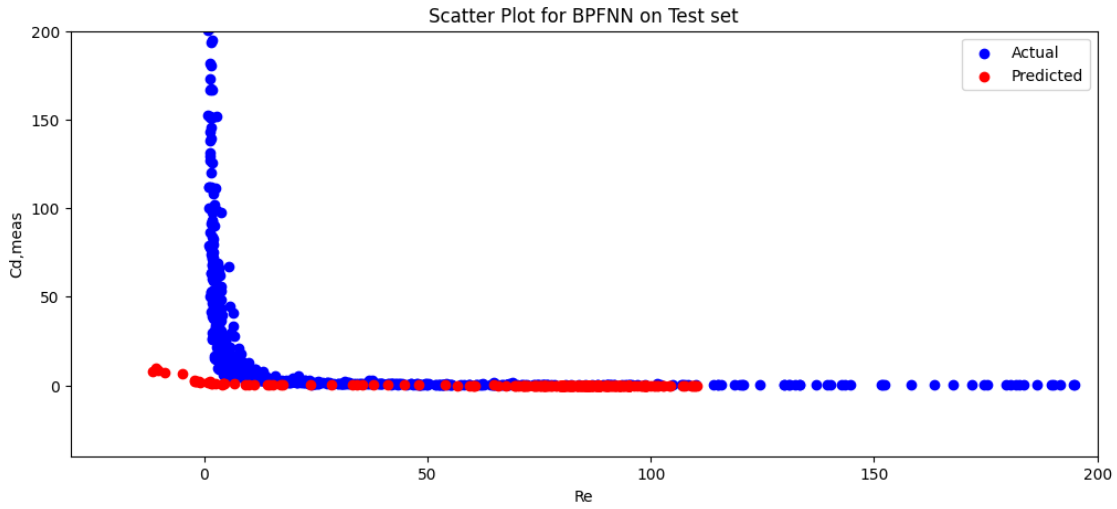
##Model Evaluation

[ ]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mse_train = mean_squared_error(y_train,y_pred_train)
r2_train = r2_score(y_train,y_pred_train)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R2 Score: {r2}")
print("R2 Train:",r2_train)
print("MSE Train:", mse_train)
```

```
Mean Squared Error: 89442.81856305864
R2 Score: 0.013524188180574992
R2 Train: 0.03273526048473152
MSE Train: 37708.93668997624
```

[ ]:
```
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  , label = 'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red', label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for BPFNN on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

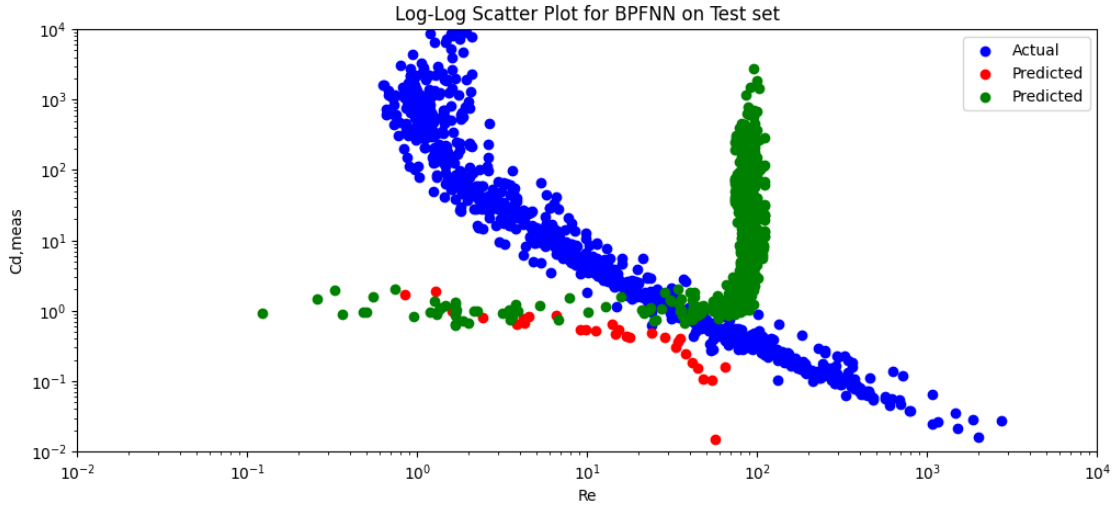
[ ]: <matplotlib.legend.Legend at 0x78b6f4eaab60>



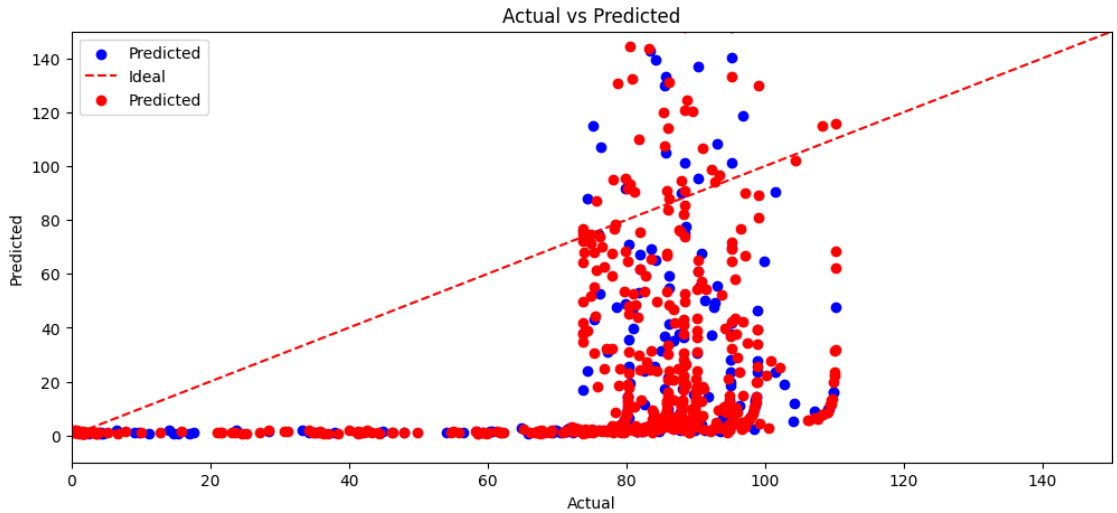
[ ]:
```
r2_fit = r2_score(y_pred ,X_test[:,-1])
print(r2_fit)
```

