CEE 498 Applied Machine Learning - HW6

Rini Jasmine Gladstone (rjg7) and Maksymilian Podraza (podraza3)

May 7 2020

1 Problem 1

We built a linear regression of the log of the concentration against the log of time.

```
X = np.log(brunhild['Sulfate'])
y = np.log(brunhild['Hours'])
```

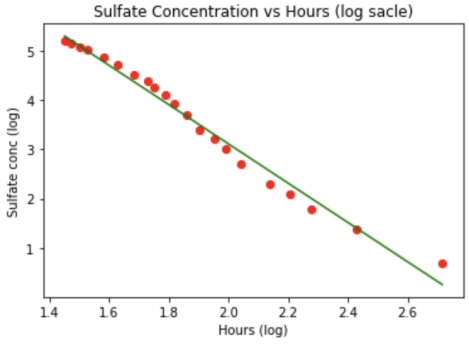
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X.values.reshape(-1, 1), y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

1.1 Part a

Please find below the plot showing the data points and the regression line in log-log coordinates

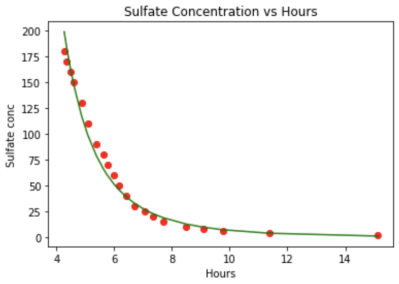
```
plt.scatter(X, y, color = "red")
plt.plot(X, lr.predict(X.values.reshape(-1,1)), color = "green")
plt.title("Sulfate Concentration vs Hours (log sacle)")
plt.xlabel("Hours (log)")
plt.ylabel("Sulfate conc (log)")
plt.show()
```



1.2 Part b

Please find below the plot showing the data points and the regression line in original coordinates

```
plt.scatter(brunhild['Sulfate'], brunhild['Hours'], color = "red")
plt.plot(brunhild['Sulfate'], np.exp(lr.predict(X.values.reshape(-1,1))), color = "green")
plt.title("Sulfate Concentration vs Hours")
plt.xlabel("Hours")
plt.ylabel("Sulfate conc")
plt.show()
```

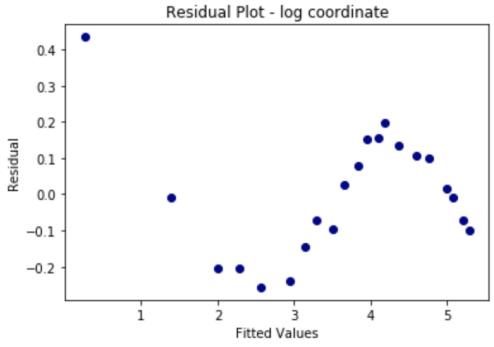


1.3 Part c

Please find below the plot showing the residual against the fitted values in log-log coordinates

```
y_predicted = lr.predict(X.values.reshape(-1,1))
residuals = y-y_predicted
plt.plot(y_predicted,residuals, 'o', color='darkblue')
plt.title("Residual Plot - log coordinate")
plt.xlabel("Fitted Values")
plt.ylabel("Residual")
```

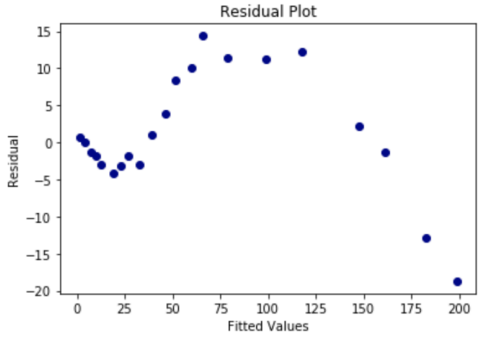
```
Text(0, 0.5, 'Residual')
```



Please find below the plot showing the residual against the fitted values in original coordinates

```
]: y_predicted = lr.predict(X.values.reshape(-1,1))
   residuals = np.exp(y)-np.exp(y_predicted)
   plt.plot(np.exp(y_predicted),residuals, 'o', color='darkblue')
   plt.title("Residual Plot")
   plt.xlabel("Fitted Values")
   plt.ylabel("Residual")
```

```
]: Text(0, 0.5, 'Residual')
```



