Uber and Lyft Ride Data

Mobina Kargar dymamsijhidjj@gmail.com

Abstract

In this assignment, you will explore and apply linear regression techniques to model and predict ride prices using a real-world dataset from Uber and Lyft services in Boston, MA.The dataset contains hundreds of thousands of ride entries with various contextual features.Your goal is to build and evaluate regression models that estimate ride prices accurately and to investigate which types of linear regression approaches yield the best results.

1 Introduction

First step to have better overview of what should we do is to check data in csv file and if necessary, remove or correct the anomalies and s.th like that, that I will show you in the next step in my report in details.

2 Features

The data is composed of 57 columns and 693071entries (Full train dataset shape is (693071, 57)). We can see all 57 dimensions of our dataset by printing out the first 3 entries:

Table 1: train dataset (3 rows x 57 columns)

	id	timestamp	hour	day	...	apparentTemperatureMin	apparentTemperatureMinTime	apparentTemperatureMax	apparentTemperatureMaxTime
0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	1544952607.89	9	16	...	33.73	1545012000	38.07	1544958000
1	4bd23055-6827-41c6-b23b-3c491f24e74d	1543284023.677	2	27	...	36.2	1543291200	43.92	1543251600
2	981a3613-77af-4620-a42a-0c0866077d1e	1543366822.198	1	28	...	31.04	1543377600	44.12	1543320000

We can inspect the types of feature columns:

Table 2: Data columns:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693071 entries, 0 to 693070
Data columns (total 57 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   id                           693071 non-null  object
 1   timestamp                    693071 non-null  float64
 2   hour                         693071 non-null  int64
 3   day                          693071 non-null  int64
 4   month                        693071 non-null  int64
 5   datetime                     693071 non-null  object