-6.505231863516917

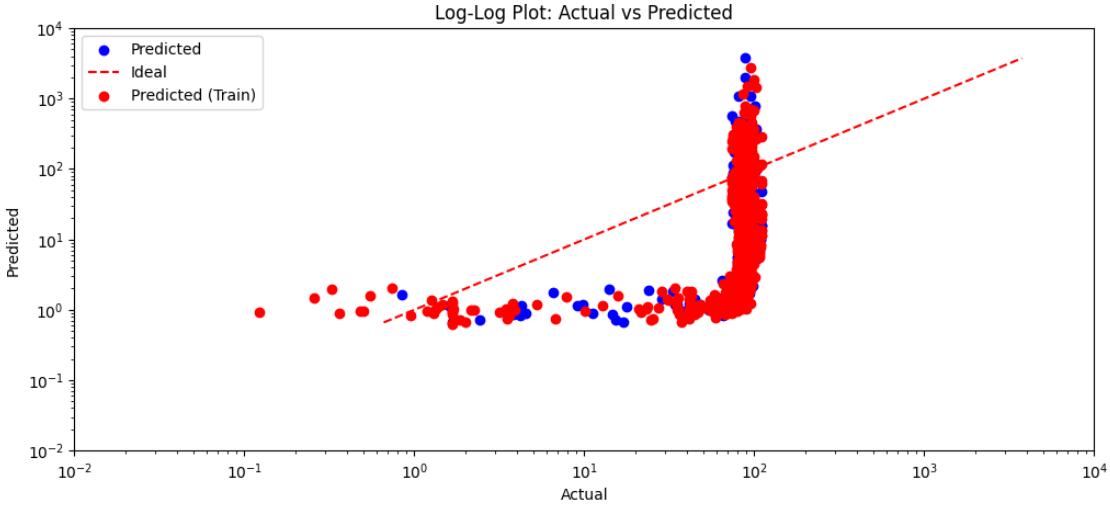
[ ]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue', label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red', label='Predicted')
plt.scatter(y_pred_train,y_train, color='green', label='Predicted')

plt.xscale('log')
plt.yscale('log')

plt.ylim(0.01, 10000)
plt.xlim(0.01, 10000)

plt.title('Log-Log Scatter Plot for BPFNN')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()

plt.show()
```



[ ]:
```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue', label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train,y_train, color='red', label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0, 150)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



[ ]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y_pred, y_test, color='blue',label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train, y_train, color='red', label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0.01, 10000)
plt.ylim(0.01, 10000)

plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```

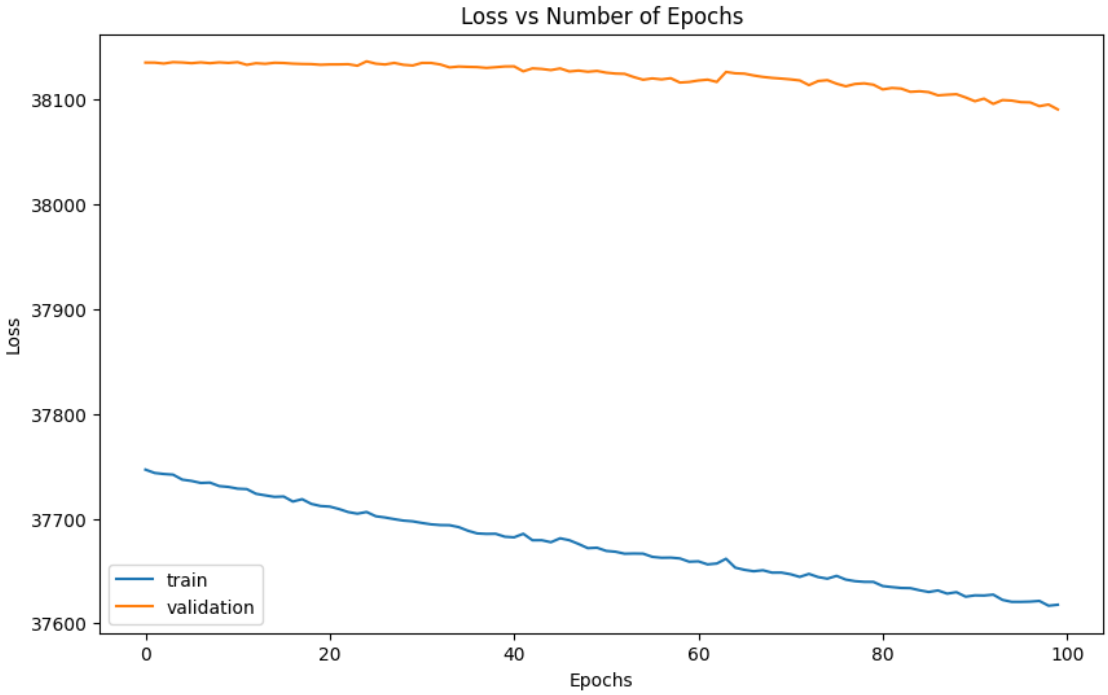


#Scatter Plot of Results

##Loss Functiion vs Epochs

[ ]:
```
# Plot the loss vs number of epochs
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

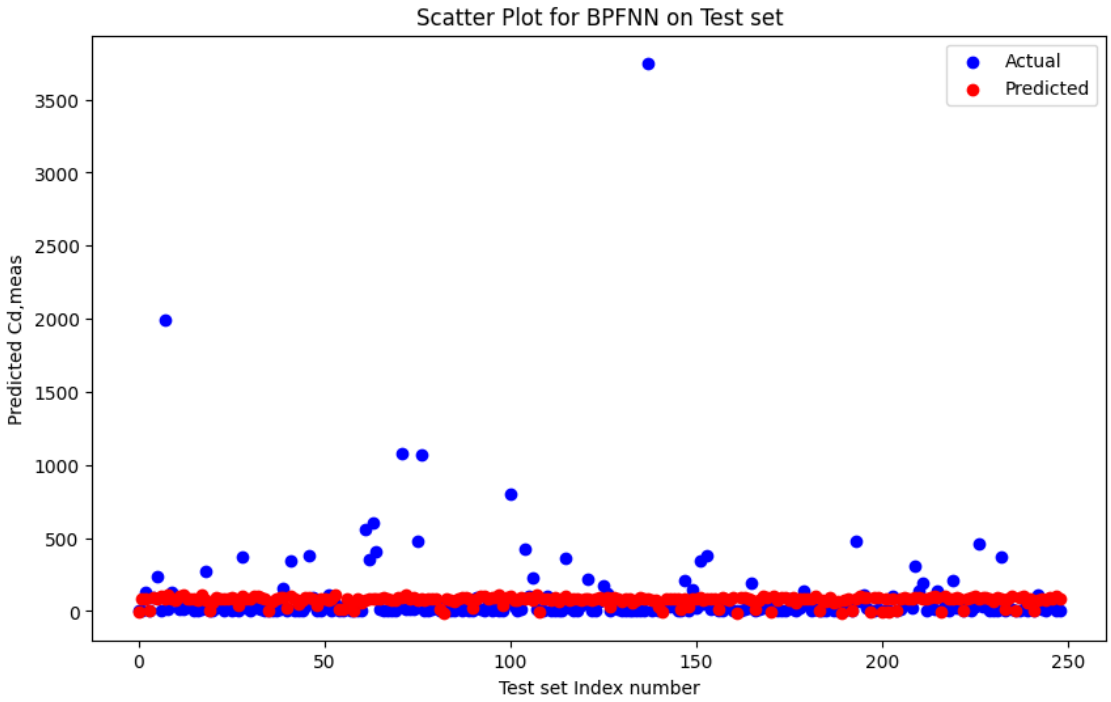
```
plt.legend()
plt.title('Loss vs Number of Epochs')

plt.show()
```



##Scatter Plot of Test set

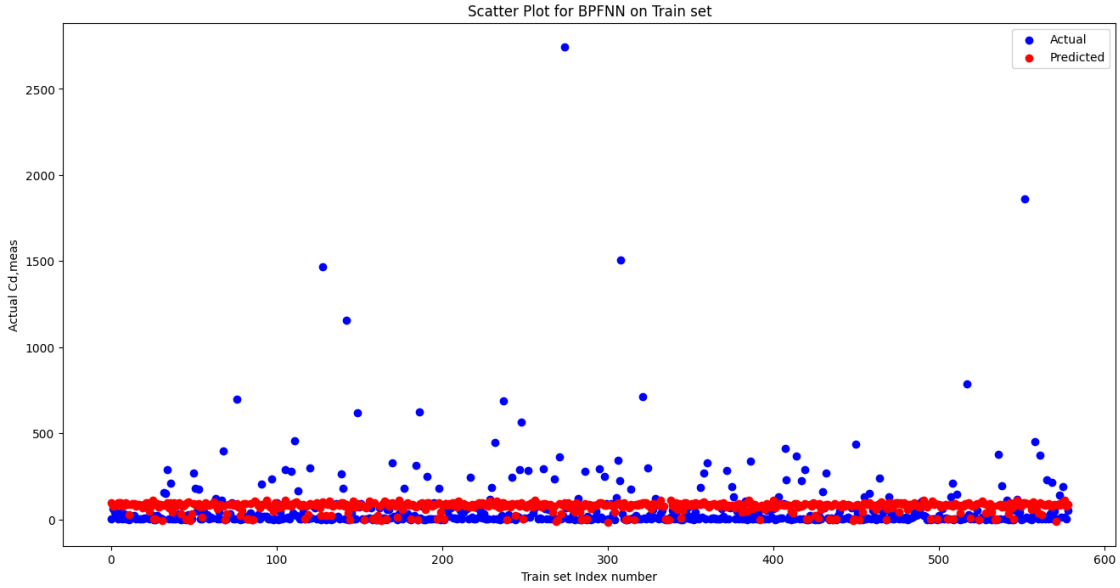
[ ]:
```
plt.figure(figsize=(16, 8))

# Plot Actual vs Predicted values
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test , c = 'blue'  ,label = 'Actual')
plt.scatter(range(len(y_pred)), y_pred , c = 'red' ,label = 'Predicted')
plt.title('Scatter Plot for BPFNN on Test set')
plt.xlabel('Test set Index number')
plt.ylabel('Predicted Cd,meas')
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x78b6e74404c0>