1.4 Part d

From the plot between the data points and the regressions line (Figure 1 and 2), we can observe that the predicted values are very close to the actual value. However, if we observe the residual plots (Figure 3 and 4), we understand that the model over-predicts for the first half of the data points and under-predicts for the second half of the data points. Its not very obvious from the first two figures as the residual values are low. This kind of residual plot is due to the non-linear nature of the datapoints.

2 Problem 2

We built a linear regression of predicting the body mass from the diameters.

```
: mass = physical["Mass"]
  features = physical.drop(["Mass"], axis=1)
```

```
: mlr = LinearRegression()
  mlr.fit(features, mass)
```

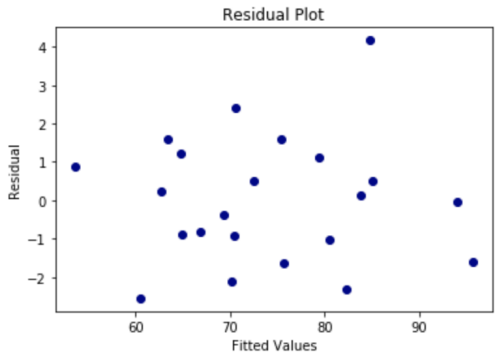
```
: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

2.1 Part a

Plot of residual against the fitted values for the regression.

```
mass_predicted = mlr.predict(features)
residual = mass - mass_predicted
plt.plot(mass_predicted, residual, 'o', color='darkblue')
plt.title('Residual Plot')
plt.xlabel('Fitted Values')
plt.ylabel('Residual')
```

```
Text(0, 0.5, 'Residual')
```



2.2 Part b

We regressed the cube root of mass against the diameters

```
cube_mass = physical
cube_mass['Mass']=np.power((cube_mass['Mass']),1/3)
```

Plot of residual against the fitted values for the regression.

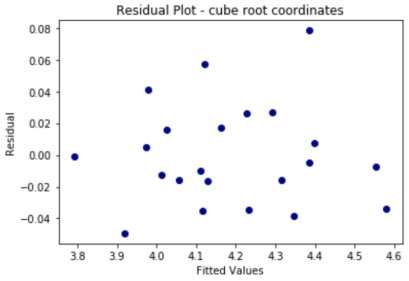
```
mlr.fit(features, cube_mass['Mass'])
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
mass_predicted = mlr.predict(features)
print(len(mass_predicted), len(cube_mass))
residual = cube_mass["Mass"] - mass_predicted
plt.plot(mass_predicted, residual, 'o', color='darkblue')
plt.title('Residual Plot - cube root coordinates')
plt.xlabel('Fitted Values')
plt.ylabel('Residual')
```

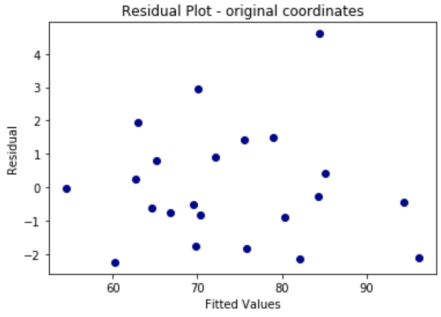
```
22 22
```

```
Text(0, 0.5, 'Residual')
```




```
mass_predicted = mlr.predict(features)
residual = np.power(cube_mass["Mass"],3) - np.power(mass_predicted, 3)
plt.plot(np.power(mass_predicted,3), residual, 'o', color='darkblue')
plt.title('Residual Plot - original coordinates')
plt.xlabel('Fitted Values')
plt.ylabel('Residual')
```

```
Text(0, 0.5, 'Residual')
```



2.3 Part c

The residual plots look very identical for part a and b. The residuals are randomly distrubted around y=0 axis. Therefore, the linear regression is really good.