 6   timezone                     693071 non-null  object
 7   source                       693071 non-null  object
 8   destination                  693071 non-null  object
 9   cab_type                     693071 non-null  object
 10  product_id                   693071 non-null  object
 11  name                         693071 non-null  object
 12  price                        637976 non-null  float64
 13  distance                     693071 non-null  float64
 14  surge_multiplier             693071 non-null  float64
 15  latitude                     693071 non-null  float64
 16  longitude                    693071 non-null  float64
 17  temperature                  693071 non-null  float64
 18  apparentTemperature          693071 non-null  float64
 19  short_summary                693071 non-null  object
 20  long_summary                 693071 non-null  object
 21  precipIntensity              693071 non-null  float64
 22  precipProbability            693071 non-null  float64
 23  humidity                     693071 non-null  float64
 24  windSpeed                    693071 non-null  float64
 25  windGust                     693071 non-null  float64
 26  windGustTime                 693071 non-null  int64
 27  visibility                   693071 non-null  float64
 28  temperatureHigh              693071 non-null  float64
 29  temperatureHighTime          693071 non-null  int64
 30  temperatureLow               693071 non-null  float64
 31  temperatureLowTime           693071 non-null  int64
 32  apparentTemperatureHigh      693071 non-null  float64
 33  apparentTemperatureHighTime  693071 non-null  int64
 34  apparentTemperatureLow       693071 non-null  float64
 35  apparentTemperatureLowTime   693071 non-null  int64
 36  icon                         693071 non-null  object
 37  dewPoint                     693071 non-null  float64
 38  pressure                     693071 non-null  float64
 39  windBearing                  693071 non-null  int64
