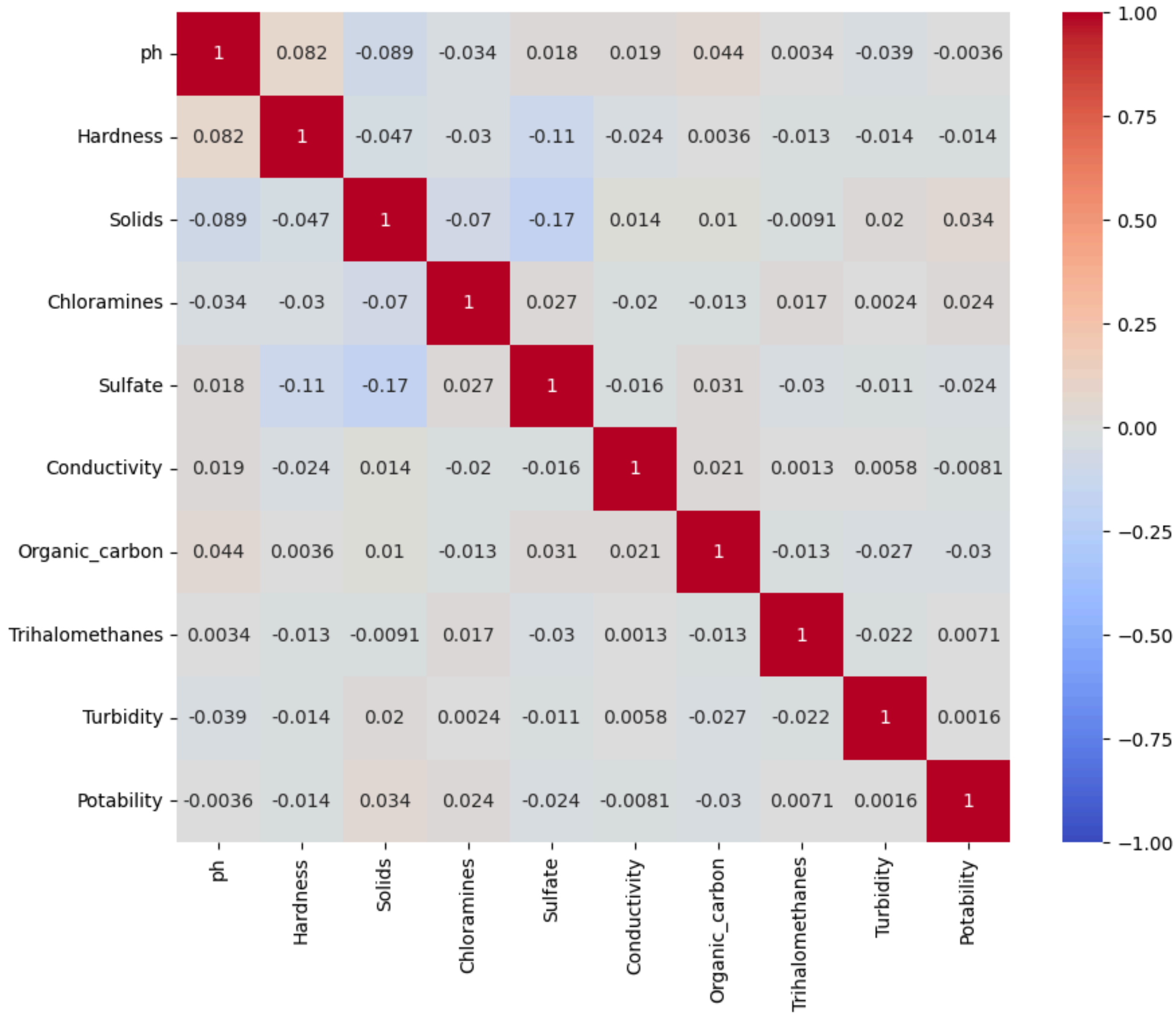
Assignment: 1

MLFlow implementation:

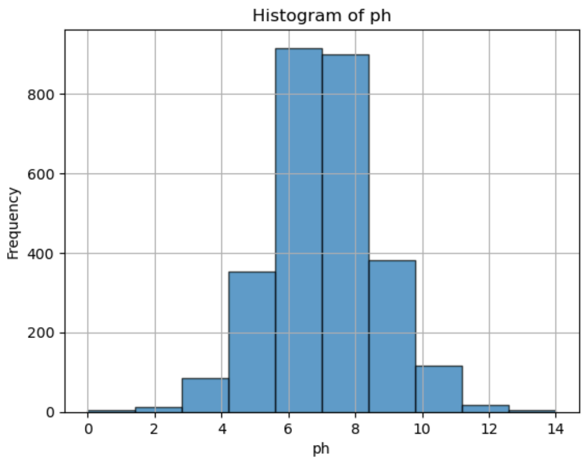
1. **Data Collection & Preprocessing:**

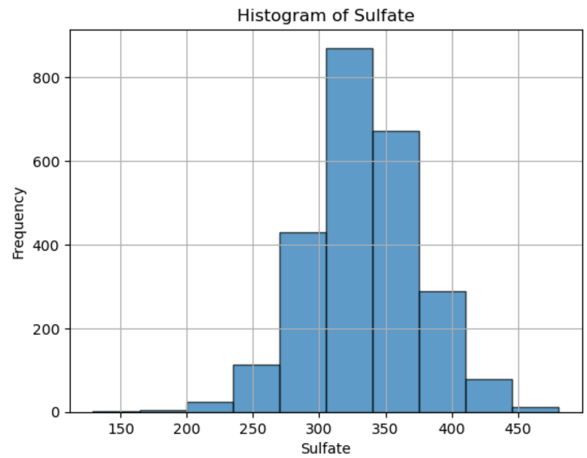
- We have obtained water_potability data from kaggle([water_potability kaggle data]). In that data there are various features (ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon, Trihalomethanes and Turbidity) that determined that the water is drinkable or not that shows in potability column.
- So, first determine that which kind of the data in the column. So using the data.info() and data.describe() we can determine the basic things like statistics and data types. So, Here in the data set all independent features are in float data type. And dependent feature is integer data type.
- And then determine the heatmap of correlation matrix but we cannot find anything specific in that. All features are uniquely distinguish.

