MET CS677 Data Science with Python
Assignment 4

Alan Szmyt

April 12, 2023

In this assignment, we will implement a number of linear models (including linear regression) to model relationships between different clinical features for heart failure in patients.

For the dataset, we use “heart failure clinical records dataset“ at UCI:

https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records.

Dataset Description: From the website: “This dataset contains the medical records of 299 patients who had heart failure, collected during their follow-up period, where each patient profile has 13 clinical features.“

These 13 features are:

1. age: age of the patient (years)
2. anaemia: decrease of red blood cells or hemoglobin (boolean)
3. high blood pressure: if the patient has hypertension (boolean)
4. creatinine phosphokinase (CPK): level of the CPK enzyme in the blood (mcg/L)
5. diabetes: if the patient has diabetes (boolean)
6. ejection fraction: percentage of blood leaving the heart at each contraction (percentage)
7. platelets: platelets in the blood (kiloplatelets/mL)
8. sex: woman or man (binary)
9. serum creatinine: level of serum creatinine in the blood (mg/dL)
10. serum sodium: level of serum sodium in the blood (mEq/L)
11. smoking: if the patient smokes or not (boolean)
12. time: follow-up periods (days)

target death event: if the patient deceased (DEATH_EVENT = 1) during the follow-up period (boolean)

We will focus on the following subset of four features:

1. creatinine phosphokinase
2. serum creatinine
3. serum sodium
4. platelets

and try to establish a relationship between some of them using various linear models and their variants.

1 Question 1

Cell 03

```
from collections import defaultdict
from pathlib import Path

import numpy as np
import pandas as pd
import seaborn as sns
from IPython.display import Latex
from pandas import DataFrame

from assignment4 import DeathEvent, LinearModelAnalytics, LinearModelMetrics
from constants import (
    COL_CPK,
    COL_DEATH_EVENT,
    COL_SERUM_SODIUM,
    INITIAL_COLS,
    COL_PLATELETS,
    COL_SERUM_CREATININE,
)
from utils import (
    artifacts,
    data,
    examine_correlation_matrix,
    plot_correlation_matrix, create_latex_table
)

# Global Seaborn options.
sns.set_theme(font_scale=1.5, rc={"text.usetex": True})

cwd: Path = Path.cwd()
```

Cell 04

```
# Heart failure clinical records dataset file from UCI.
dataset_csv: str = "heart_failure_clinical_records_dataset.csv"
heart_failure_dataset_file: Path = data.joinpath(dataset_csv)
```

Cell 05

```
# dtype mapping to use for the csv file. Set death_event to categorical.
dtypes: dict = defaultdict(
    np.float64,
    {
        COL_DEATH_EVENT: pd.CategoricalDtype.name,
        COL_CPK: np.int64,
        COL_SERUM_SODIUM: np.int64,
    },
)

# Load the heart failure records into a dataframe.
heart_failure_dataset: DataFrame = pd.read_csv(
    heart_failure_dataset_file, usecols=INITIAL_COLS, dtype=dtypes
)
```

1.1 Load the data into a Pandas dataframe. Extract two dataframes with the above 4 features: df_0 for surviving patients (DEATH_EVENT = 0) and df_1 for deceased patients (DEATH_EVENT = 1)

Answer:

Cell 06

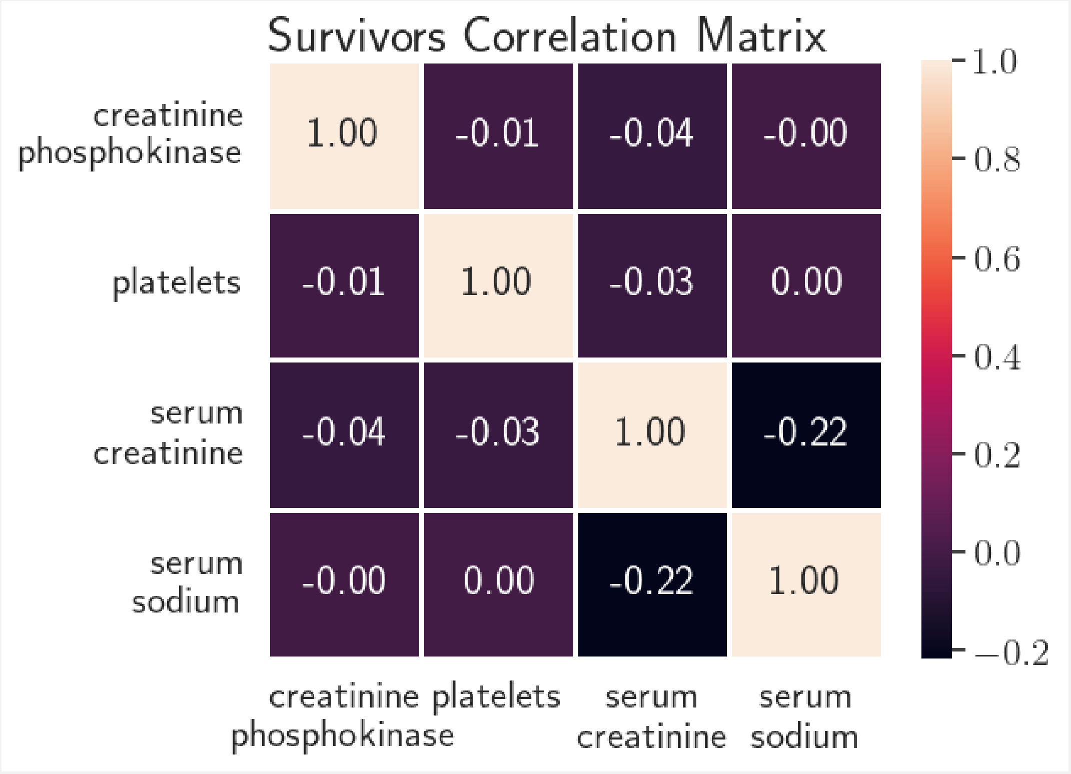
```
# Load the survivors into their own dataframe.
survivors: DataFrame = heart_failure_dataset.loc[
    heart_failure_dataset[COL_DEATH_EVENT].astype(int) == DeathEvent.SURVIVOR
]

# Load the deceased into their own dataframe.
deceased: DataFrame = heart_failure_dataset.loc[
    heart_failure_dataset[COL_DEATH_EVENT].astype(int) == DeathEvent.DECEASED
]
```

1.2 For each dataset, construct the visual representations of the corresponding correlation matrices M_0 (from df_0) and M_1 (from df_1) and save the plots into two separate files

Answer:

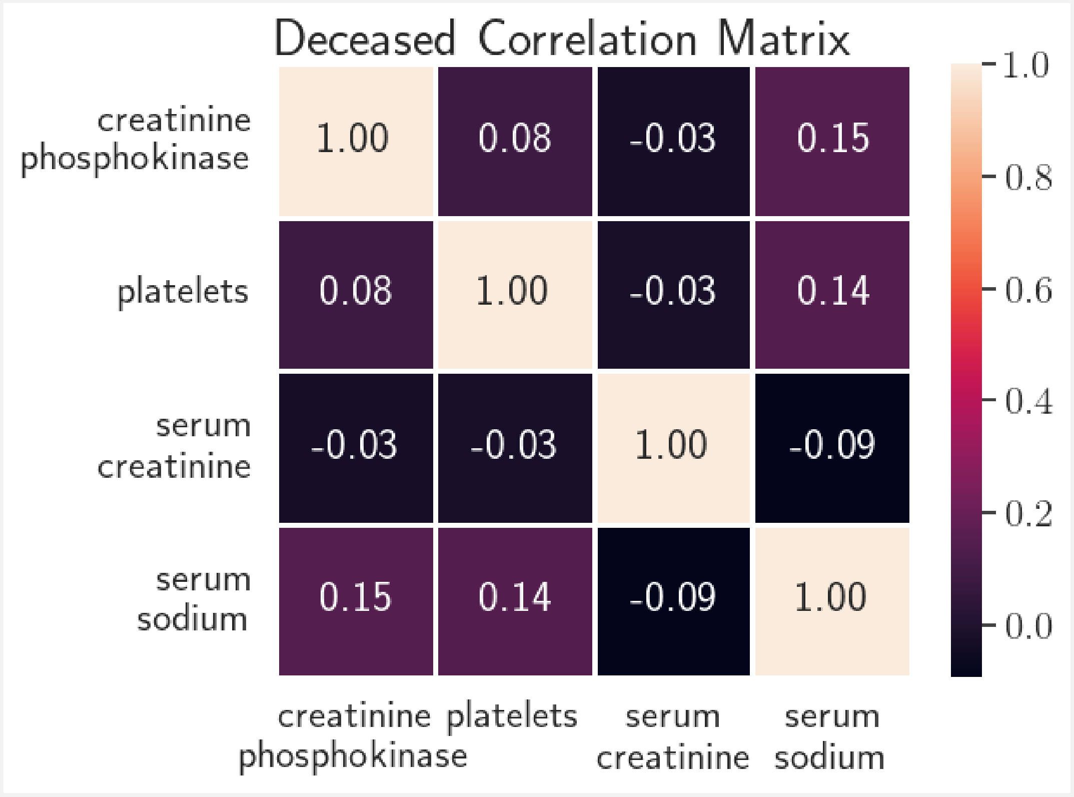
Cell 07

```
# Plot a correlation matrix for the 'survivors' dataset.
survivors_correlations: DataFrame = plot_correlation_matrix(
    survivors,
    title="Survivors Correlation Matrix",
    output_dir=artifacts
)
```



Cell 08

```python
# Plot a correlation matrix for the 'deceased' dataset.
deceased_correlations: DataFrame = plot_correlation_matrix(
    deceased,
    title="Deceased Correlation Matrix",
    output_dir=artifacts
)
```



1.3 Examine your correlation matrix plots visually and answer the following

Answer:

Cell 09

```python
# Examine the survivor patients' correlation matrix.
s_correlations = examine_correlation_matrix(survivors_correlations)

# Examine the deceased patients' correlation matrix.
d_correlations = examine_correlation_matrix(deceased_correlations)
```

(a) Which features have the highest correlation for surviving patients?

Answer:

CELL 10

```python
# Survivors highest correlation features.
shc_features = list(s_correlations.head(1).to_dict().items())[0]
print(
    f"The features with the highest correlation for surviving patients"
    f" were '{shc_features[0][0]}' and '{shc_features[0][1]}' with"
    f" a correlation value of {shc_features[1]:.3f}."
)
```

```
The features with the highest correlation for surviving patients were 'serum_creatinine' a
```

(b) Which features have the lowest correlation for surviving patients?

Answer:

CELL 11

```python
# Question 1.3.b
# Survivors lowest correlation features.
slc_features = list(s_correlations.tail(1).to_dict().items())[0]
print(
    f"The features with the lowest correlation for surviving patients"
    f" were '{slc_features[0][0]}' and '{slc_features[0][1]}' with"
    f" a correlation value of {slc_features[1]:.3f}."
)
```

```
The features with the lowest correlation for surviving patients were 'platelets' and 'seru
```

(c) Which features have the highest correlation for deceased patients?

Answer:

CELL 12

```python
# Question 1.3.c
# Deceased patients' highest correlation features.
dhc_features = list(d_correlations.head(1).to_dict().items())[0]
print(
    f"The features with the highest correlation for deceased patients"
    f" were '{dhc_features[0][0]}' and '{dhc_features[0][1]}' with"
    f" a correlation value of {dhc_features[1]:.3f}."
)
```

```
The features with the highest correlation for deceased patients were 'creatinine_phosphoki
```

(d) Which features have the lowest correlation for deceased patients?

Answer:

CELL 13

```
# Question 1.3.d
# Deceased patients' lowest correlation features.
dlc_features = list(d_correlations.tail(1).to_dict().items())[0]
print(
    f"The features with the lowest correlation for deceased patients"
    f" were '{dlc_features[0][0]}' and '{dlc_features[0][1]}' with"
    f" a correlation value of {dlc_features[1]:.3f}."
)
```

```
The features with the lowest correlation for deceased patients were 'platelets' and 'serum
```

(e) Are results the same for both cases?

Answer:

The results are not the same for both cases.

2 Question 2: In this question you will compare a number of different models using linear systems (including linear regression).

You choose one feature X as independent variable X and another feature Y as dependent. Your choice of X and Y will depend on your facilitator group as follows:

1. Group 1: X: creatinine phosphokinase (CPK), Y: platelets
2. Group 2: X: platelets, Y: serum sodium
3. Group 3: X: serum sodium, Y: serum creatinine
4. Group 4: X: platelets, Y: serum creatinine

We will now look for the best model (from the list below) that best explains the relationship for surviving and deceased patients. Consider surviving patients (DEATH_EVENT = 0). Extract the corresponding columns for X and Y. For each of the models, below we will take a $50/50$ split, fit model with X_{train} and predict Y_{test} using X_{test}. From the predicted values $Pred(y_i)$ we compute the residuals $r_i = y_i - Pred(y_i)$. We can then estimate the loss function (SSE sum of the squared residuals).

$$L = \sum_{x_i \epsilon X_{test}} {e_i}^2$$

You do the same analysis for deceased patients. You will consider the following models for both deceased and surviving patients:

1. $y = ax + b$ (simple linear regression)
2. $y = ax^2 + bx + c$ (quadratic)
3. $y = ax^3 + bx^2 + cx + d$ (cubic spline)
4. $y = a \log x + b$ (GLM - generalized linear model)

5. $\log y = a \log x + b$ (GLM - generalized linear model)

For each of the models below, you will do the following (for both deceased and surviving patients)

(a) Fit the model on X_{train}.

(b) Print the weights $(a, b, \dots)$.

(c) Compute predicted values using X_{test}.

(d) Plot (if possible) predicted and actual values in X_{test}.

(e) Compute (and print) the corresponding loss function.