 40  cloudCover                 693071 non-null  float64
 41  uvIndex                    693071 non-null  int64
 42  visibility.1               693071 non-null  float64
 43  ozone                      693071 non-null  float64
 44  sunriseTime                693071 non-null  int64
 45  sunsetTime                 693071 non-null  int64
 46  moonPhase                  693071 non-null  float64
 47  precipIntensityMax         693071 non-null  float64
 48  uvIndexTime                693071 non-null  int64
 49  temperatureMin             693071 non-null  float64
 50  temperatureMinTime         693071 non-null  int64
 51  temperatureMax             693071 non-null  float64
 52  temperatureMaxTime         693071 non-null  int64
 53  apparentTemperatureMin     693071 non-null  float64
 54  apparentTemperatureMinTime 693071 non-null  int64
 55  apparentTemperatureMax     693071 non-null  float64
 56  apparentTemperatureMaxTime 693071 non-null  int64
dtypes: float64(29), int64(17), object(11)
memory usage: 301.4+ MB
```

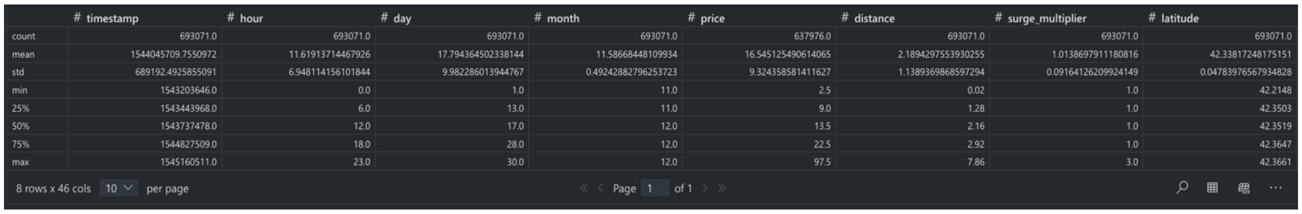
3 Distribution

3.1 Handling Missing Values & Duplicates:

In this part the result become [637976 rows x 57 columns] by removing missing values(because they are lower than half of the code) and remove duplicate (I mean one of the 'visibility' column)

3.2 Detect Outliers (remove)

By using 25% and 75% from image below we can easily detect outliers and because of the rare outliers we can remove them. And the result become [86093 rows x 57 columns]



	# timestamp	# hour	# day	# month	# price	# distance	# surge_multiplier	# latitude
count	693071.0	693071.0	693071.0	693071.0	637976.0	693071.0	693071.0	693071.0
mean	1544045709.7550972	11.61913714467926	17.794364502338144	11.58668448109934	16.545125490614065	2.1894297553930255	1.0138697911180816	42.33817248175151
std	689192.4925855091	6.948114156101844	9.982286013944767	0.49242882796253723	9.324358581411627	1.1389369868597294	0.09164126209924149	0.04783976567934828
min	1543203646.0	0.0	1.0	11.0	2.5	0.02	1.0	42.2148