<Figure size 1600x800 with 0 Axes>
```



##Scatter Plot on Training set

[ ]:
```
plt.figure(figsize=(16, 8))
plt.scatter(range(len(y_train)), y_train , c = 'blue'  , label = 'Actual')
plt.scatter(range(len(y_pred_train)), y_pred_train , c = 'red' , label =␣
 ↪'Predicted')
plt.title('Scatter Plot for BPFNN on Train set')
plt.ylabel('Actual Cd,meas')
plt.xlabel('Train set Index number')
plt.legend()
plt.show()
```


Scatter Plot for BPFNN on Train set
Actual
Predicted
Actual Cd,meas
Train set Index number
0
500
1000
1500
2000
2500
0
100
200
300
400
500
600

rbfnn-gridcv

July 15, 2024

##Importing the Libraries

[157]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

##Importing the Dataset and datasplitting

[158]:
```
dataset = pd.read_csv('drag_coef.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values


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 = 42)
```

##Feature Scaling

[159]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

##Radial basis function Neural Network + GridSearchCV with K means for intialization of centers for RBF

[160]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import  mean_squared_error,make_scorer
from sklearn.base import BaseEstimator, ClassifierMixin
from sklearn.cluster import KMeans

#Defining the custom scored for GridSearchCV
def custom_scorer(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    return mse

#Defining the parameter grid
param_grid = {
```

```
    'num_centers': [50, 100, 150],
    'sigma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 10.0]
}

class RBFNN(BaseEstimator, ClassifierMixin):
    def __init__(self, num_centers=100, sigma=1.0):
        self.num_centers = num_centers
        self.sigma = sigma

    def fit(self, X, y):
        # Use K-means for center initialization
        kmeans = KMeans(n_clusters=self.num_centers, random_state=42)
        kmeans.fit(X)
        self.centers = kmeans.cluster_centers_

        # Compute RBF activations
        self.rbf_activations = self._rbf(X)

        # Solve for weights
        self.weights = np.linalg.pinv(self.rbf_activations) @ y
        return self

    def predict(self, X):
        rbf_activations = self._rbf(X)
        return rbf_activations @ self.weights

    def _rbf(self, X):
        distances = np.sum((X[:, np.newaxis, :] - self.centers[np.newaxis, :, :
 ↪]) ** 2, axis=2)
        return np.exp(-distances / (2 * self.sigma ** 2))

    def get_params(self, deep=True): # Add get_params method
        return {"num_centers": self.num_centers, "sigma": self.sigma}

    def set_params(self, **parameters): # Add set_params method
        for parameter, value in parameters.items():
            setattr(self, parameter, value)
        return self

#Creating the RBFNN model
rbfnn = RBFNN(num_centers=100)

#Creating the GridSearchCV object
grid_search = GridSearchCV(rbfnn,
                           param_grid,
                           cv = 10,
                           n_jobs = -1)
```

```
#Fitting the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

#Getting the best parameters
best_parameters = grid_search.best_params_
print("Best Parameters:", best_parameters)
```

```
Best Parameters: {'num_centers': 50, 'sigma': 0.1}
```

```
c:\Users\91897\anaconda3\envs\myenv\lib\site-
packages\sklearn\model_selection\_search.py:1051: UserWarning: One or more of
the test scores are non-finite: [nan nan nan nan nan nan nan nan nan nan nan nan
nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan]
  warnings.warn(
c:\Users\91897\anaconda3\envs\myenv\lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

##Creating and training of RBFNN model on Training set

[161]:
```
num_centers = 50  # Tuning parameter
rbfnn = RBFNN(num_centers=num_centers)
rbfnn.fit(X_train, y_train)
```

```
c:\Users\91897\anaconda3\envs\myenv\lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

