CMSC178DA

Activity 4

Applying Multiple Linear Regression Analysis to Multivariate Data

Submitted by:

Peladas, Daenielle Rai

Section B (TF 10:30 AM - 12:00 PM)

The Case

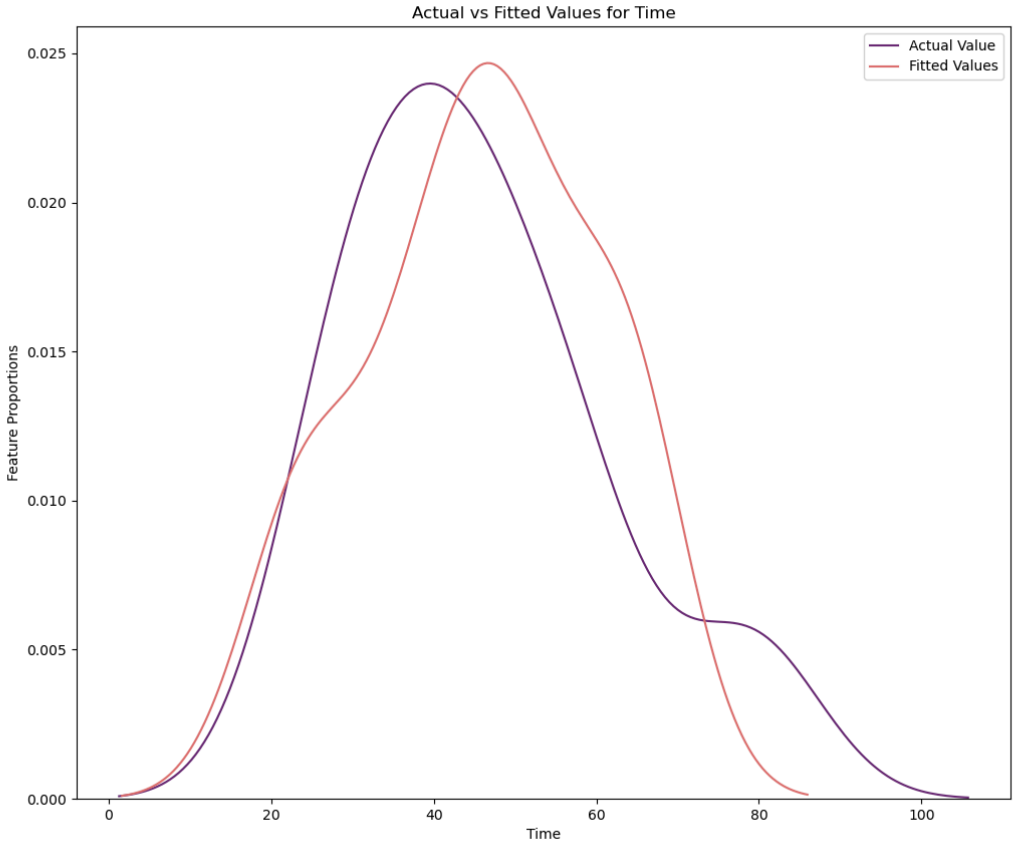
Let us reuse the performance data on remote procedure call (RPC) mechanism for two (2) mainframe operating systems (OS) - UNIX and ARGUS – in Activity 3. The performance metric was total elapsed time (in milliseconds), which was measured for various data sizes. Fit a multiple linear regression model that combine the sample measurements on total elapsed time in processing with various data sizes for the two OSes.

Unix		Argus	
Data Bytes	Time	Data Bytes	Time
64	26.4	92	32.8
64	26.4	92	34.2
64	26.4	92	32.4
64	26.2	92	34.4
234	33.8	348	41.4
590	41.6	604	51.2
846	50.0	860	76.0
1060	48.4	1074	80.8
1082	49.0	1074	79.8
1088	42.0	1088	58.6
1088	41.8	1088	57.6
1088	41.8	1088	59.8
1088	42.0	1088	57.4

The data analyst poses this question to himself: **Can we reasonably predict the total elapsed time of RPCs based on the various processed data sizes and type of OS (i.e., UNIX and ARGUS)?**

