COMP551 Assignment 1: Analysis and Discussion

Thomas Lewis, Gabe Woloz, Vivek Motta

February 2026

Abstract

In this project, we analyze the Capitol Bike Sharing dataset and implement and train a linear regression model on the data in order to predict the total count (i.e. number of rentals) based on environmental data. We cleaned and processed our data to better fit our model and then, using the closed form equation for linear regression, we created the model and trained it. We finally applied feature engineering techniques such as polynomial, log, interactions and sine cosine transformations on the features and compared the resulting data. We found that essentially all of our feature engineering resulting in better performance of the model (lower MSE) compared to the unchanged data.

Introduction

The dataset used for this project captures the daily and hourly usage of a bike sharing program in Washington D.C. between 2011 and 2012. It contains environmental information in the area as well as the number of bikes being rented at the time. This project aims to train a simple linear regression model on this data to accurately predict the rental count based on the other features such as current weather conditions and date.

Data Preprocessing and Exploration

Handling missing values

As the dataset's website stated, there were no missing values in the data. We confirmed this with Pandas.

Cleaning

We dropped the 'casual' and 'registered' features as their sum is equal to the target, 'cnt'. This prevents these features from leaking the target when training.

Scaling and encoding choices

We encoded weather and season with one-hot as they are categorical, non-ordinal features. We chose not to scale any features as the histograms we generated for the continuous features showed no significant skew or scaling issues (see figure 2).

Effect on numerical stability

Numerical stability should decrease with one-hot encoding since more columns/features are being considered with the risk of collinearity.

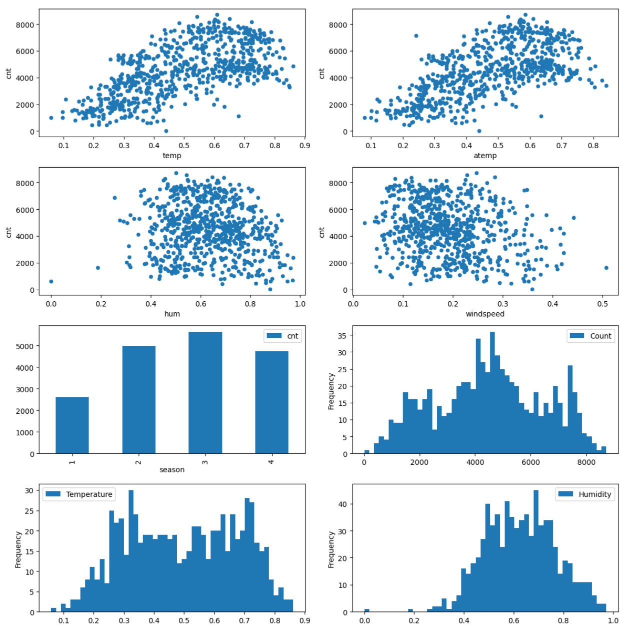


Figure 1: Data Visualization Plots

Methods

Linear Regression Implementation

We used the closed form equation discussed in the lecture to implement our linear regression model, where instead of inverting the matrix, least squares (pseudo inverse) was used to keep numerical stability.

$$w^* = (X^\top X)^{-1} X^\top y \rightarrow X^\dagger y$$

Transformations evaluated

We evaluated polynomial, log, interaction, and sine cosine transformations. Each choice was guided by patterns we observed directly in the data. In particular, squared (polynomial) features were added for the continuous environmental variables ('temp', 'atemp', 'hum', 'windspeed') because the scatter plots showed clear curved relationships with rental counts rather than straight lines. As shown in Figure 2 "Temperature vs Count with quadratic fit", temperature follows an inverted-U trend, where rentals peak at moderate temperatures and drop off at both low and high extremes. The quadratic fit highlights this non-linear behavior that a simple linear term cannot capture. Including squared terms allows linear regression to better approximate these curved trends.

We applied a log transformation to 'hum' to compress its scale and check whether changes in lower humidity levels (for example, 20% to 40%) have a larger impact than similar changes at higher levels (60% to 80%), suggesting a proportional rather than additive effect.

Interaction terms were also created between environmental pairs ('temp' $\times$ 'hum', 'temp' $\times$ 'windspeed') to model combined effects. Figure 2 "Temp–Humidity interaction colored by count" shows why this is useful. Rental demand depends on specific combinations of temperature and humidity, not just either variable on its own. For example, high temperatures may feel comfortable in low humidity but uncomfortable in high humidity. Finally, sine and cosine transformations were applied to cyclical time features ('mnth', 'weekday', 'hr') to reflect their periodic structure. Figure 2 "Average rentals by month (seasonal pattern)" demonstrates this seasonal behavior. Without cyclical encoding, the model treats hour 23 and hour 0 as far apart even though they are consecutive in time, which makes it harder to capture patterns like commute peaks or the transition from December to January.