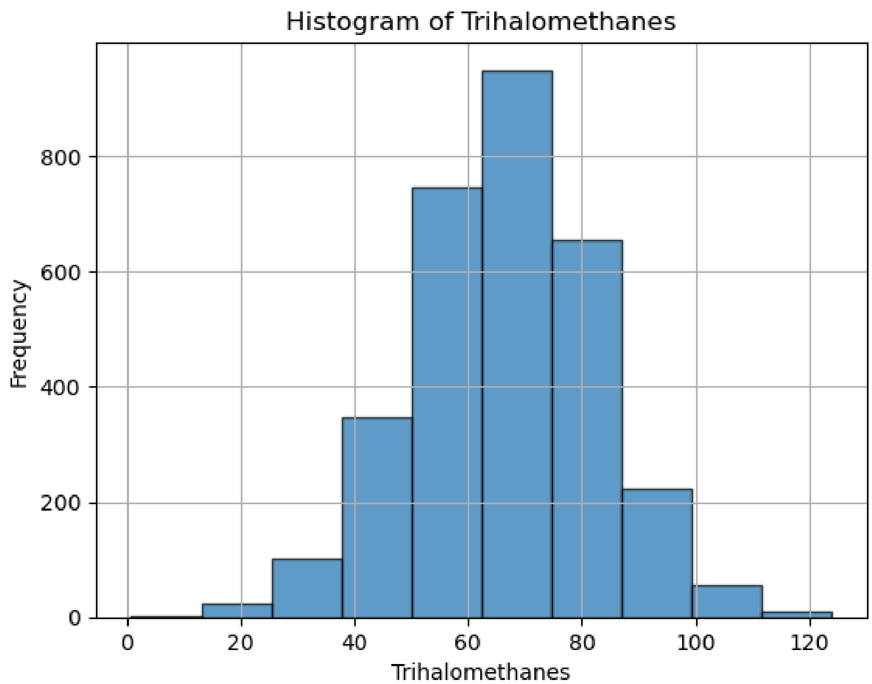


	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1	0.082	-0.089	-0.034	0.018	0.019	0.044	0.0034	-0.039	-0.0036
Hardness	0.082	1	-0.047	-0.03	-0.11	-0.024	0.0036	-0.013	-0.014	-0.014
Solids	-0.089	-0.047	1	-0.07	-0.17	0.014	0.01	-0.0091	0.02	0.034
Chloramines	-0.034	-0.03	-0.07	1	0.027	-0.02	-0.013	0.017	0.0024	0.024
Sulfate	0.018	-0.11	-0.17	0.027	1	-0.016	0.031	-0.03	-0.011	-0.024
Conductivity	0.019	-0.024	0.014	-0.02	-0.016	1	0.021	0.0013	0.0058	-0.0081
Organic_carbon	0.044	0.0036	0.01	-0.013	0.031	0.021	1	-0.013	-0.027	-0.03
Trihalomethanes	0.0034	-0.013	-0.0091	0.017	-0.03	0.0013	-0.013	1	-0.022	0.0071
Turbidity	-0.039	-0.014	0.02	0.0024	-0.011	0.0058	-0.027	-0.022	1	0.0016
Potability	-0.0036	-0.014	0.034	0.024	-0.024	-0.0081	-0.03	0.0071	0.0016	1

- Here plot the histogram for the feature who contains the null value.







- After that find the Null, NA and Nan value in the column and replace by mean or median. Here data is normally distributed(As we can see in the above plots of histogram) for feature ph,Sulfate and Trihalomethanes so we can replace the null or na value with the mean.
- And to scale the feature with outliers I’ve used MinMaxScaler.

2. **Model Implementation:**

Ridge Regression:

- Ridge Regression extends linear regression to improve model robustness. It introduces L2 regularization, adding a penalty based on the sum of squared coefficients (scaled by the hyperparameter “alpha”).
- By adjusting alpha, we control the trade-off between fitting the data well and keeping coefficients small. Ridge is particularly useful for handling multicollinearity and stabilizing vanilla linear regression against outliers and overfitting.

Lasso Regression:

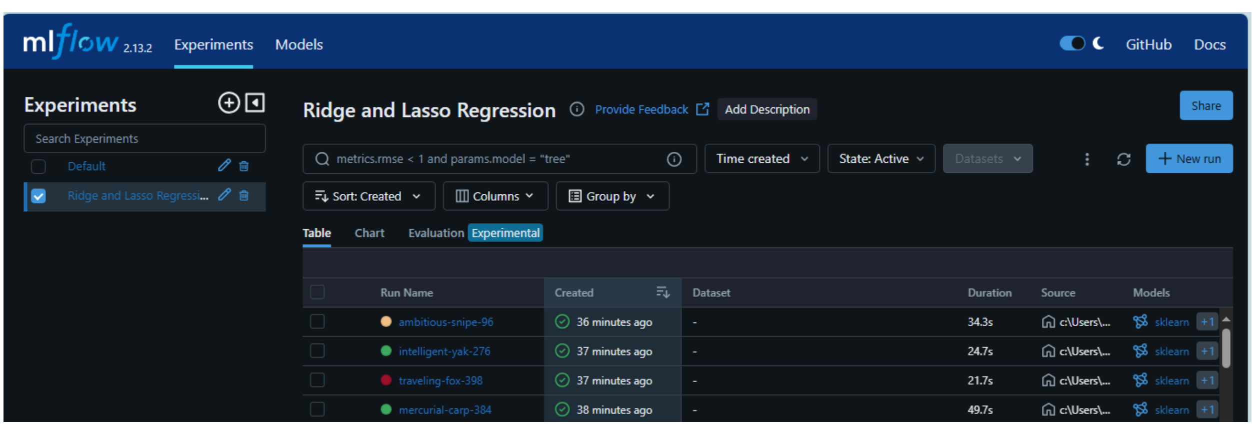
- Lasso Regression is another regularization technique which I’ve applied. It encourages sparse models by using L1 regularization.
- The L1 term penalizes absolute coefficient values, leading some coefficients to become exactly zero (feature selection). Lasso simplifies models and prevents overfitting, similar to Ridge.