Answer:

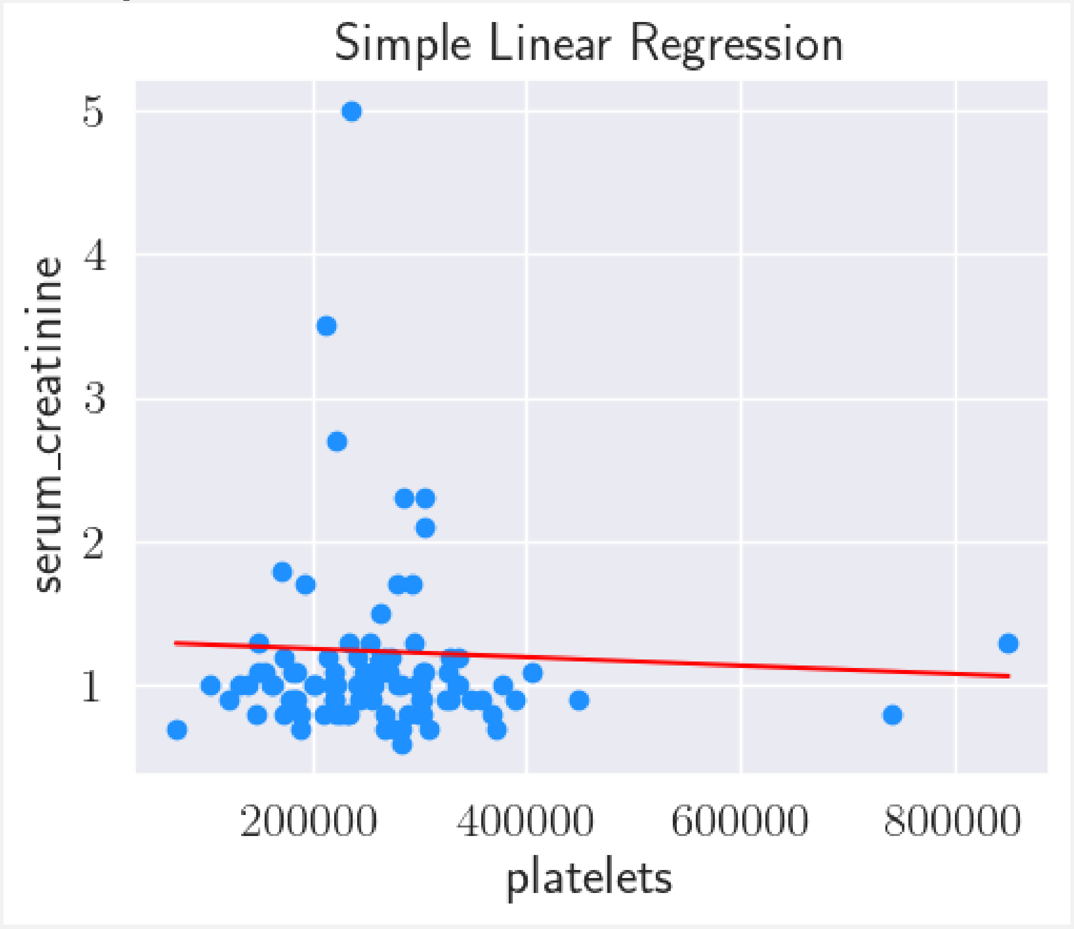
Cell 14

```
# Linear model analytics for different linear regression models for survivors.
survivors_linear_model_analytics: LinearModelAnalytics = LinearModelAnalytics(
    survivors,
    predictor_col=COL_PLATELETS,
    response_col=COL_SERUM_CREATININE,
)

# Linear model analytics for different linear regression models for deceased.
deceased_linear_model_analytics: LinearModelAnalytics = LinearModelAnalytics(
    deceased,
    predictor_col=COL_PLATELETS,
    response_col=COL_SERUM_CREATININE,
)
```

Cell 15

```
# Metrics for simple linear regression model for survivors.
s_slr_metrics: LinearModelMetrics = survivors_linear_model_analytics.simple
print(f"Simple linear coefficients: {s_slr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {s_slr_metrics.sse}")
s_slr_metrics.plot("Simple Linear Regression")
```

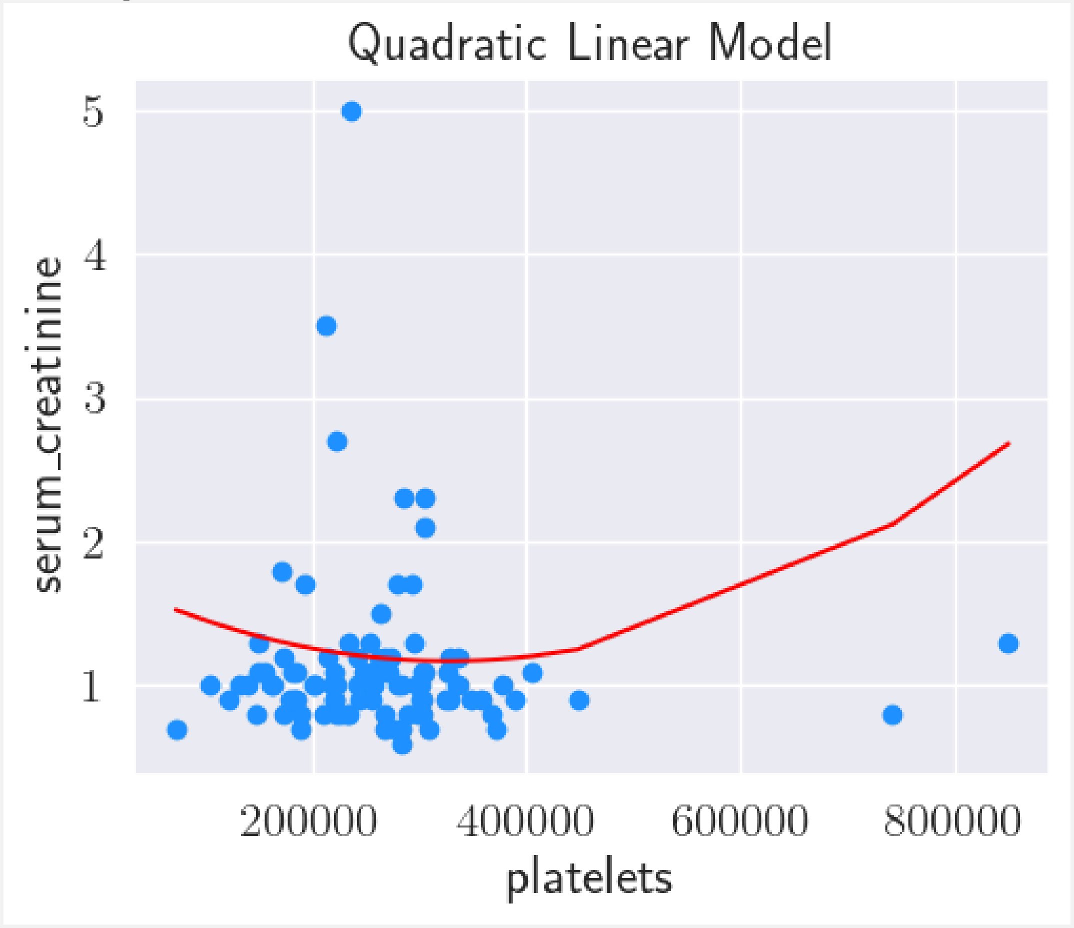
```
Simple linear coefficients: [-2.93473058e-07]
Sum of Squared Errors (SSE): 33.850873557207514
```