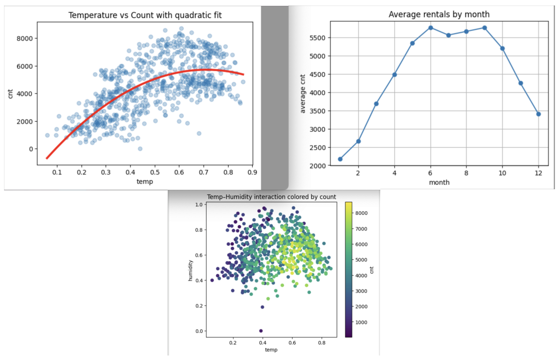


Figure 2: Data Visualization Plots Extended

Results

Effect of feature engineering on model performance

Most engineered features reduced the model's mean squared error, although the improvements varied. The largest gains came from the polynomial (squared) features ('temp','atemp','hum','windspeed'). These variables showed stronger curved relationships with "cnt" compared to the other engineered features. As seen in Figure 2 "Temperature vs Count with quadratic fit", rentals clearly peak at moderate temperatures, and the quadratic fit captures this inverted-U pattern that a linear model alone would miss. This explains why squared terms led to a noticeable drop in MSE.

The interaction terms ('temp' $\times$ 'hum', 'temp' $\times$ 'windspeed') produced moderate MSE improvements, especially for the hourly dataset (Figure 3). Figure 2 "Temp–Humidity interaction colored by count" illustrates how these features work together to influence demand. High rental counts occur in specific temperature-humidity combinations rather than being determined by either variable independently. Their smaller gains compared to polynomial features indicate that while interactions matter, the primary non-linearity is captured by squared terms.

The log transformation of humidity gave mixed results. While it slightly reduced MSE in some cases, it also increased the condition number, introducing multicollinearity without much predictive value. This suggests humidity's relationship with rentals is not logarithmic.

The sine cosine features ('month', 'weekday', 'hr') had a strong effect on hourly data but increased MSE for daily data. This difference comes from how cyclical patterns appear in each dataset. For hourly data, these transformations reduced MSE by capturing diurnal patterns and the continuity between hour 23 and hour 0. For daily data, Figure 2 "Average rentals by month (seasonal pattern)" shows more discrete seasonal regimes rather than smooth sinusoidal cycles. This reflects the assumption that months and weekdays have a purely cyclical effect on "cnt", while in practice specific months and weekends, which are better captured with one-hot encoding, have a larger effect. Hours in a day, however, follow a clear cycle and strongly affect total count. This shows feature engineering should match the data generation process, using cyclical encoding for truly periodic patterns and categorical encoding for distinct regimes.

Numerical stability of the model

The numerical stability of the model is considered moderately conditioned (between 10^2 and 10^6)[1] Each feature that was add increased the condition number, which decreased the numerical stability. The log transformation made the most significant difference in numerical stability, multiplying it five to ten times. This result can be explained by the feature selected ("hum"), which has no visible log relation to the target feature "cnt". This points to its weight probably being close to zero, which would increase the condition number and decrease numerical stability.
It is also important to note the lower condition number of the hour data. This is attributed to the higher sample to feature ratio which reduces the impact of localized multicollinearity.
[1]https://www.cs.usask.ca/ spiteri/CMPT898/notes/numericalStability.pdf

Discussion and Conclusion

Signs of overfitting or capacity change

Overfitting was measured with the ratio between the training and testing MSEs. Sine cosine transformations either reduced overfitting or kept it stable while others only increased it. This can be seen in the table in figure 3.