Result Table:

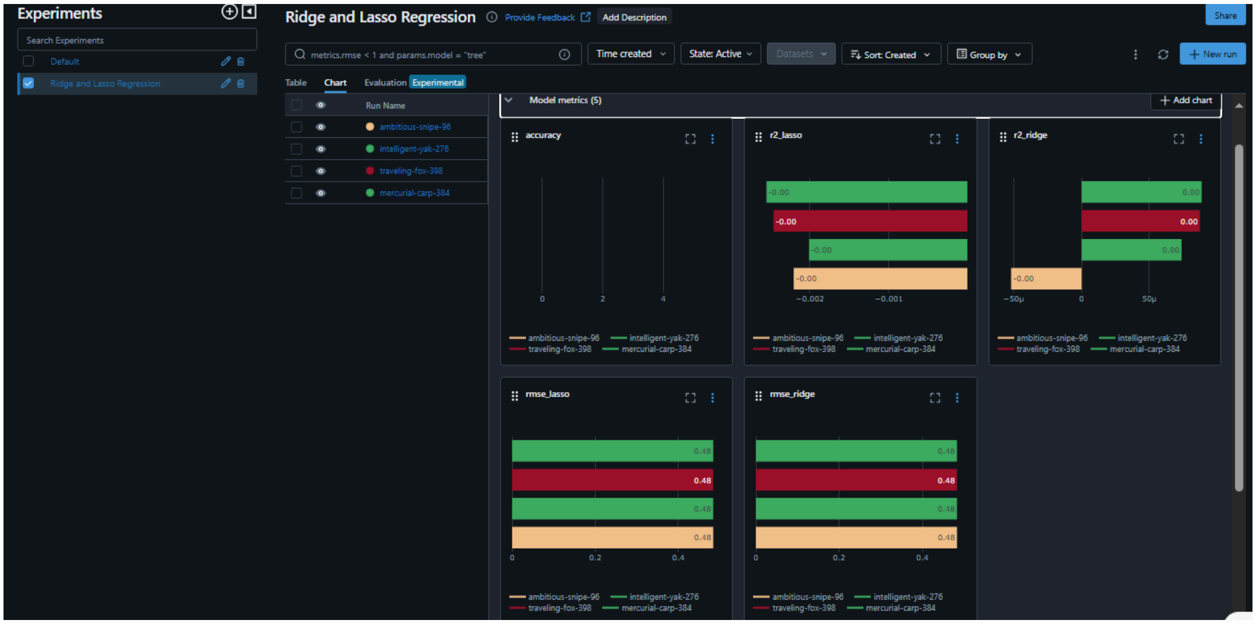
Alpha Value	Ridge Regression		Lasso Regression	
	RMSE	R-squared	RMSE	R-squared
1	0.4833	0.0001	0.4839	-0.0026
10	0.4833	0.0001	0.4839	-0.0025
100	0.4833	0.0001	0.4838	-0.0020
1000	0.4833	-0.0001	0.4839	-0.0022