The Multiple Linear Regression Output

Fitted Model Summary Output



OLS Regression Results

Dep. Variable:	Time	R-squared:	0.765
Model:	OLS	Adj. R-squared:	0.744
Method:	Least Squares	F-statistic:	37.36
Date:	Fri, 26 Apr 2024	Prob (F-statistic):	5.95e-08
Time:	22:04:46	Log-Likelihood:	-89.579
No. Observations:	26	AIC:	185.2
Df Residuals:	23	BIC:	188.9
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	36.7390	3.251	11.302	0.000	30.015	43.463
Data Bytes	0.0252	0.004	7.137	0.000	0.018	0.033
OS	-14.9266	3.165	-4.717	0.000	-21.473	-8.380

Omnibus:	6.001	Durbin-Watson:	0.636
Prob(Omnibus):	0.050	Jarque-Bera (JB):	4.814
Skew:	1.052	Prob(JB):	0.0901
Kurtosis:	3.139	Cond. No.	1.98e+03

Narrative Text

Importing the Dataset

Import library

Import the `pandas` library for data manipulation and analysis.

```python
import pandas as pd
```

Load dataset to dataframe

The previously used dataset from Activity 3 was exported as a csv file and was reused for this activity. Define the `path` containing the dataset and use `read_csv` to load the said dataset to a pandas DataFrame labeled `df`.

```python
path = r"C:\Users\daeni\Desktop\LOVE\Academics\CMSC178DA/Activity/Multiple Linear Regression/os_concat_rpc.csv"
df = pd.read_csv(path)
```

One Hot Encoding the Categorical Values for OS Type

Update the column values under `'OS'` found in `df` and assign a numerical representation of `0` for OS types equal to `Argus` (reference point) and `1` for `Unix`.

```python
df['OS'] = df['OS'].map({'Argus': 0, 'Unix': 1})
```

	Data Bytes	Time	OS
0	64	26.4	1
1	64	26.4	1
2	64	26.4	1
3	64	26.2	1
4	234	33.8	1
5	590	41.6	1
6	846	50.0	1
7	1060	48.4	1
8	1082	49.0	1
9	1088	42.0	1
10	1088	41.8	1
11	1088	41.8	1
12	1088	42.0	1
13	92	32.8	0
14	92	34.2	0
15	92	32.4	0
16	92	34.4	0
17	348	41.4	0
18	604	51.2	0
19	860	76.0	0
20	1074	80.8	0
21	1074	79.8	0
22	1088	58.6	0
23	1088	57.6	0
24	1088	59.8	0
25	1088	57.4	0

Model Development

Import visualization libraries

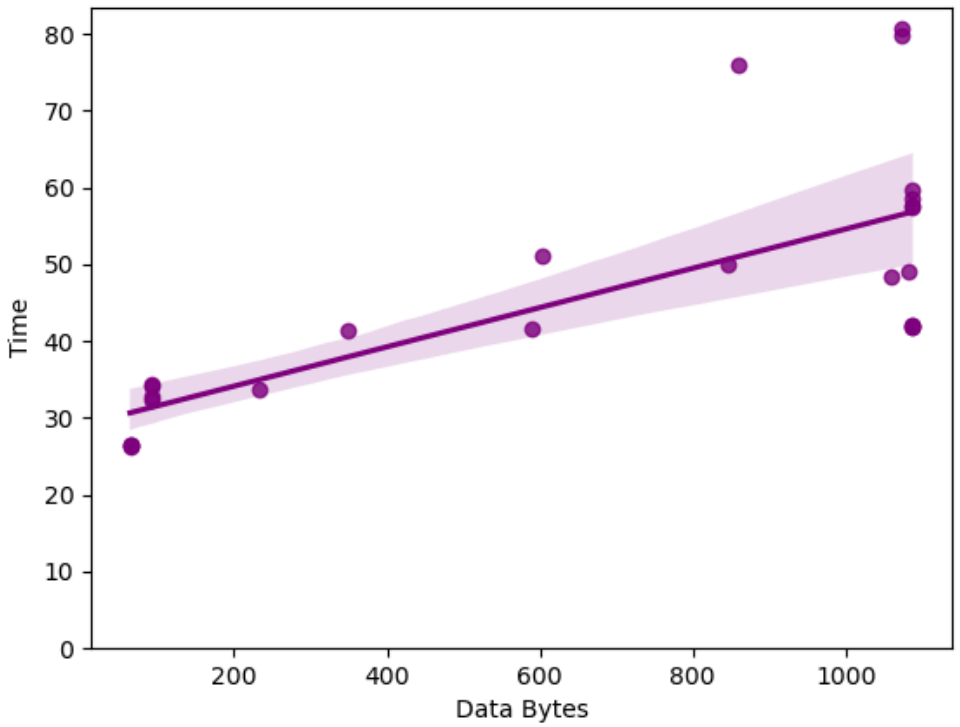
Import the `matplotlib and seaborn` libraries for plotting graphs and other data visualization functions.

```python
import seaborn as sns
import matplotlib.pyplot as plt
```