Cell 16

```python
# Metrics for quadratic linear regression model for survivors.
s_qlr_metrics: LinearModelMetrics = survivors_linear_model_analytics.quadratic
print(f"Quadratic coefficients: {s_qlr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {s_qlr_metrics.sse}")
s_qlr_metrics.plot("Quadratic Linear Model")
```

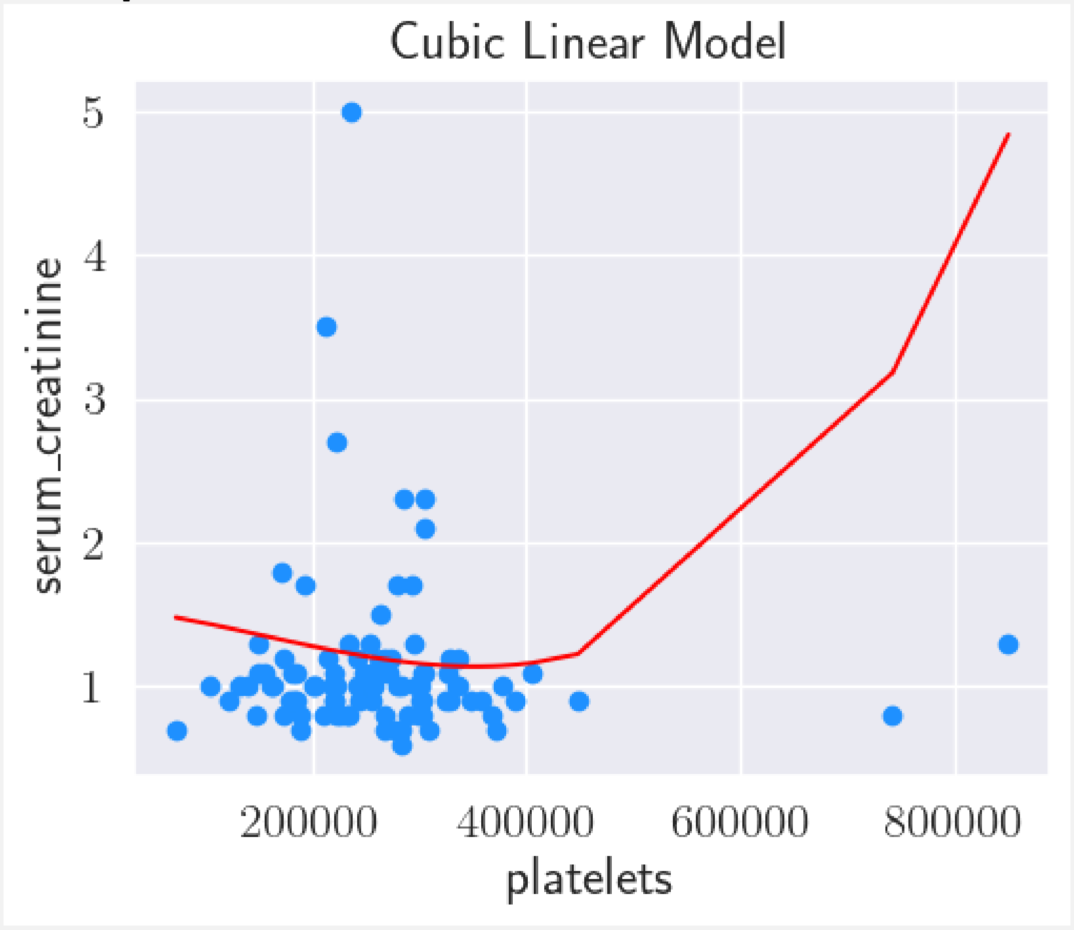
```
Quadratic coefficients: [-3.60956544e-06  5.51837146e-12]
Sum of Squared Errors (SSE): 38.036189854381476
```



Cell 17

```python
# Metrics for cubic linear regression model for survivors.
s_clr_metrics: LinearModelMetrics = survivors_linear_model_analytics.cubic
print(f"Cubic coefficients: {s_clr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {s_clr_metrics.sse}")
s_clr_metrics.plot("Cubic Linear Model")
```

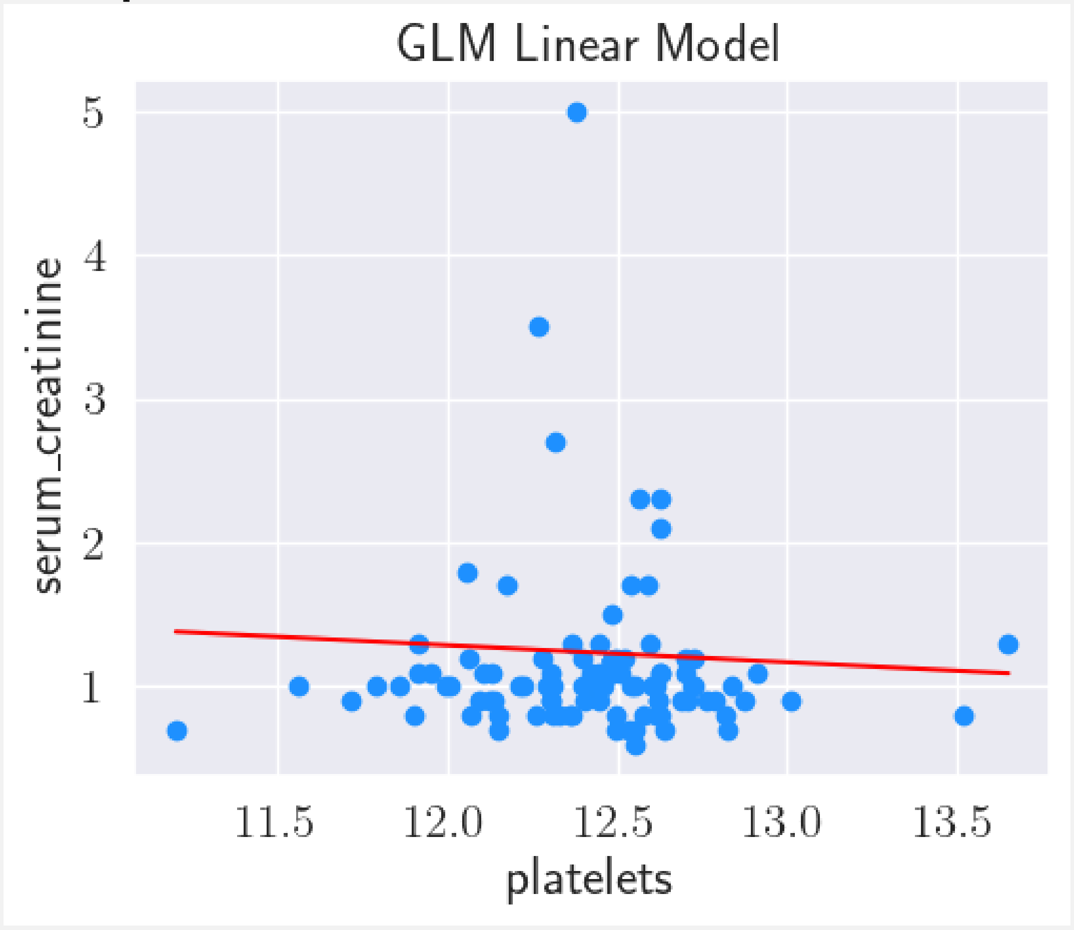
```
Cubic coefficients: [-6.61695812e-07 -6.29160600e-12  1.36579315e-17]
Sum of Squared Errors (SSE): 52.42579302568393
```