25%	1543443968.0	6.0	13.0	11.0	9.0	1.28	1.0	42.3503
50%	1543737478.0	12.0	17.0	12.0	13.5	2.16	1.0	42.3519
75%	1544827509.0	18.0	28.0	12.0	22.5	2.92	1.0	42.3647
max	1545160511.0	23.0	30.0	12.0	97.5	7.86	3.0	42.3661

8 rows x 46 cols 10 per page Page 1 of 1

3.3 Categorial Variables to Numerical

In this part I convert the categorical to numerical for easily use all data in latest code. If you want to avoid the dummy variable trap (multicollinearity in regression models), you can drop the first category in each encoded column by setting 'drop_first = True'

3.4 Distribution for Numerical Data

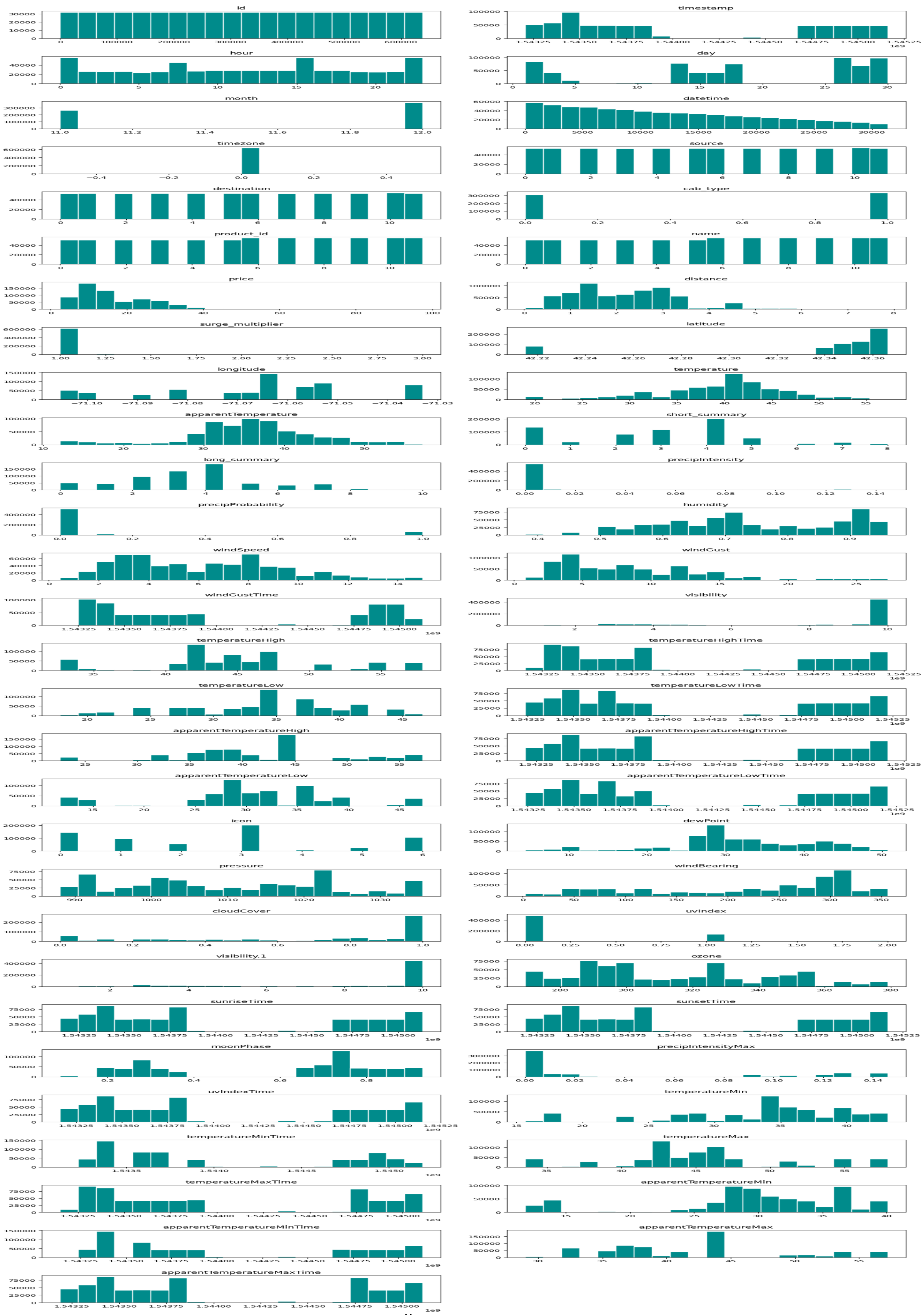

id
timestamp
hour
day
month
datetime
timezone
source
destination
cab_type
product_id
name
price
distance
surge_multiplier
latitude
longitude
temperature
apparentTemperature
short_summary
long_summary
precipIntensity
precipProbability
humidity
windSpeed
windGust
windGustTime
visibility
temperatureHigh
temperatureHighTime
temperatureLow
temperatureLowTime
apparentTemperatureHigh
apparentTemperatureHighTime
apparentTemperatureLow
apparentTemperatureLowTime
icon
dewPoint
pressure
windBearing
cloudCover
uvIndex
visibility.1
ozone
sunriseTime
sunsetTime
moonPhase
precipIntensityMax
uvIndexTime
temperatureMin
temperatureMinTime
temperatureMax
temperatureMaxTime
apparentTemperatureMin
apparentTemperatureMinTime
apparentTemperatureMax
apparentTemperatureMaxTime

Figure1: [Grid of histograms] each representing the distribution of values in one of the numeric columns from dataset

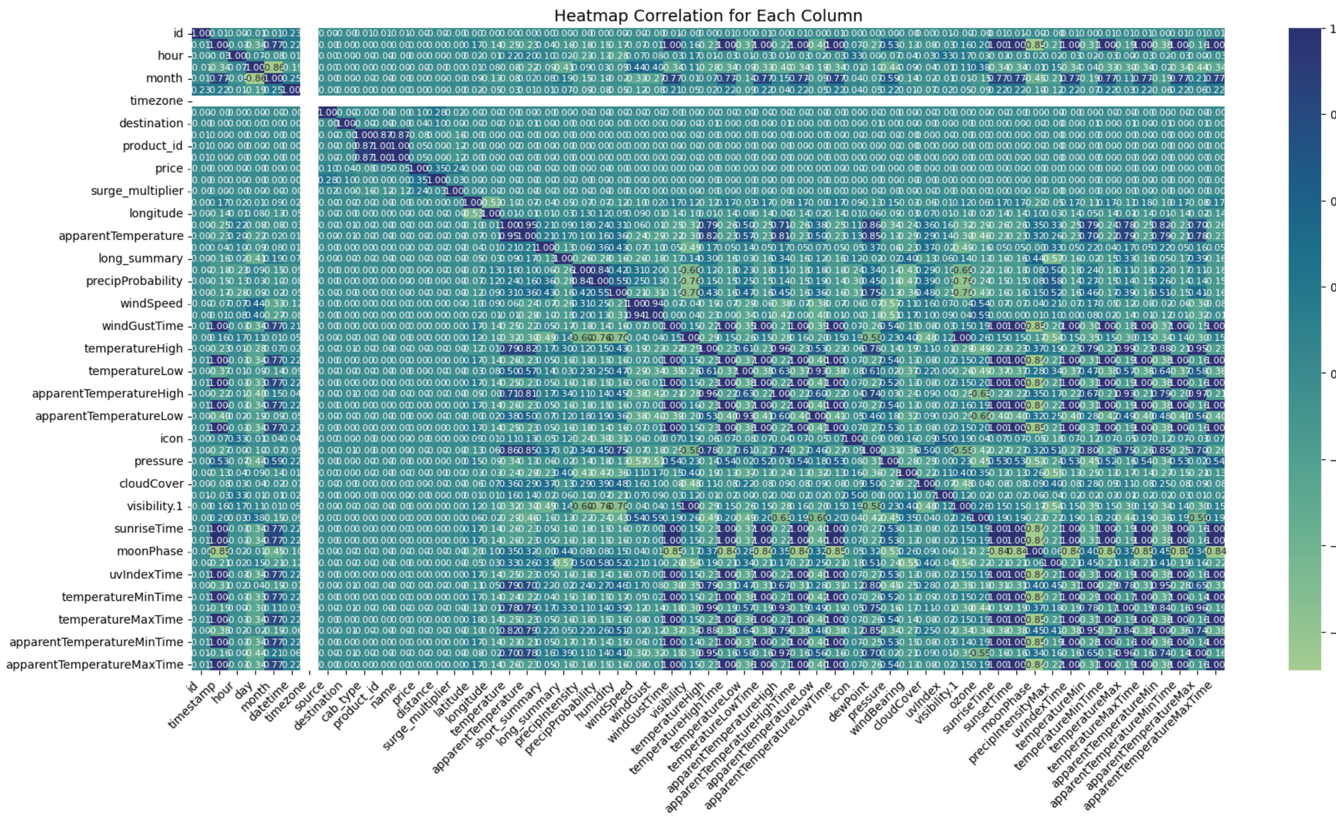


Figure2: [HeatmapCorrelation Matrix] Correlation states how the features are related to each other or the target variable.Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable) Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

3.5 Anomaly Detection with Python

Anomaly detection is the process of identifying data points that deviate significantly from the expected pattern or behavior within a dataset.