Visualizing relationships between variables

Generate a regression plot using `seaborn`'s `regplot` function to explore the relationship between data byte size (`x`) and time (`y`) in the data (`df`).

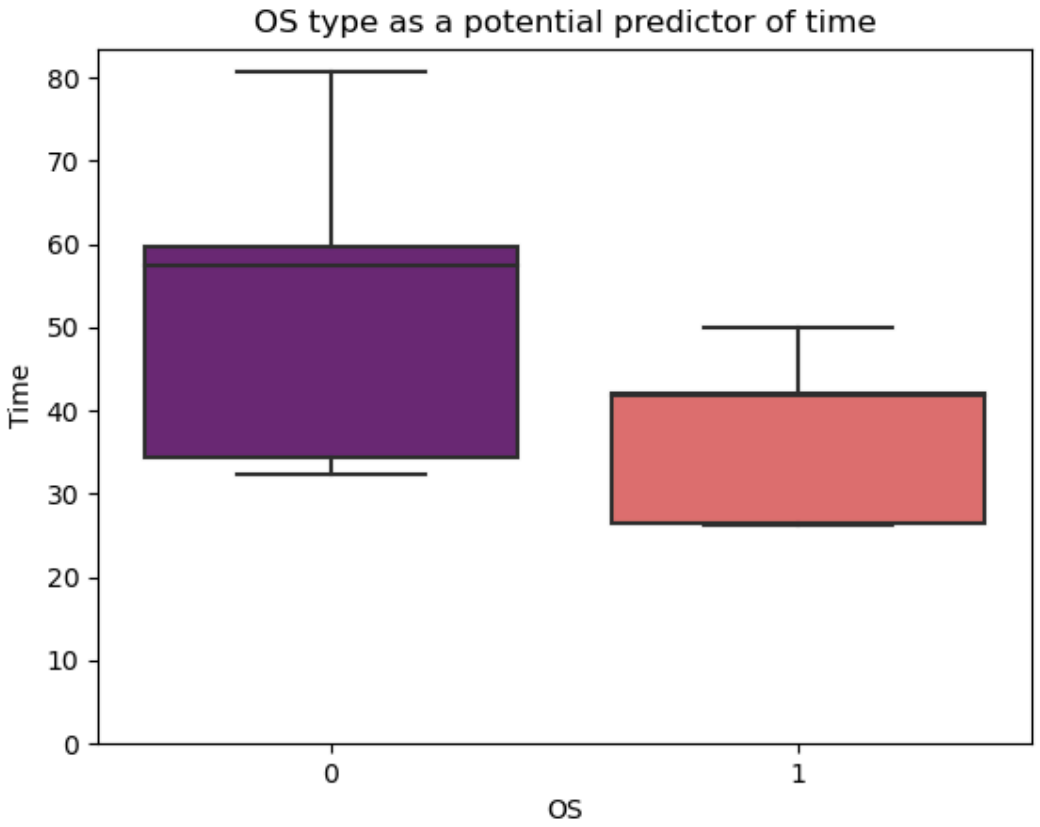
```python
sns.regplot(x="Data Bytes", y="Time", data=df, color='purple')
plt.ylim(0,)
plt.title("Data bytes size as a potential predictor of time")
```



The regression plot reveals a weakly positive linear relationship between data byte size and time elapsed. The shallow slope suggests data size may not be a strong predictor of time in this case.

Here, a `seaborn boxplot` is employed to visually explore the distribution of `Time` elapsed data for the two `OS` types.

```python
sns.boxplot(x="OS", y="Time", data=df, palette='magma')
plt.ylim(0,)
plt.title("OS type as a potential predictor of time")
```



The box plot reveals a wider distribution of time elapsed values for Argus OS (represented by 0). The box extends further upwards, indicating that Argus OS experiences a greater range in completion times compared to Unix. Conversely, the tighter box for Unix suggests a more consistent performance, with data points clustered closer together.