[161]:
```
RBFNN(num_centers=50)
```

##Predictions on the Training set and Training set

[162]:
```
y_pred = rbfnn.predict(X_test)
y_pred_train = rbfnn.predict(X_train)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
 ↪reshape(len(y_test),1)),1))
```

```
[[ 1.19e+01  1.77e+00]
 [ 2.50e+01  3.63e+00]
 [ 9.63e+01  1.33e+02]
 [ 4.13e+01  1.75e+00]
 [ 1.13e+02  9.56e+01]
```

```
[ 8.73e+01  2.36e+02]
[ 1.25e+02  2.16e+00]
[ 8.85e+01  1.99e+03]
[ 3.99e+01  9.83e+00]
[ 9.69e+01  1.30e+02]
[ 1.10e+02  7.45e+01]
[ 1.14e+02  8.66e+00]
[ 3.90e+01  9.84e+00]
[ 8.06e+01  3.55e+01]
[ 9.22e+01  5.47e+01]
[ 7.50e+01  4.30e+00]
[ 2.28e+01  1.89e+00]
[ 4.28e+01  1.59e+01]
[ 1.29e+02  2.71e+02]
[ 1.90e+01  1.14e+00]
[-2.47e+01  1.33e+00]
[ 1.58e+02  2.00e+01]
[ 8.11e+01  7.10e+01]
[ 5.92e+01  2.75e+00]
[ 8.01e+01  2.48e+01]
[ 3.83e+01  1.70e+00]
[ 7.85e+01  8.23e+00]
[ 1.47e+01  9.68e-01]
[ 2.06e+02  3.67e+02]
[ 1.04e+02  2.41e+01]
[ 7.94e+01  6.60e+00]
[ 2.05e+02  1.92e+01]
[ 1.26e+02  2.36e+01]
[ 1.51e+02  1.01e+01]
[ 6.08e+01  1.79e+00]
[ 4.42e+00  7.11e-01]
[-8.39e+00  8.28e-01]
[ 1.01e-02  1.51e+00]
[ 1.55e+02  1.32e+01]
[ 1.62e+02  1.52e+02]
[ 3.83e+01  1.91e+00]
[ 1.29e+02  3.41e+02]
[ 7.77e+01  4.61e+00]
[ 1.14e+01  1.18e+00]
[ 3.96e+01  1.62e+00]
[ 1.60e+02  2.79e+01]
[ 1.43e+02  3.76e+02]
[ 7.43e+01  9.17e+01]
[ 3.10e+01  1.21e+00]
[ 3.21e+01  2.38e+00]
[ 9.60e+01  3.68e+01]
[ 9.82e+01  1.07e+02]
[ 5.08e+01  2.71e+00]
```

```
[ 4.44e+01  4.75e+01]
[ 3.04e+01  2.00e+00]
[ 8.12e+01  1.14e+00]
[ 1.05e+02  6.51e+01]
[ 4.37e+00  1.43e+00]
[-1.47e+02  1.67e+00]
[ 1.07e+01  1.80e+00]
[-5.34e+01  1.45e+00]
[ 6.04e+01  5.63e+02]
[ 9.82e+01  3.56e+02]
[ 8.85e+01  6.02e+02]
[ 1.01e+02  4.03e+02]
[ 7.85e+01  1.43e+01]
[ 5.33e+01  1.97e+00]
[ 7.82e+01  7.12e+00]
[ 4.19e+00  1.45e+00]
[ 3.71e+01  1.98e+00]
[ 1.31e+02  5.02e+01]
[ 1.62e+02  1.08e+03]
[ 7.09e+01  9.08e+00]
[ 7.86e+01  1.44e+01]
[ 1.53e+02  1.13e+01]
[ 6.99e+01  4.77e+02]
[ 8.16e+01  1.07e+03]
[ 3.18e+01  1.57e+00]
[ 8.02e+01  3.04e+00]
[ 9.01e+01  9.92e+00]
[ 8.88e+01  7.76e+01]
[ 8.26e+01  8.90e-01]
[ 2.35e+00  1.73e+00]
[ 7.81e+01  1.31e+01]
[ 7.08e+01  5.61e+00]
[-6.86e+00  1.29e+00]
[ 8.00e+01  2.56e+01]
[ 1.48e+02  8.04e+00]
[ 1.61e+02  4.18e+01]
[ 4.93e+01  3.47e+00]
[-2.84e+01  1.09e+00]
[ 8.61e+01  9.01e+01]
[ 1.85e+02  1.19e+01]
[ 1.09e+02  6.89e+00]
[ 9.88e+01  5.32e+01]
[ 9.82e+01  1.97e+00]
[ 1.40e+02  1.43e+01]
[ 3.25e+01  6.73e+00]
[-3.07e+01  1.19e+00]
[ 1.59e+02  5.55e+01]
[ 1.68e+02  7.98e+02]
```

```
[ 7.07e+01  3.12e+01]
[-2.91e+01  1.31e+00]
[ 1.06e+02  1.08e+01]
[ 9.25e+01  4.28e+02]
[ 1.61e+02  1.01e+02]
[ 1.15e+02  2.24e+02]
[ 4.23e+01  1.59e+01]
[-4.17e+01  1.28e+00]
[ 1.76e+01  2.20e+00]
[ 8.81e+01  1.01e+02]
[ 4.65e+01  2.63e+00]
[ 1.37e+02  6.09e+00]
[-6.37e+01  2.07e+00]
[ 5.58e+01  3.72e+00]
[ 1.62e+02  3.59e+02]
[ 9.26e+01  2.39e+01]
[ 3.72e+01  3.64e+00]
[ 3.44e+01  1.81e+00]
[ 1.28e+02  4.29e+01]
[ 8.78e+01  7.51e+01]
[ 9.24e+01  2.20e+02]
[ 7.00e+01  5.57e+00]
[ 6.49e+01  4.74e+00]
[ 1.24e+02  6.77e+01]
[ 1.07e+02  1.72e+02]
[ 1.39e+02  1.19e+02]
[ 2.14e+01  1.80e+00]
[ 1.49e+02  3.72e+01]
[ 7.90e+01  8.52e+00]
[-5.95e+01  1.46e+00]
[ 6.43e+01  3.65e+00]
[ 5.43e+01  2.48e+00]
[-2.07e+01  1.00e+00]
[-1.46e+01  1.31e+00]
[ 5.71e+01  2.14e+00]
[ 6.09e+01  2.29e+00]
[ 8.85e+01  3.75e+03]
[ 6.15e+01  2.06e+00]
[ 9.94e+01  5.26e+01]
[ 2.00e+01  1.21e+00]
[-7.11e+01  1.46e+00]
[ 7.48e+01  4.91e+01]
[ 8.67e+01  3.50e+01]
[ 9.94e+01  6.72e+01]
[ 7.58e+01  2.06e+00]
[ 4.90e+01  7.26e-01]
[ 9.92e+01  2.10e+02]
[ 5.15e+01  1.44e+00]
```

```
[ 1.08e+02  1.43e+02]
[ 9.06e+01  2.14e+01]
[ 1.59e+02  3.40e+02]
[ 6.58e+01  4.75e+01]
[ 8.84e+01  3.81e+02]
[ 1.03e+02  1.15e+01]
[ 9.41e+01  1.74e+01]
[-2.51e+01  8.60e-01]
[ 9.89e+01  3.46e+00]
[ 9.16e+01  4.13e+01]
[ 6.02e+01  7.60e+01]
[ 6.50e+01  1.86e+00]
[ 1.17e+00  1.57e+00]
[ 9.99e+01  6.76e+00]
[ 1.02e+02  1.53e+01]
[ 8.28e+01  1.94e+01]
[ 8.12e+01  1.95e+02]]
```