Create a dataset using the generate_data function from pyod. This function generates a synthetic dataset for training, where outliers and inliers are labeled. Convert the generated data into a Pandas DataFrame for easier handling and visualization. Add a column for the labels. Visualize the generated dataset using Seaborn’s scatter plot. The color of each point represents its label (outlier or not).

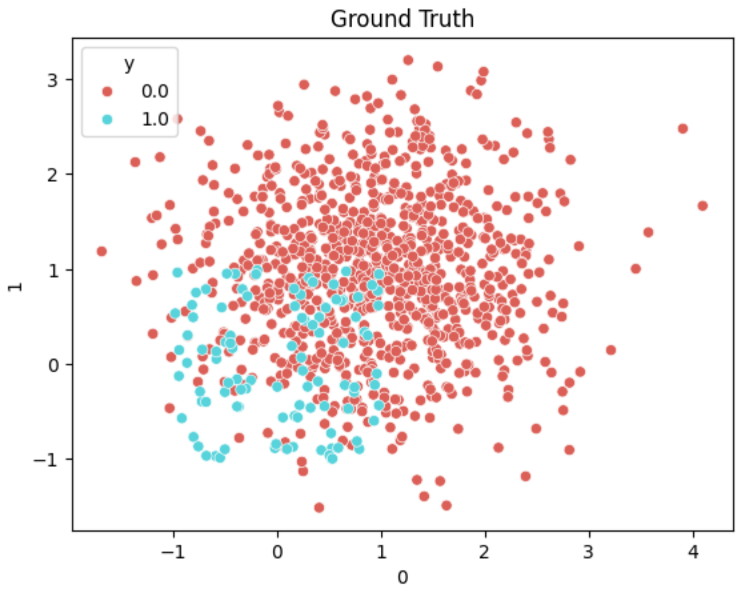


Figure3: Text(0.5, 1.0, 'Ground Truth')

Initialize a PCA model from pyod. PCA (Principal Component Analysis) is used for anomaly detection by identifying outliers based on principal components.