Import machine learning libraries

Import the `LinearRegression` tool from `scikit-learn` to create linear regression models. These models find straight line relationships between data points.

```python
from sklearn.linear_model import LinearRegression
```

Build the model

Instantiate the `LinearRegression()` model into a variable named `lm`.

```python
lm = LinearRegression()
```

Define the predictor and to-be predicted variables. For this case, the "`Data Bytes`" size and "`OS`" type are used as `features` and the `Time` elapsed as the `target` variable.

```python
features = df[["Data Bytes", "OS"]]
target = df["Time"]
```

The `fit` function trains a model by uncovering the relationship between the `features` and the `target` variable. It uses the `features` to learn how they influence the `target` variable. This allows the model to make predictions for new data based on the patterns it discovers in the training data.

```python
lm.fit(features, target)
```

The trained model offers the intercept (`b0`) from its `intercept_` attribute and coefficients `(b1, b2)` from `coef_`, corresponding to the regression coefficients from data byte size and OS type, respectively.

```python
b0 = lm.intercept_
b1, b2 = lm.coef_
print(f"Values\nb0: {b0}\nb1: {b1}\nb2: {b2}")
```

```
Values
b0: 36.739006054661886
b1: 0.025206557752234512
b2: -14.926638075724542
```

Model Evaluation

Via Visualization

The `predict` function performs predictions using a previously trained linear regression model `lm`. Using the knowledge lm gained from analyzing past data to estimate what the target variable would be for completely new data points. The resulting predictions are stored in `y_hat`.

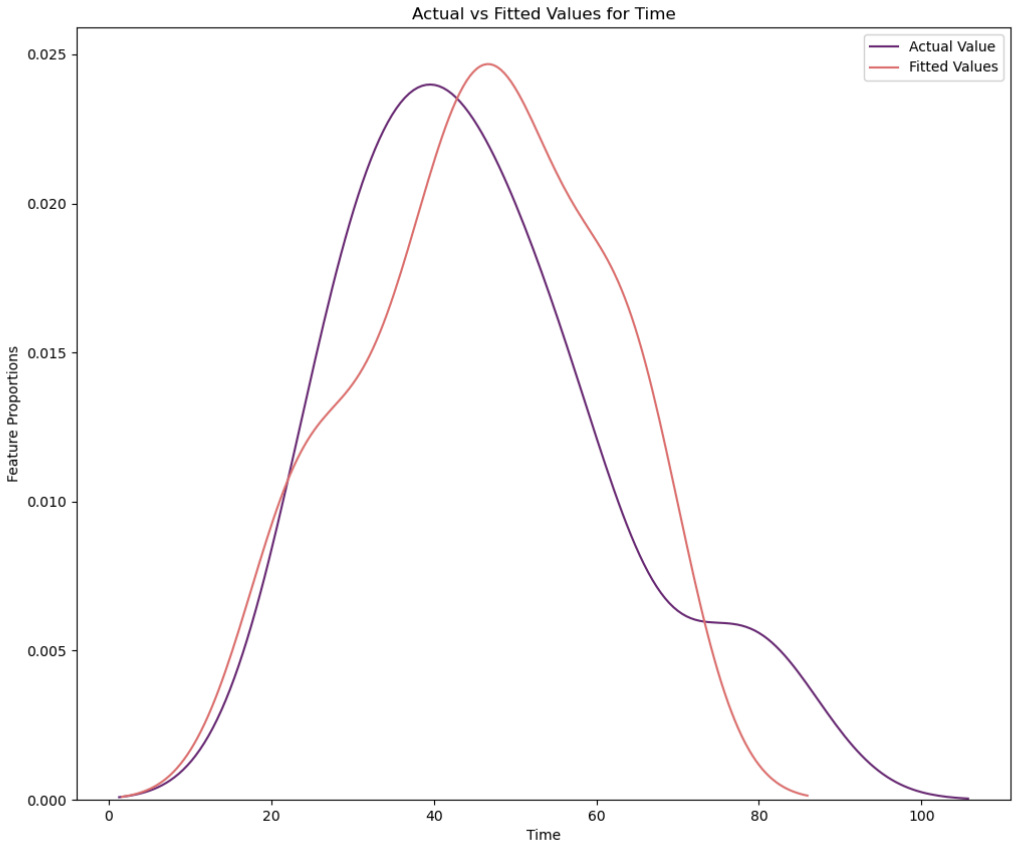
```python
y_hat = lm.predict(features)
```