3. **MLflow Integration:**

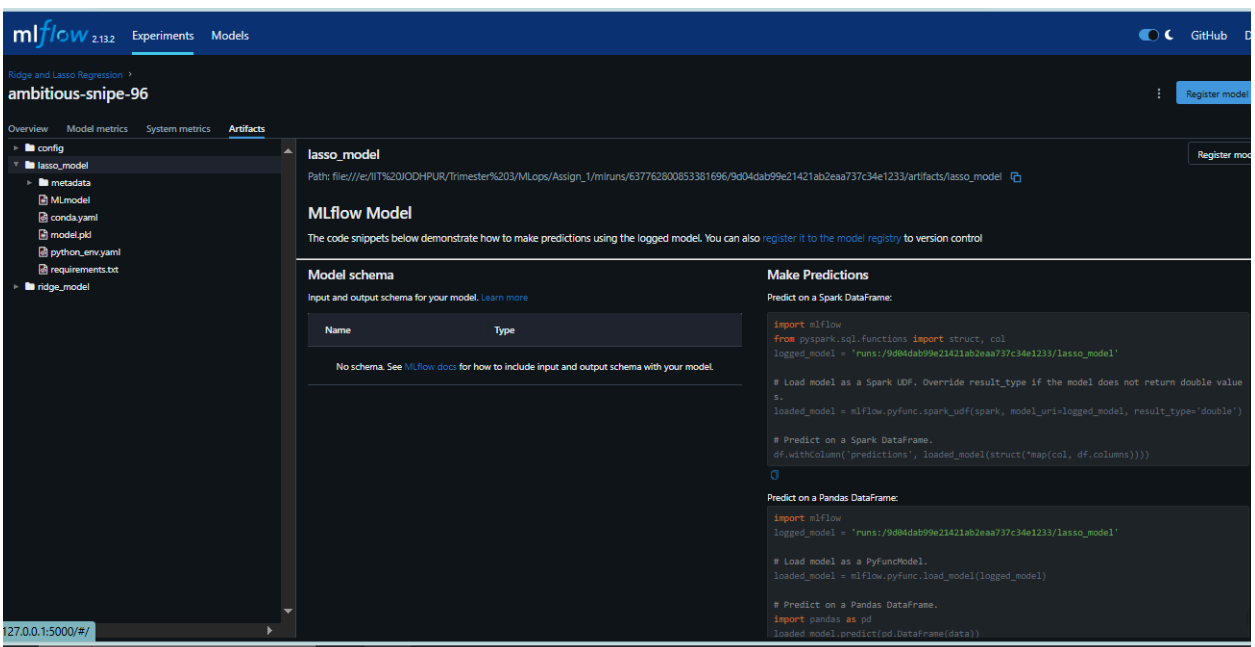
Dashboard:

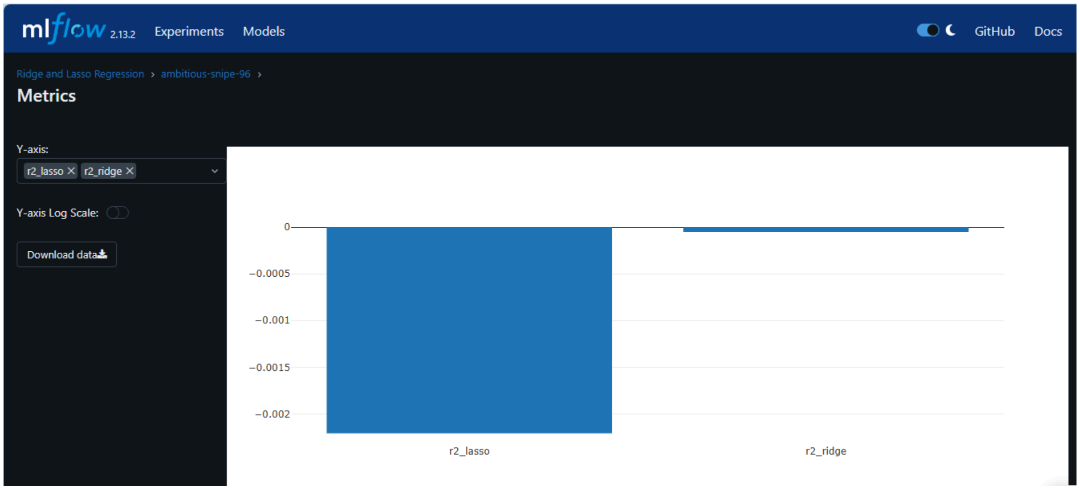


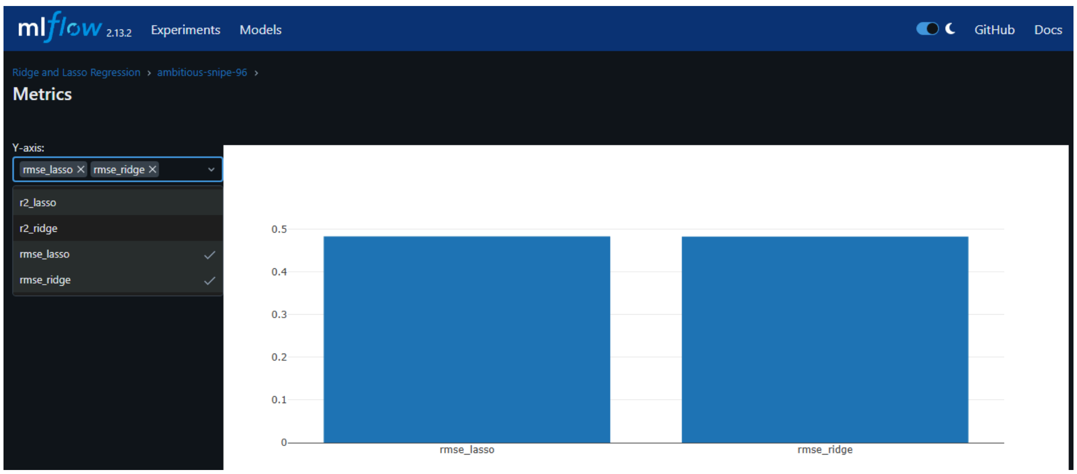
Model metrics:



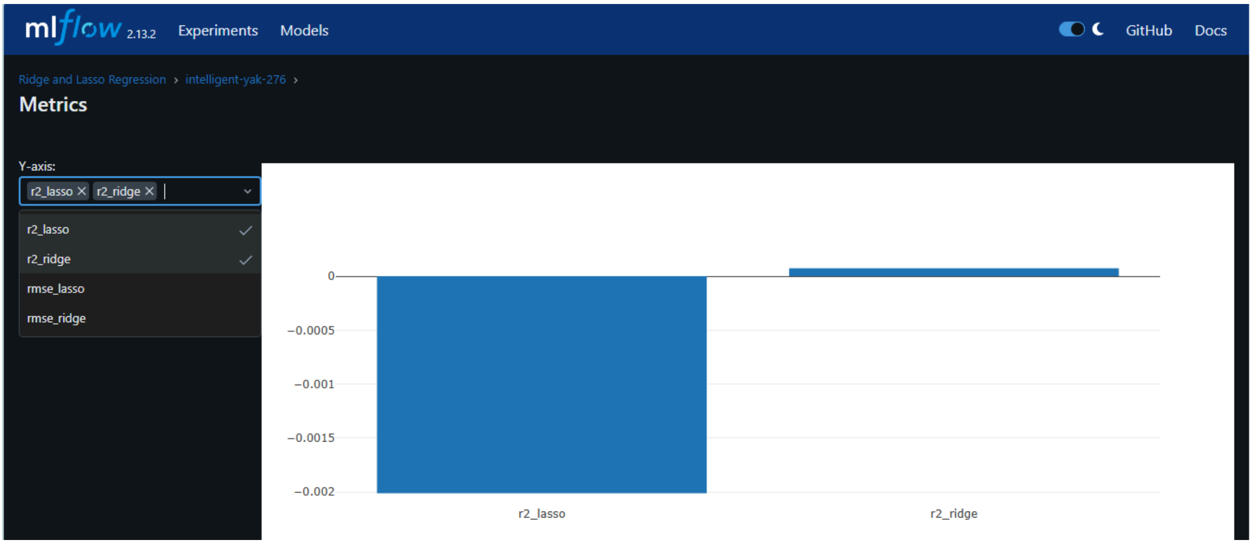
Model 1 metrics (Alpha = 1000):