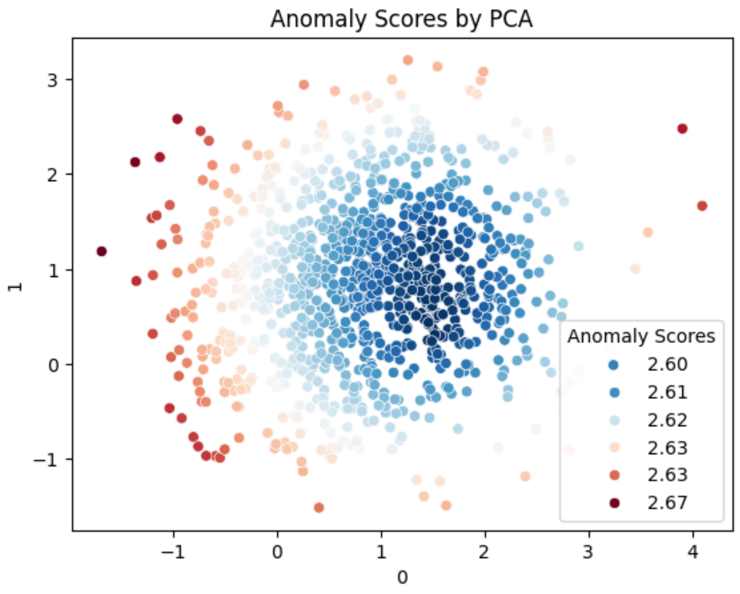


Figure4: Text(0.5, 1.0, 'Anomaly Scores by PCA')

price

3.6 Correlation Price With Each Column

```
price               1.000000
distance            0.345061
surge_multiplier    0.240458
source              0.096249
name                0.045805
product_id          0.045805
destination         0.037621
                 :
                 :
longitude          -0.001417
id                 -0.001505
moonPhase          -0.001602
long_summary       -0.001665
short_summary      -0.001892
cab_type           -0.083385
```

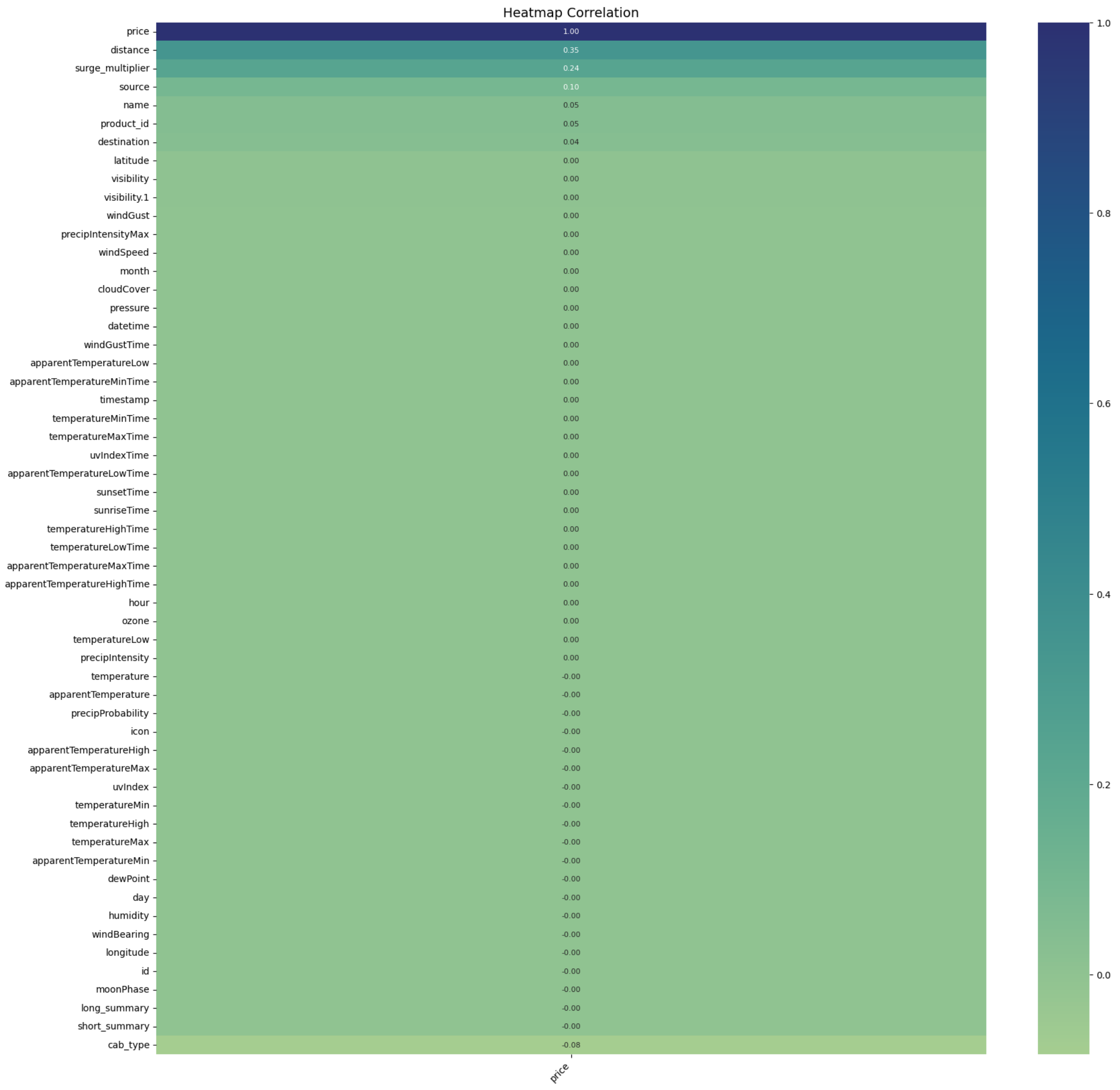