Cell 18

```
# Metrics for GLM #1 linear regression model for survivors.
s_glr_metrics1: LinearModelMetrics = survivors_linear_model_analytics.glm1
print(f"GLM coefficients: {s_glr_metrics1.coefficients}")
print(f"Sum of Squared Errors (SSE): {s_glr_metrics1.sse}")
s_glr_metrics1.plot("GLM Linear Model")
```

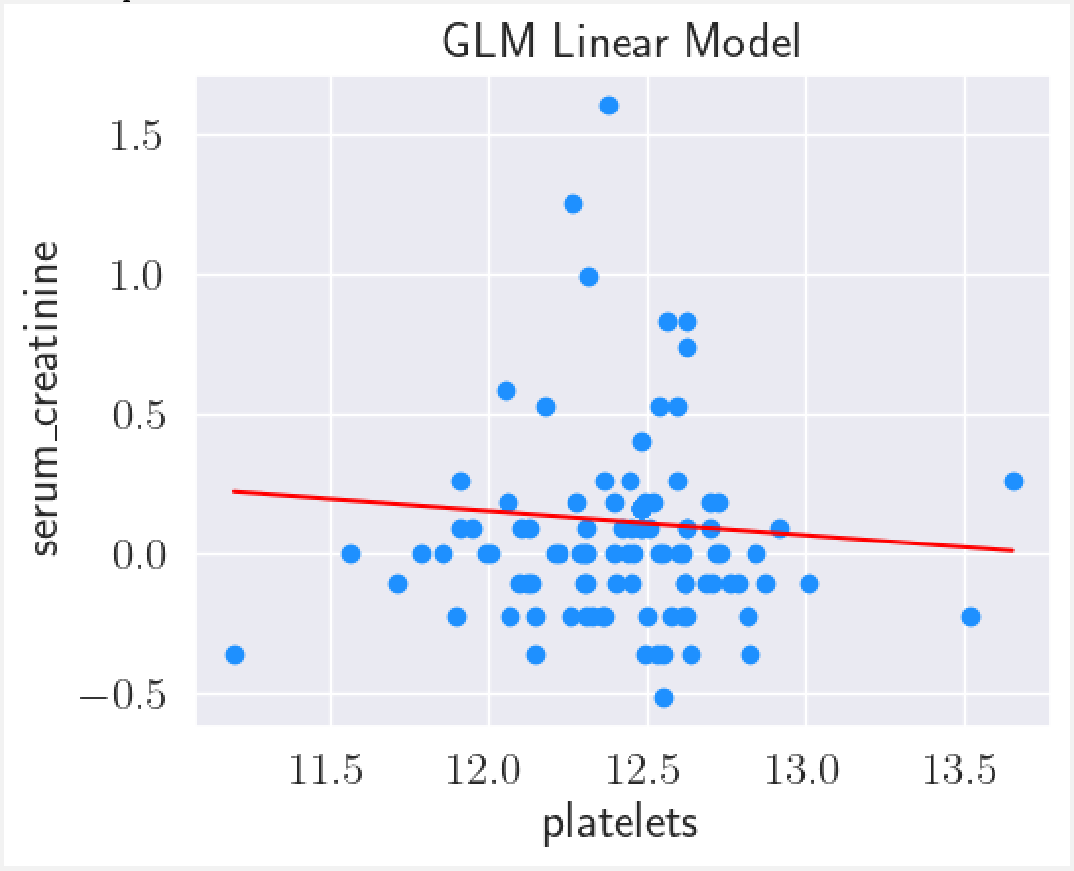
```
GLM coefficients: [-0.11801909]
Sum of Squared Errors (SSE): 34.02122454853615
```



Cell 19

```
# Metrics for GLM #2 linear regression model for survivors.
s_glr_metrics2: LinearModelMetrics = survivors_linear_model_analytics.glm2
print(f"GLM coefficients: {s_glr_metrics2.coefficients}")
print(f"Sum of Squared Errors (SSE): {s_glr_metrics2.sse}")
s_glr_metrics2.plot("GLM Linear Model")
```

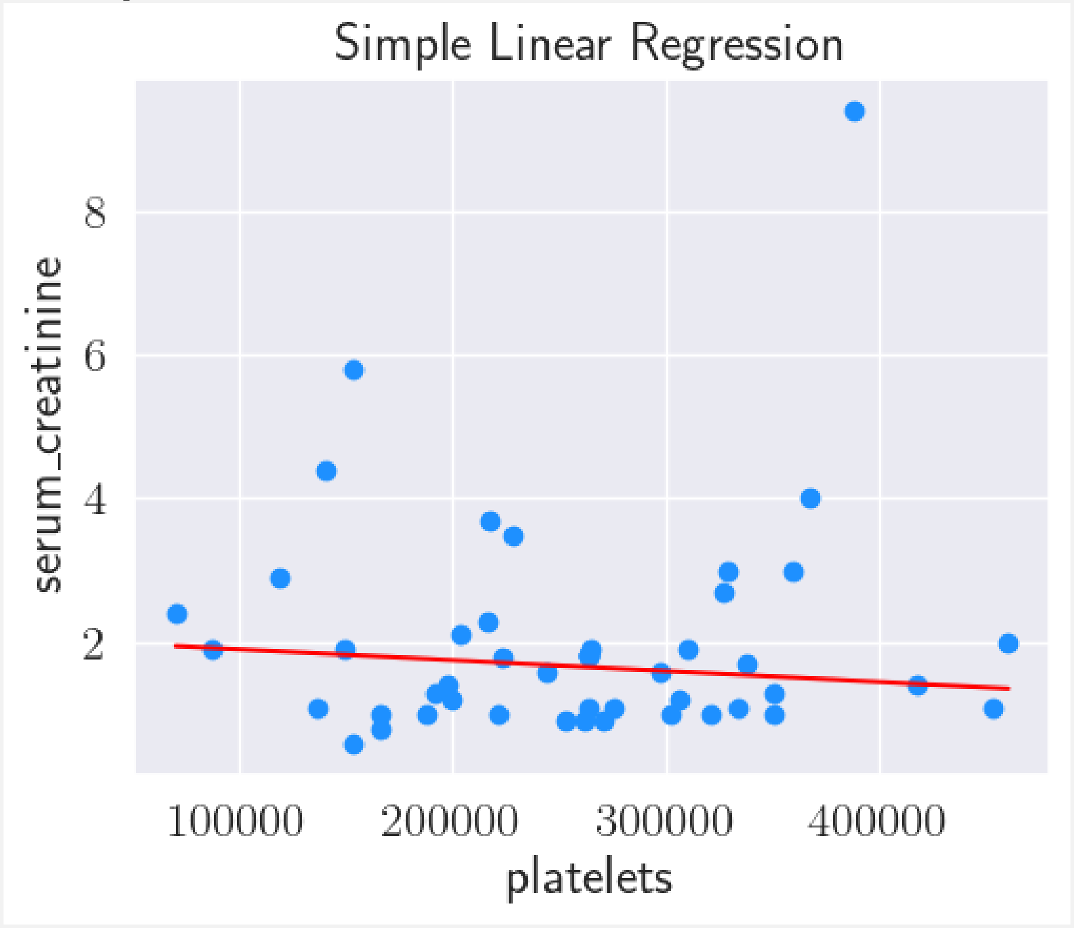
```
GLM coefficients: [-0.08590434]
Sum of Squared Errors (SSE): 11.441515033367178
```