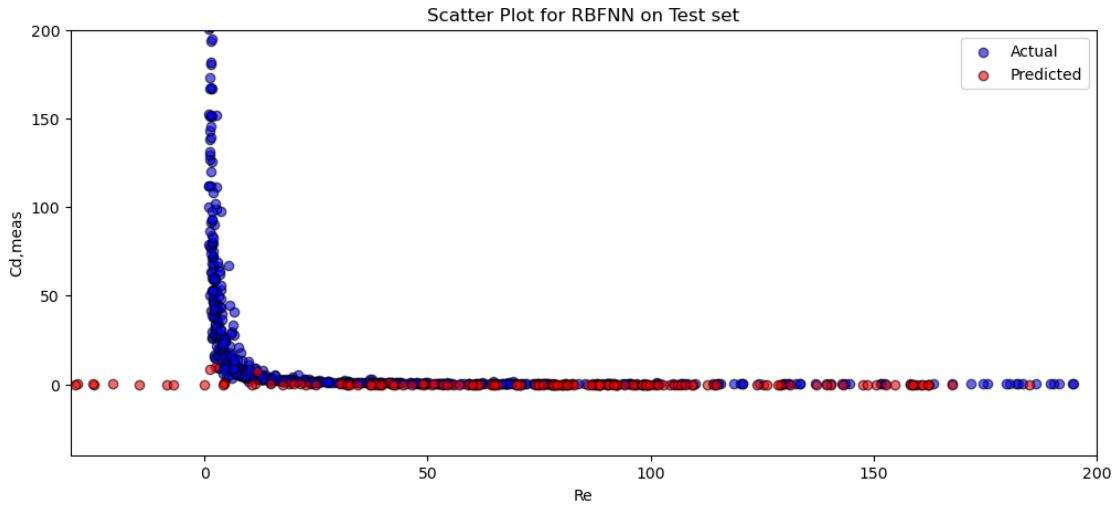
##Evaluatig the Model Performance

[163]:
```
from sklearn.metrics import mean_squared_error, r2_score
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R-squared Score:", r2_score(y_test, y_pred))
print("R-squared Score on Training set:", r2_score(y_train, y_pred_train))
print("Mean Squared Error on Training set:", mean_squared_error(y_train,␣
 ↪y_pred_train))
```

```
Mean Squared Error: 127724.51529101482
R-squared Score: 0.3210613740318629
R-squared Score on Training set: 0.39290534127541415
Mean Squared Error on Training set: 31946.8775140433
```

[164]:
```
plt.figure(figsize=(12,5))
plt.scatter( y ,X[:,-1], c = 'blue'  ,edgecolors='black', alpha=0.6, label =␣
 ↪'Actual')
plt.scatter( y_pred ,X_test[:,-1], c = 'red' ,edgecolors='black', alpha=0.6,␣
 ↪label = 'Predicted')
##plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.ylim(-40, 200)
plt.xlim(-30, 200)
plt.title('Scatter Plot for RBFNN on Test set')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()
```

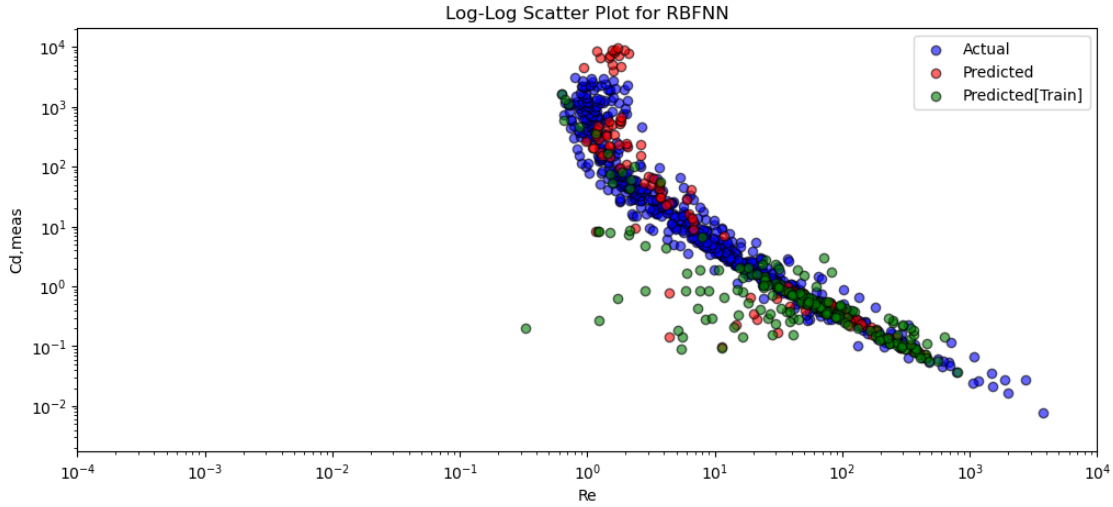
[164]:
```
<matplotlib.legend.Legend at 0x18f6bc2b160>
```