Figure5: Correlation Price With Each Column(by sorting) Only distance and surge_multiplier show meaningful linear relationships with price. Most other features are weakly or not at all correlated

3.7 Standardization

Standardization is a preprocessing technique that transforms numerical features to have a mean of 0 and a standard deviation of 1. It’s used to ensure that features with different scales

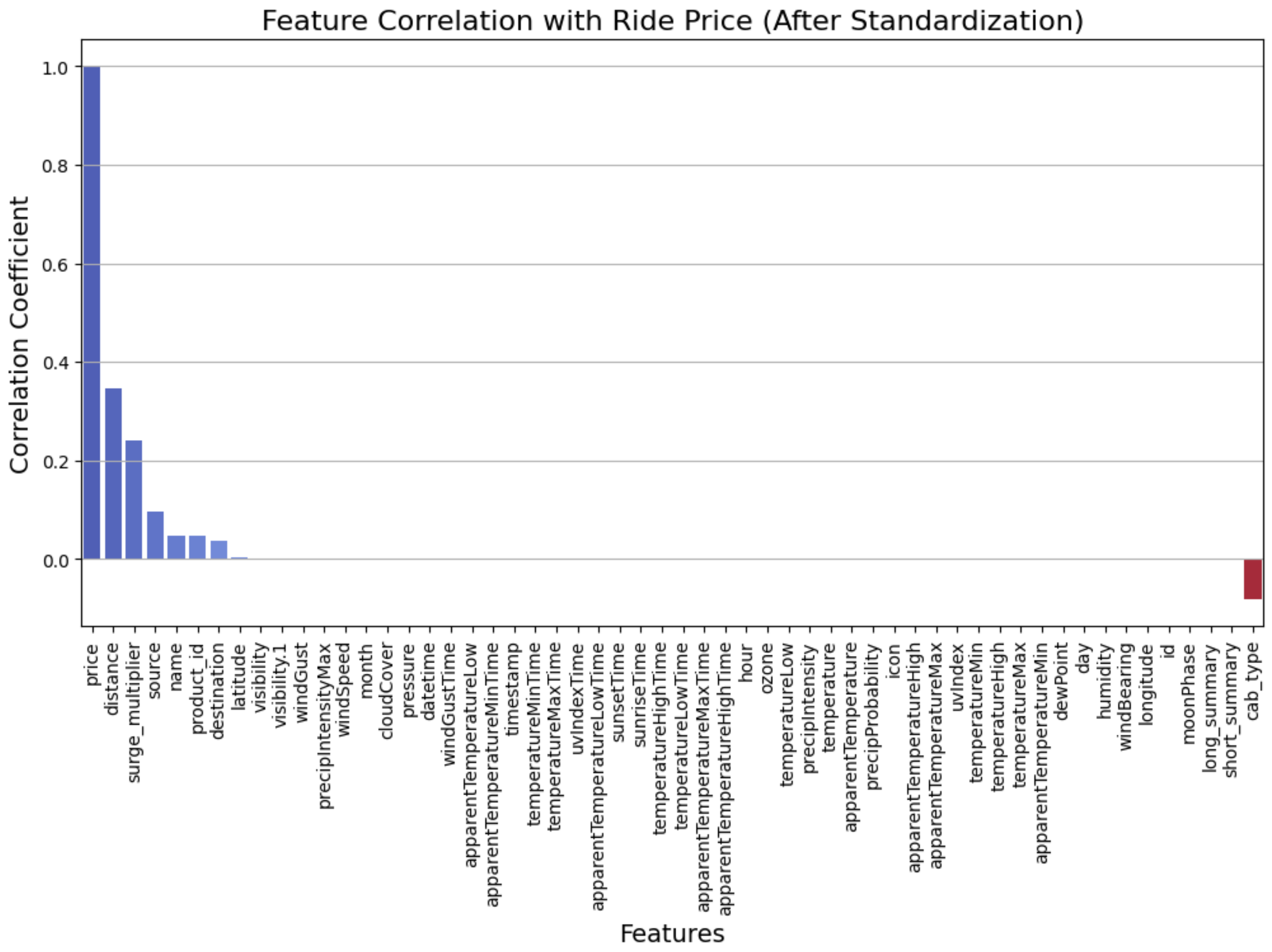


Figure6: The plot confirms that ride price is most affected by distance and surge pricing.Most weather and time-related features have minimal to no influence on price.Feature selection should prioritize distance and surge_multiplier for modeling or interpretation.