Practical improvements and further experiments

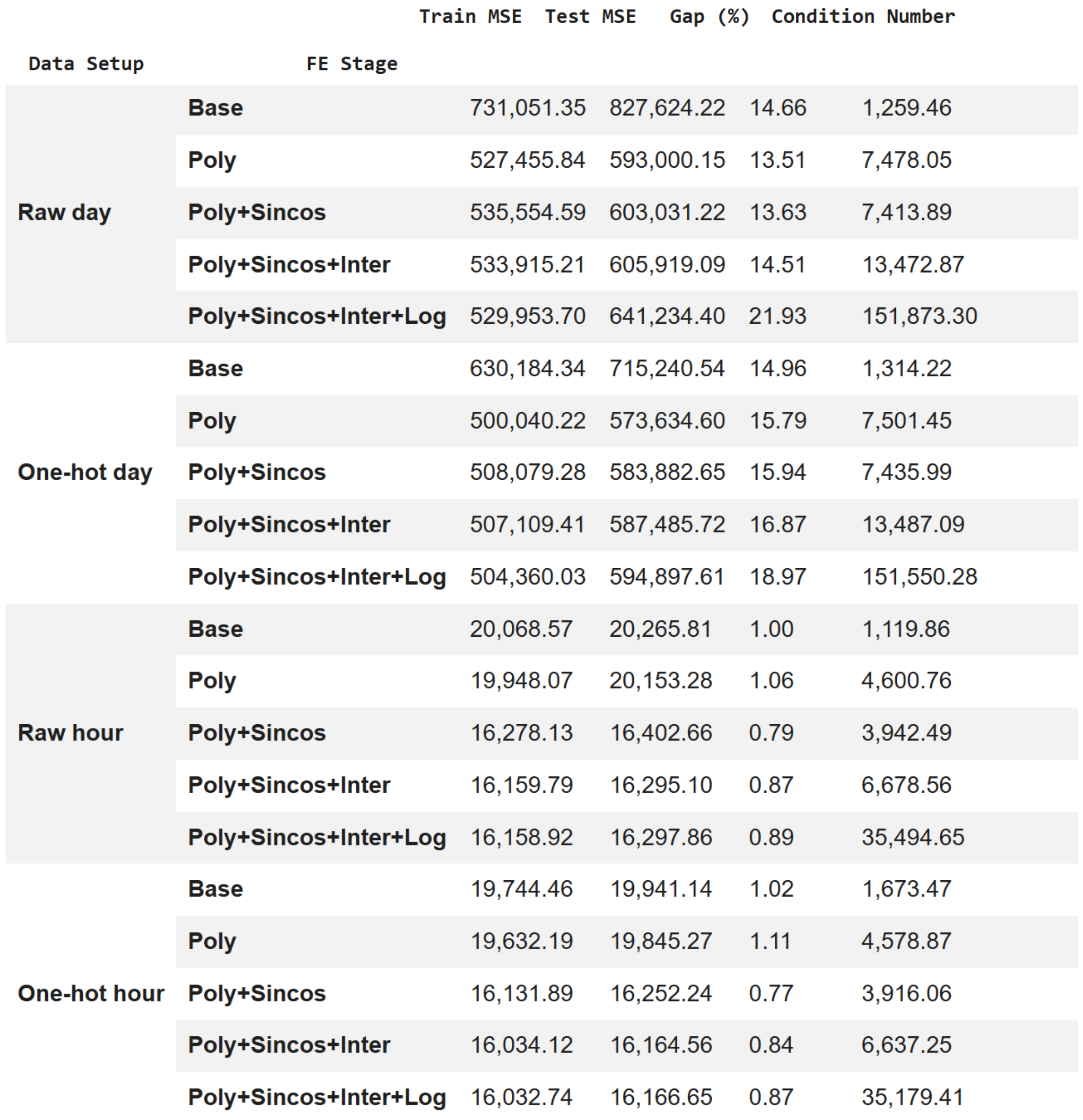
We could possibly experiment with creating new features with either of the strategies used in this experiment. Others, such as scaling, also have to potential to affect MSE even though very little skew was observed in the data. Finally, we could explore more advanced strategies such as regularization and observe it's effects on numerical stability and MSE.

Statement of Contributions

Thomas worked on tasks 2 and 3 and contributing to both the data preprocessing and the writeup document. Gabe worked on preprocessing and encoding as well as formatting data and creating plots for the data visualization section. Vivek worked on the writeup and verified what was done on other tasks.

Statement on the Use of LLMs

LLMs were used to debug code (find and explain errors in code) and to help interpret data that (Ex. Adding a feature leading to significantly worse numerical stability).



Data Setup	FE Stage	Train MSE	Test MSE	Gap (%)	Condition Number
Raw day	Base	731,051.35	827,624.22	14.66	1,259.46
	Poly	527,455.84	593,000.15	13.51	7,478.05
	Poly+Sincos	535,554.59	603,031.22	13.63	7,413.89
	Poly+Sincos+Inter	533,915.21	605,919.09	14.51	13,472.87
	Poly+Sincos+Inter+Log	529,953.70	641,234.40	21.93	151,873.30
One-hot day	Base	630,184.34	715,240.54	14.96	1,314.22
	Poly	500,040.22	573,634.60	15.79	7,501.45
	Poly+Sincos	508,079.28	583,882.65	15.94	7,435.99
	Poly+Sincos+Inter	507,109.41	587,485.72	16.87	13,487.09
	Poly+Sincos+Inter+Log	504,360.03	594,897.61	18.97	151,550.28
Raw hour	Base	20,068.57	20,265.81	1.00	1,119.86
	Poly	19,948.07	20,153.28	1.06	4,600.76
	Poly+Sincos	16,278.13	16,402.66	0.79	3,942.49
	Poly+Sincos+Inter	16,159.79	16,295.10	0.87	6,678.56
	Poly+Sincos+Inter+Log	16,158.92	16,297.86	0.89	35,494.65
One-hot hour	Base	19,744.46	19,941.14	1.02	1,673.47
	Poly	19,632.19	19,845.27	1.11	4,578.87
	Poly+Sincos	16,131.89	16,252.24	0.77	3,916.06
	Poly+Sincos+Inter	16,034.12	16,164.56	0.84	6,637.25
	Poly+Sincos+Inter+Log	16,032.74	16,166.65	0.87	35,179.41

Figure 3: MSE data