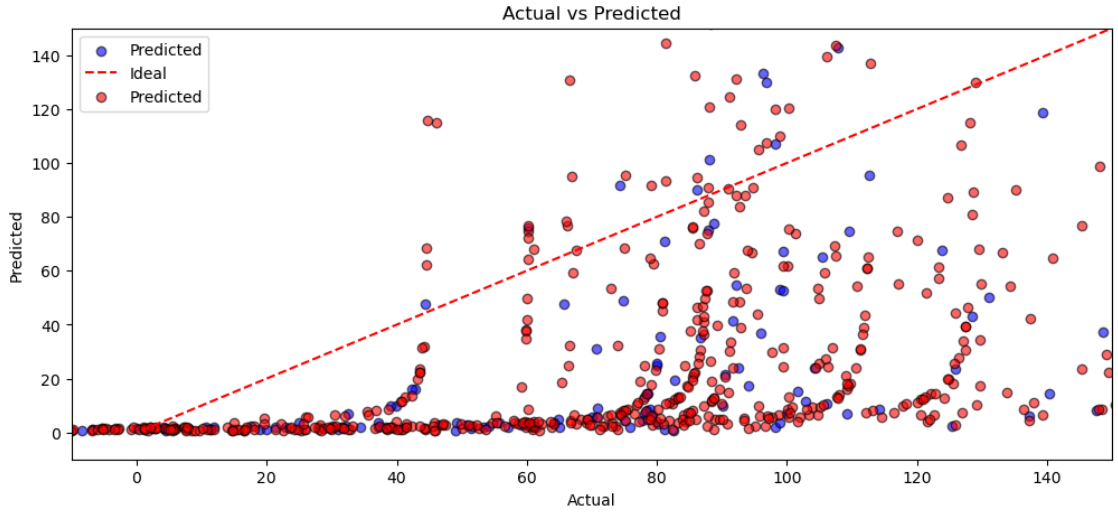
[165]:
```
print("R-squared Score:", r2_score(y_pred ,X_test[:,-1]))
```

```
R-squared Score: -1.6587983623804599
```

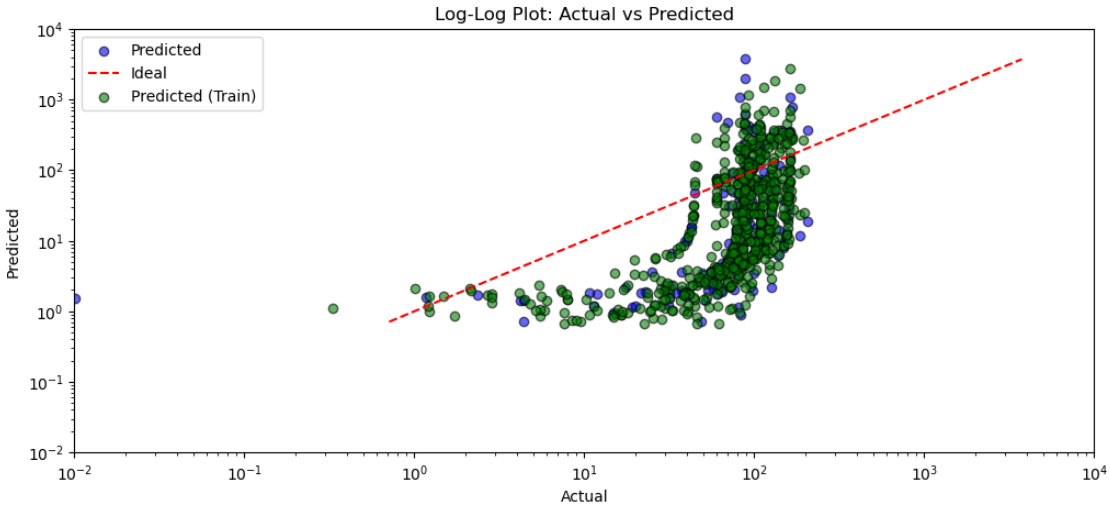
[166]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y, X[:, -1], c='blue',edgecolors='black', alpha=0.6, label='Actual')
plt.scatter(y_pred, X_test[:, -1], c='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.scatter( y_pred_train ,X_train[:,-1], c = 'green' ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Predicted[Train]')

plt.xscale('log')
plt.yscale('log')
plt.xlim(0.0001,10000)

plt.title('Log-Log Scatter Plot for RBFNN')
plt.ylabel('Cd,meas')
plt.xlabel('Re')
plt.legend()

plt.show()
```



[167]:
```
plt.figure(figsize=(12,5))
plt.scatter(y_pred,y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train,y_train, color='red',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(-10, 150)
plt.ylim(-10, 150)
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