Utilizing the `sns.distplot` twice to create a combined distribution plot comparing actual and predicted values for the `Time` variable. It avoids histograms by setting `hist=False` and generates plots for both the actual values (labeled `Actual Value`) from the `Time` column in `df` and the predicted values (labeled `Fitted Values`) previously stored in `y_hat`. Both plots are overlaid on the same axis using `ax=ax1`.

```python
ax1 = sns.distplot(df['Time'], hist=False, color="#692c75", label="Actual
Value")
sns.distplot(y_hat, hist=False, color="#de7270", label="Fitted Values" ,
ax=ax1)

ax1.legend()
plt.title('Actual vs Fitted Values for Time')
plt.xlabel('Time')
plt.ylabel('Feature Proportions')

plt.show()
```



While the model visually appears to capture the trend for RPC time elapsed based on data size and OS type, there are limitations to its accuracy. The fitted values deviate from the actual values, particularly for times exceeding 80, but also around the middle, starting from a time elapsed of 20. This mismatch suggests the model's inability to perfectly predict all scenarios, even for moderate values. The limited size of the dataset might be a contributing factor. With more data, the model could potentially learn the underlying patterns more effectively and reduce these deviations. To gain a deeper understanding, further exploration of the data, both visually and statistically, is described further below.

Via Statistical Evaluation

Using the `score` function from the regression model `lm`, the `features` and `target` value described previously are inputted to get the R-squared value of the model. This measure can tell how well a regression model fits a set of data, specifically focusing on the proportion of variance in the dependent variable that the model can explain.

```python
lm.score(features, target)
```

```
0.764643674084501
```

While regular R-squared tells a data scientist how well a model fits the data, it can increase simply by adding more features, even if those features aren't truly explanatory. This can lead to overfitting, where the model performs well on the training data but poorly on unseen data.

Adjusted R-squared penalizes the model for using more features, giving a better sense of how well the model would perform on unseen data.

Mathematically, adjusted R-squared is derived from the regular R-squared using the following formula:

$$Adjusted\ R^2 = 1 - (1 - R^2) * ((n - 1) / (n - p - 1))$$

Where:
n is the number of observations in the data.
p is the number of features used in the model.

```python
1 - (1-lm.score(features,
target))*(len(target)-1)/(len(target)-features.shape[1]-1)
```

```
0.764643674084501
```

To get the p-values, a different library to build the same linear regression model is utilized. Here, `statsmodels.api` functionalities are imported from the `statsmodels` library for statistical modeling. It is used to build another linear regression model which operates in the same manner. This is useful for comparison but also for getting values that would be hard to derive from the `scikit-learn` library solely.

```python
import statsmodels.api as sm
```

Prepare the data and extract the p-values for a linear regression model built with `statsmodels`. It first adds a constant term (intercept) `X` using `sm.add_constant` since `statsmodels` doesn't include it by default. Then, it builds the model using `sm.OLS` with the `target` variable and the data with the intercept.

```python
X = sm.add_constant(features)
model = sm.OLS(target, X).fit()
```

Finally, the code extracts a summary of the model results using `model.summary()`, which likely includes p-values in the `P>|t|` column. These p-values, obtained from the summary table, indicate the significance of each feature in explaining the target variable.

```python
summary = model.summary()
print(summary)
```

OLS Regression Results

Dep. Variable:	Time	R-squared:	0.765
Model:	OLS	Adj. R-squared:	0.744
Method:	Least Squares	F-statistic:	37.36
Date:	Fri, 26 Apr 2024	Prob (F-statistic):	5.95e-08
Time:	22:04:46	Log-Likelihood:	-89.579
No. Observations:	26	AIC:	185.2
Df Residuals:	23	BIC:	188.9
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	36.7390	3.251	11.302	0.000	30.015	43.463
Data Bytes	0.0252	0.004	7.137	0.000	0.018	0.033
OS	-14.9266	3.165	-4.717	0.000	-21.473	-8.380

Omnibus:	6.001	Durbin-Watson:	0.636
Prob(Omnibus):	0.050	Jarque-Bera (JB):	4.814
Skew:	1.052	Prob(JB):	0.0901
Kurtosis:	3.139	Cond. No.	1.98e+03

This model produces similar outputs to the previous model, suggesting consistency across libraries and potentially validating its accuracy.