Cell 20

```python
# Metrics for simple linear regression model for deceased.
d_slr_metrics: LinearModelMetrics = deceased_linear_model_analytics.simple
print(f"Simple linear coefficients: {d_slr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {d_slr_metrics.sse}")
d_slr_metrics.plot("Simple Linear Regression")
```

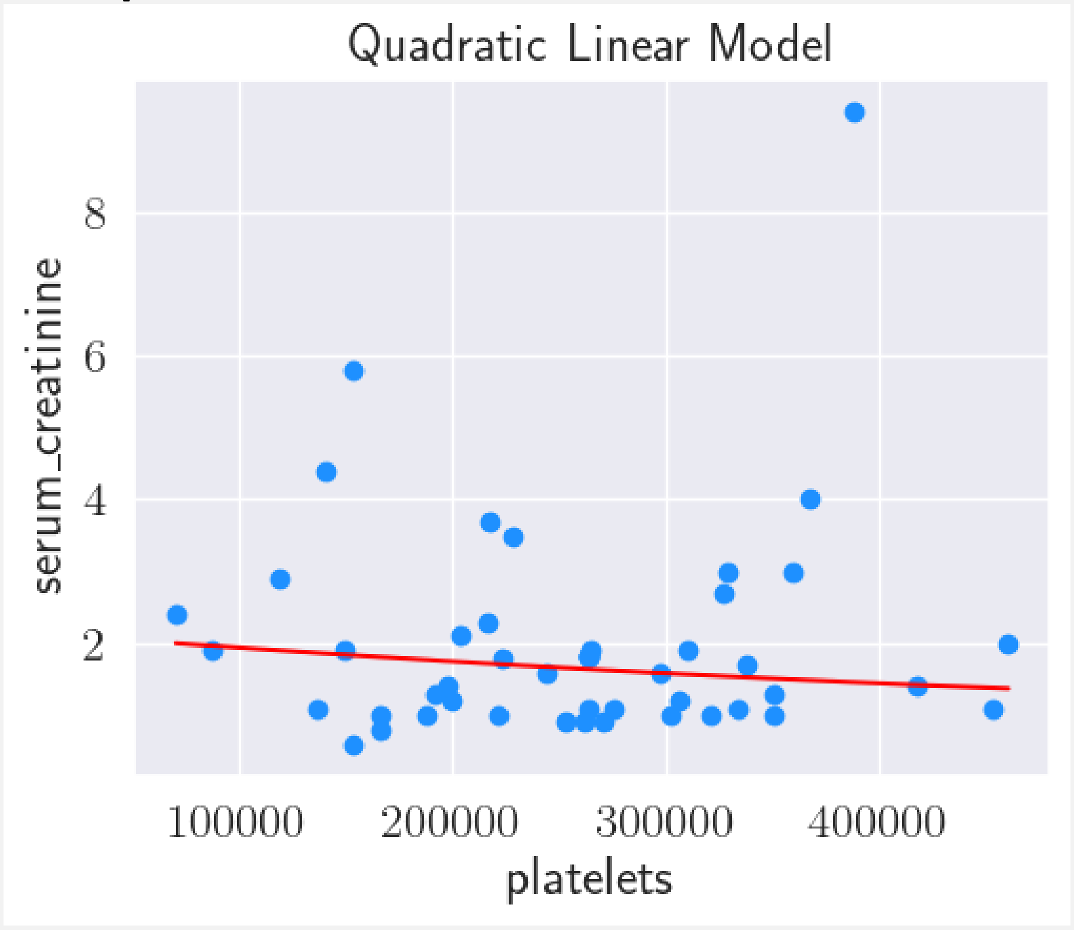
```
Simple linear coefficients: [-1.50858345e-06]
Sum of Squared Errors (SSE): 115.9134660820279
```



Cell 21

```python
# Metrics for quadratic linear regression model for deceased.
d_qlr_metrics: LinearModelMetrics = deceased_linear_model_analytics.quadratic
print(f"Quadratic coefficients: {d_qlr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {d_qlr_metrics.sse}")
d_qlr_metrics.plot("Quadratic Linear Model")
```

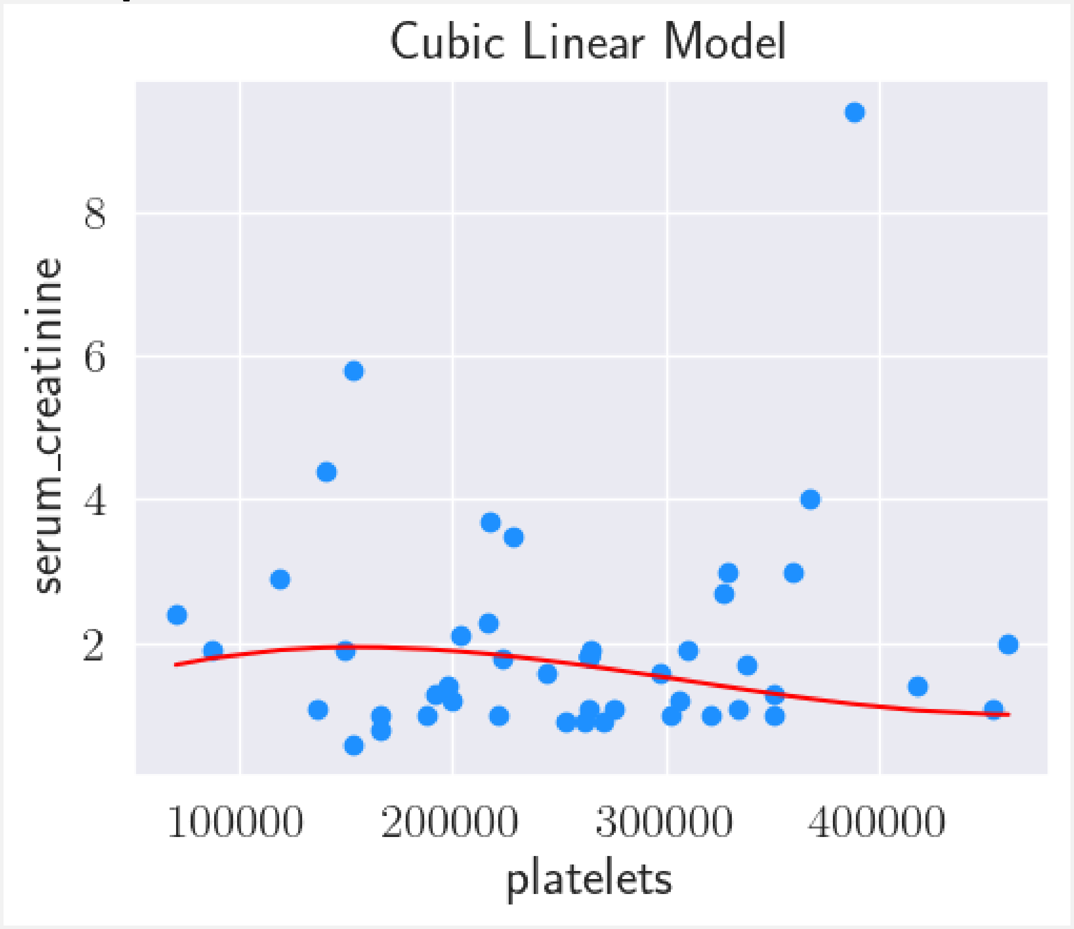
```
Quadratic coefficients: [-2.30057834e-06  1.30680369e-12]
Sum of Squared Errors (SSE): 115.93228954984559
```