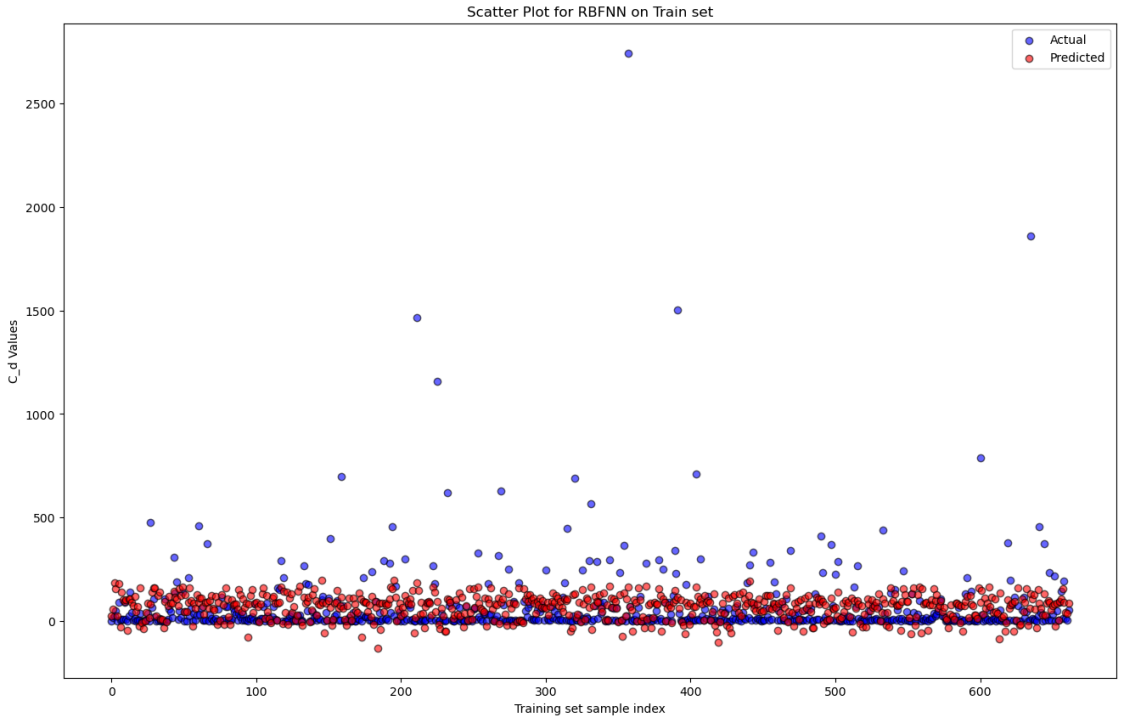
[168]:
```
plt.figure(figsize=(12, 5))

plt.scatter(y_pred, y_test, color='blue',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',␣
 ↪label='Ideal')
plt.scatter(y_pred_train, y_train, color='green',edgecolors='black', alpha=0.6,␣
 ↪label='Predicted (Train)')

plt.xscale('log')
plt.yscale('log')

plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.xlim(0.01, 10000)
plt.ylim(0.01, 10000)


plt.title('Log-Log Plot: Actual vs Predicted')
plt.legend()

plt.show()
```



##Plotting the results of Training set

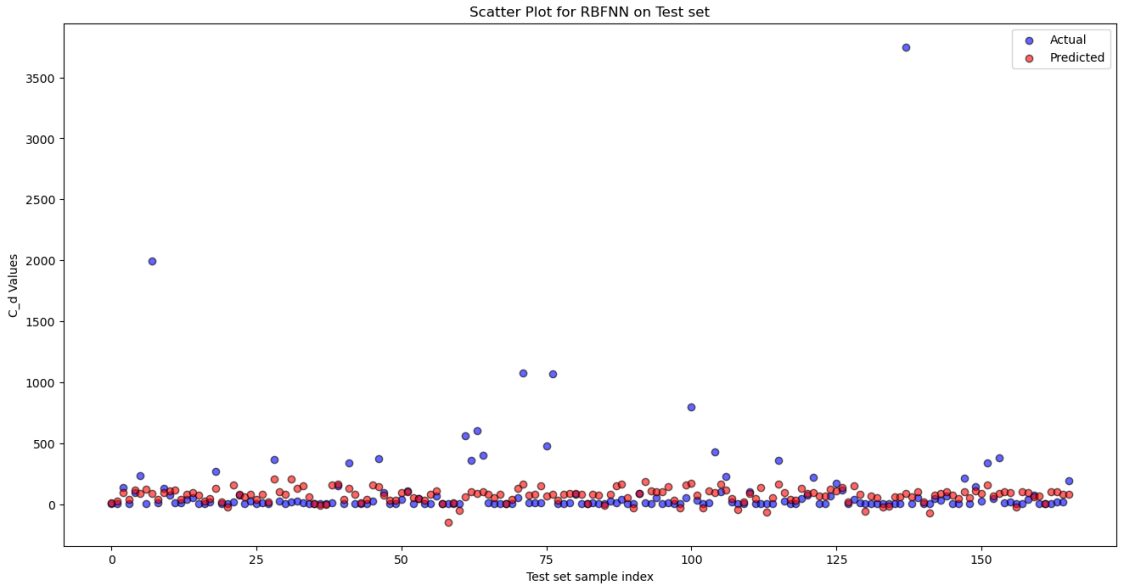
[169]:
```
plt.figure(figsize=(16, 10))
```

```
plt.scatter(range(len(y_train)), y_train , c = 'blue'  ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Actual')
plt.scatter(range(len(y_pred_train)), y_pred_train , c = 'red'␣
 ↪,edgecolors='black', alpha=0.6, label = 'Predicted')
plt.title('Scatter Plot for RBFNN on Train set')
plt.xlabel('Training set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```



##Plotting the results on Testset

[170]:
```
plt.figure(figsize=(16, 8))
plt.scatter(range(len(y_test)), y_test , c = 'blue'  ,edgecolors='black',␣
 ↪alpha=0.6, label = 'Actual')
plt.scatter(range(len(y_pred)), y_pred , c = 'red' ,edgecolors='black', alpha=0.
 ↪6, label = 'Predicted')
plt.title('Scatter Plot for RBFNN on Test set')
plt.xlabel('Test set sample index')
plt.ylabel('C_d Values')
plt.legend()
plt.show()
```


Scatter Plot for RBFNN on Test set
Actual
Predicted
C_d Values
Test set sample index
0
500
1000
1500
2000
2500
3000
3500
0
25
50
75
100
125
150