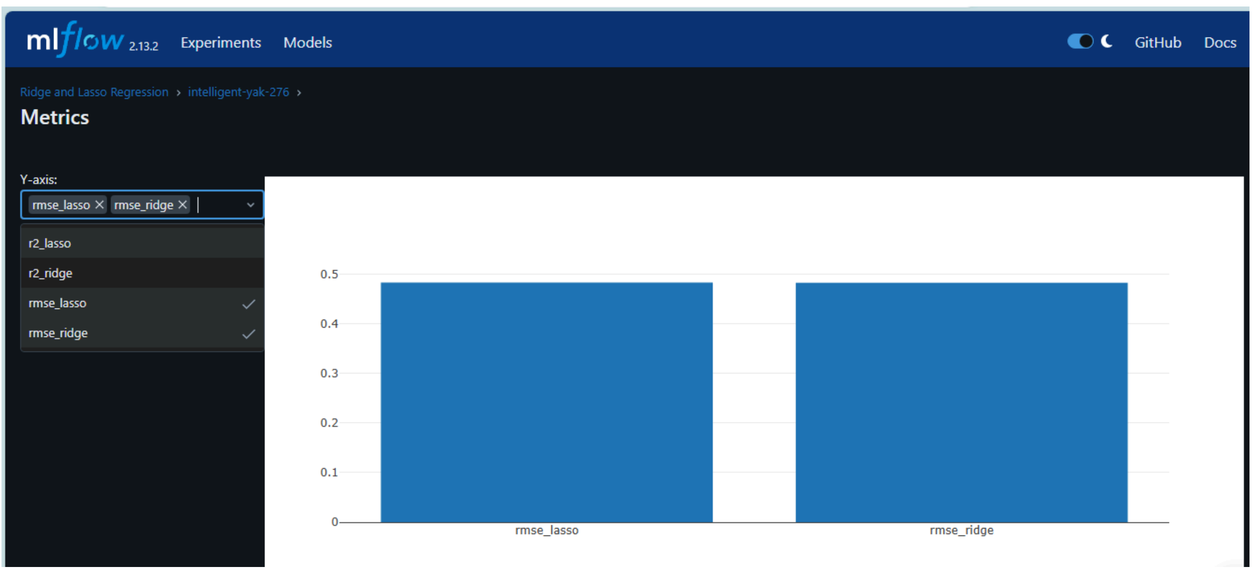



mlflow 2.13.2
Experiments
Models
GitHub
Docs
Ridge and Lasso Regression › ambitious-snipe-96 ›
Metrics
Y-axis:
r2_lasso
r2_ridge
Y-axis Log Scale:
Download data
0
−0.0005
−0.001
−0.0015
−0.002
r2_lasso
r2_ridge