Cell 22

```python
# Metrics for cubic linear regression model for deceased.
d_clr_metrics: LinearModelMetrics = deceased_linear_model_analytics.cubic
print(f"Cubic coefficients: {d_clr_metrics.coefficients}")
print(f"Sum of Squared Errors (SSE): {d_clr_metrics.sse}")
d_clr_metrics.plot("Cubic Linear Model")
```

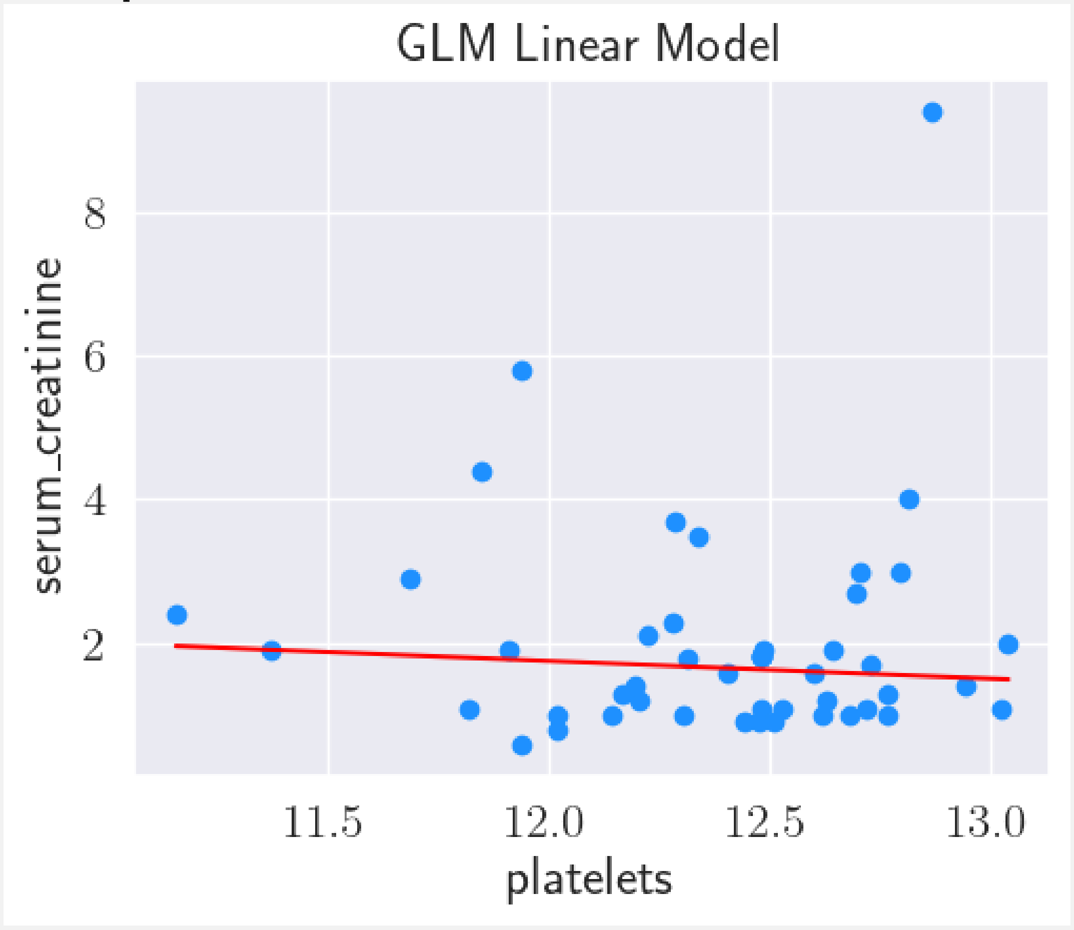
```
Cubic coefficients: [ 1.35239944e-05 -5.81138353e-11  6.23505583e-17]
Sum of Squared Errors (SSE): 123.9189215252172
```



Cell 23

```python
# Metrics for GLM #1 linear regression model for deceased.
d_glr_metrics1: LinearModelMetrics = deceased_linear_model_analytics.glm1
print(f"GLM coefficients: {d_glr_metrics1.coefficients}")
print(f"Sum of Squared Errors (SSE): {d_glr_metrics1.sse}")
d_glr_metrics1.plot("GLM Linear Model")
```

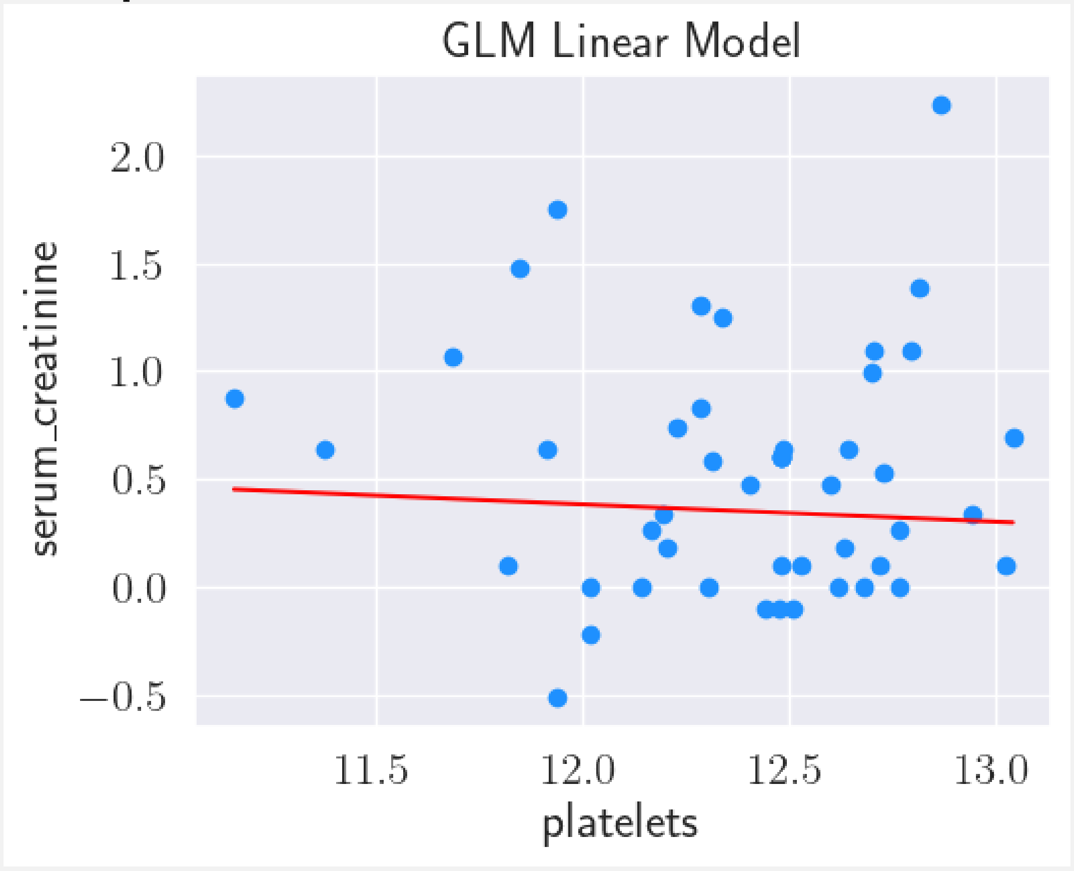
```
GLM coefficients: [-0.24702019]
Sum of Squared Errors (SSE): 114.70447814044553
```