Interpretation

Is the fitted multiple regression model statistically valid, or meaningful? Justify.

Employing a hypothesis testing to assess the significance of coefficients for data size and OS type in predicting time elapsed, the null hypothesis (H_0) posits that these coefficients are zero, implying that the features data bytes size and OS type have no influence on the predicted time. On the other hand, the alternative hypothesis (H_1) proposes that the coefficients are not zero, indicating some level of impact from the features in predicting the target value.

A significance level (α) of 0.05 is utilized in this case as it is a common parameter in testing a hypothesis. The model summary revealed a p-value of 0 in the 'P>|t|' column for both data size and OS type. This exceptionally low p-value signifies a strong rejection of the null hypothesis.

Therefore, we can confidently conclude that both data byte size and OS type have a statistically significant effect on predicted time elapsed. However, it's important to acknowledge that obtaining such an extreme p-value (0) is uncommon. This could be partially attributed to the size of the dataset. A limited dataset might not encompass a diverse range of data points, potentially hindering the model's training effectiveness.

How much variation of the total elapsed time can be explained by the multiple regression model? Does this answer the question posed by the data analyst? Justify.

The multiple linear regression model explains 76.46% of the variance in time elapsed given that it is its R-squared value. This indicates a good fit between the model's predictions and the actual data. It is also important to consider the adjusted R-squared value which is 74.42%. This statistic penalizes the model for including more features in the training and provides a better estimate of how well the model will perform on unseen data and increasing features. In this case, there were only two features — data byte size and OS type — which would explain the relatively low difference between the R-squared and adjusted R-squared values.

The high adjusted R-squared, alongside the high R-squared, strengthens the model's validity and suggests it effectively captures the relationship between the feature variables — OS type and data bytes size — and target variable — time elapsed.

Which model parameter(s) contributed meaningfully in predicting the total elapsed time for RPCs? Identify. Justify.

The analysis suggests that OS type is a more significant predictor of total elapsed time for RPCs compared to data byte size. This conclusion is based on the coefficient values: absolute value of 14.93 for OS type and 0.03 for data byte size. The larger magnitude of the OS type coefficient indicates a stronger influence on the predicted time.

This finding aligns with our observations of the data. Since data byte sizes for both OS types are nearly identical, it's reasonable to expect minimal impact on the predicted time elapsed from this variable.

What is the per-byte processing cost (i.e., elapsed time) on the UNIX OS? Identify. Justify.

The provided formula to predict elapsed time based on data byte size and OS type was used. Assuming a data byte of 1, the calculation is as follows:

$$y = b_0 + b_1 x_1 + b_2 x_2$$

$$y = 36.739006054661886 + (1 \cdot 0.025206557752234512) + \ldots$$
$$\ldots + (1 \cdot -14.926638075724542$$
$$y = 21.83757453668958$$

This indicates that the Unix OS (represented by OS type = 1) can process one data byte in approximately 21.84 seconds. For the Argus OS (represented by OS type = 0), the calculation is:

$$y = 36.739006054661886 + (1 \cdot 0.025206557752234512) + \ldots$$
$$\ldots + (0 \cdot -14.926638075724542$$
$$y = 36.76421261241412$$

This indicates that the Argus OS (represented by OS type = 0) can process one data byte in approximately 36.76 seconds.

As expected, the negative coefficient (-14.927) for the OS type results in a processing time difference of about 14.93 seconds between Unix and Argus. This aligns with previous observations where Unix consistently outperformed Argus in processing data bytes.

What is the final multiple linear regression model specification which you can now use in predicting the processing elapsed times for RPCs? Specify.

Final Linear Function:

$$y = b_0 + b_1 x_1 + b_2 x_2$$

Where:

- predicted elapsed time (y)
- slope intercept (b_0) = 21.812367978937342
- coefficient for data bytes (b_1) = 0.025206557752234512
- coefficient for OS type (b_2) = -14.926638075724542
- data size (x_1)
- OS type (x_2)