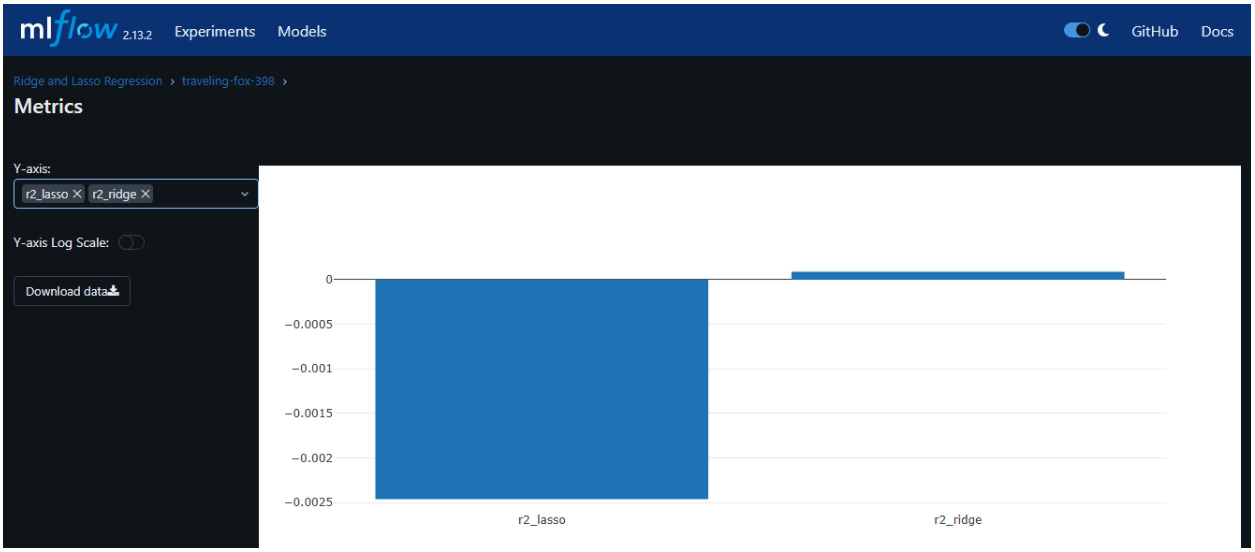

mlflow 2.13.2
Experiments
Models
GitHub
Docs
Ridge and Lasso Regression › ambitious-snipe-96 ›
Metrics
Y-axis:
rmse_lasso
rmse_ridge
r2_lasso
r2_ridge
rmse_lasso
rmse_ridge
0.5
0.4
0.3
0.2
0.1
0
rmse_lasso
rmse_ridge

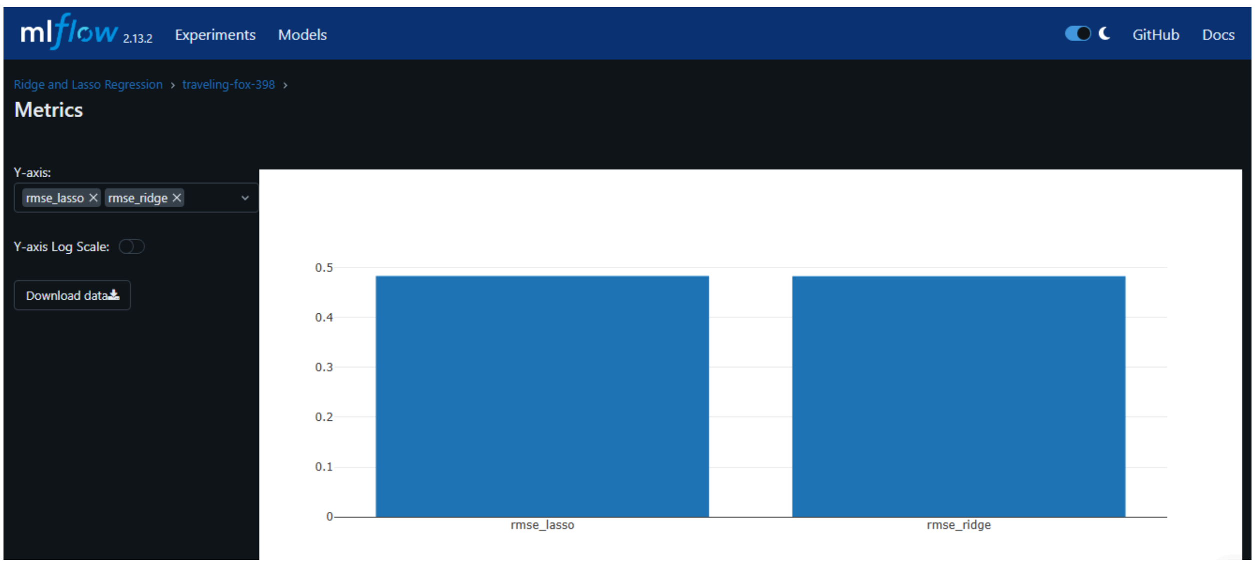
Model 2 metrics (alpha = 100):





Model 3 metrics (alpha = 10)





Model 4 metrics (Alpha = 1):