Cell 24

```
# Metrics for GLM #2 linear regression model for deceased.
d_glr_metrics2: LinearModelMetrics = deceased_linear_model_analytics.glm2
print(f"GLM coefficients: {d_glr_metrics2.coefficients}")
print(f"Sum of Squared Errors (SSE): {d_glr_metrics2.sse}")
d_glr_metrics2.plot("GLM Linear Model")
```

```
GLM coefficients: [-0.08149637]
Sum of Squared Errors (SSE): 16.00486682454328
```



3 Question 3: Summarize your results from question 2 in a table like shown below:

Answer:

Cell 25

```
# Summarize the results of SSE of all models in a table.
sse_results: DataFrame = LinearModelAnalytics.sse_table(
    survivors_linear_model_analytics,
    deceased_linear_model_analytics
)
```

Cell 26

```
# Build latex SSE table to render in the document.
sse_table = create_latex_table(
    sse_results,
    label="tab:question3",
    caption="Question 3"
)
Latex(sse_table)
```

Table 1: Question 3

	Model	SSE (death_event=0)	SSE (death_event=1)
0	$y = ax + b$	33.850874	115.913466
1	$y = ax^2 + bx + c$	38.036190	115.932290
2	$y = ax^3 + bx^2 + cx + d$	52.425793	123.918922
3	$y = a \log x + b$	34.021225	114.704478
4	$\log y = a \log x + b$	11.441515	16.004867

3.1 Which model was the best (smallest SSE) for surviving patients? for deceased patients?

Answer:

Cell 27

```
# Get the best model for the survivors.
s_best_model = survivors_linear_model_analytics.best_model
print(
    f"The best model for the survivors is {s_best_model.name}"
    f" with an SSE value of "
    f"\n{s_best_model.sse:.3f}"
)

# Get the best model for the deceased.
d_best_model = deceased_linear_model_analytics.best_model
print(
    f"The best model for the deceased is {d_best_model.name}"
    f" with an SSE value of "
    f"\n{d_best_model.sse:.3f}"
)
```

```
The best model for the survivors is GLM #2 Model with an SSE value of
11.442
The best model for the deceased is GLM #2 Model with an SSE value of
16.005
```

3.2 Which model was the worst (largest SSE) for surviving patients? for deceased patients?

Answer:

Cell 28

```python
# Get the worst model for the survivors.
s_worst_model = survivors_linear_model_analytics.worst_model
print(
    f"The worst model for the survivors is {s_worst_model.name}"
    f" with an SSE value of "
    f"\n{s_worst_model.sse:.3f}"
)

# Get the worst model for the deceased.
d_worst_model = deceased_linear_model_analytics.worst_model
print(
    f"The worst model for the deceased is {d_worst_model.name}"
    f" with an SSE value of "
    f"\n{d_worst_model.sse:.3f}"
)
```

```
The worst model for the survivors is Cubic Spline Model with an SSE value of
52.426
The worst model for the deceased is Cubic Spline Model with an SSE value of
123.919
```

References

[1] 1.1. Linear Models — scikit-learn.org. **https://scikit-learn.org/stable/modules/linear_model.html**. [Accessed 09-Apr-2023].

[2] 3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn.org. **https://scikit-learn.org/stable/modules/model_evaluation.html**. [Accessed 10-Apr-2023].

[3] Coefficient of Determination: How to Calculate It and Interpret the Result — investopedia.com. **https://www.investopedia.com/terms/c/coefficient-of-determination.asp**. [Accessed 11-Apr-2023].

[4] Common pitfalls in the interpretation of coefficients of linear models — scikit-learn.org. **https://scikit-learn.org/stable/auto_examples/inspection/plot_linear_model_coefficient_interpretation.html#sphx-glr-auto-examples-inspection-plot-linear-model-coefficient-interpretation-py**. [Accessed 10-Apr-2023].

[5] Linear Regression Example — scikit-learn.org. **https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html**. [Accessed 09-Apr-2023].

[6] Polynomial and Spline interpolation — scikit-learn.org. **https://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html#sphx-glr-auto-examples-linear-model-plot-polynomial-interpolation-py**. [Accessed 10-Apr-2023].

[7] Residual Sum of Squares (RSS): What It Is, How to Calculate It — investopedia.com. **https://www.investopedia.com/terms/r/residual-sum-of-squares.asp**. [Accessed 11-Apr-2023].

[8] The Correlation Coefficient: What It Is, What It Tells Investors — investopedia.com.

https://www.investopedia.com/terms/c/correlationcoefficient.asp. [Accessed 11-Apr-2023].

[9] Shwetha Acharya. What are RMSE and MAE? — towardsdatascience.com. **https://towardsdatascience.com/what-are-rmse-and-mae-e405ce230383**. [Accessed 11-Apr-2023].

[10] Christian Lee. How least squares regression estimates are actually calculated — towardsdatascience.com. **https://towardsdatascience.com/how-least-squares-regression-estimates-are-actually-calculated-662d237a4d7e**. [Accessed 11-Apr-2023].

[11] Ted Petrou. Automatically Wrap Graph Labels in Matplotlib and Seaborn — medium.com. **https://medium.com/dunder-data/automatically-wrap-graph-labels-in-matplotlib-and-seaborn-a48740bc9ce**. [Accessed 09-Apr-2023].

[12] Karun Thankachan. Should you use pandas correlation function? — towardsdatascience.com. **https://towardsdatascience.com/should-you-use-pandas-corr-function-af82c454bc0**. [Accessed 09-Apr-2023].

[13] Tamas Ujhelyi. Polynomial Regression in Python using scikit-learn (with example) — data36.com. **https://data36.com/polynomial-regression-python-scikit-learn/**. [Accessed 10-Apr-2023].

[14] Zach. How to Interpret the Intercept in a Regression Model (With Examples) — statology.org. **https://www.statology.org/intercept-in-regression/**. [Accessed 11-Apr-2023].

[15] Zach. How to Read a Correlation Matrix - Statology — statology.org. **https://www.statology.org/how-to-read-a-correlation-matrix/**. [Accessed 09-Apr-2023].