4 Baseline Linear Regression

4.1 MAE,MSE, , R²

A baseline linear regression model was trained and evaluated on ride pricing data, showing performance metrics (MAE, MSE, R²), visualized prediction accuracy and residuals, and identified key features impacting price via coefficient analysis. After having our model fitted, we can get the results to check whether the model works satisfactorily and to interpret it.

Mean Absolute Error: 6.931000416823012

Mean Squared Error: 66.91942535017813

Coefficient of Determination (R2): 0.2274588429085248

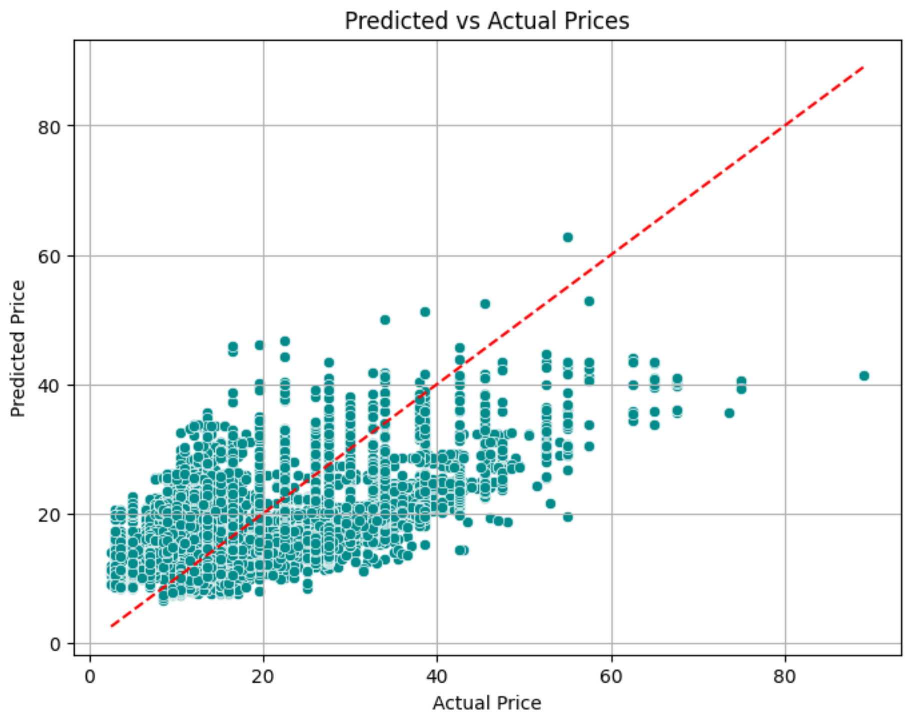


Figure7: This scatter plot compares predicted ride prices to actual prices from the linear regression model. The red dashed line represents the ideal case where predicted values equal actual values. Most points fall below this line, indicating that the model tends to underpredict ride prices, especially for higher-priced rides, revealing some degree of prediction bias or underfitting

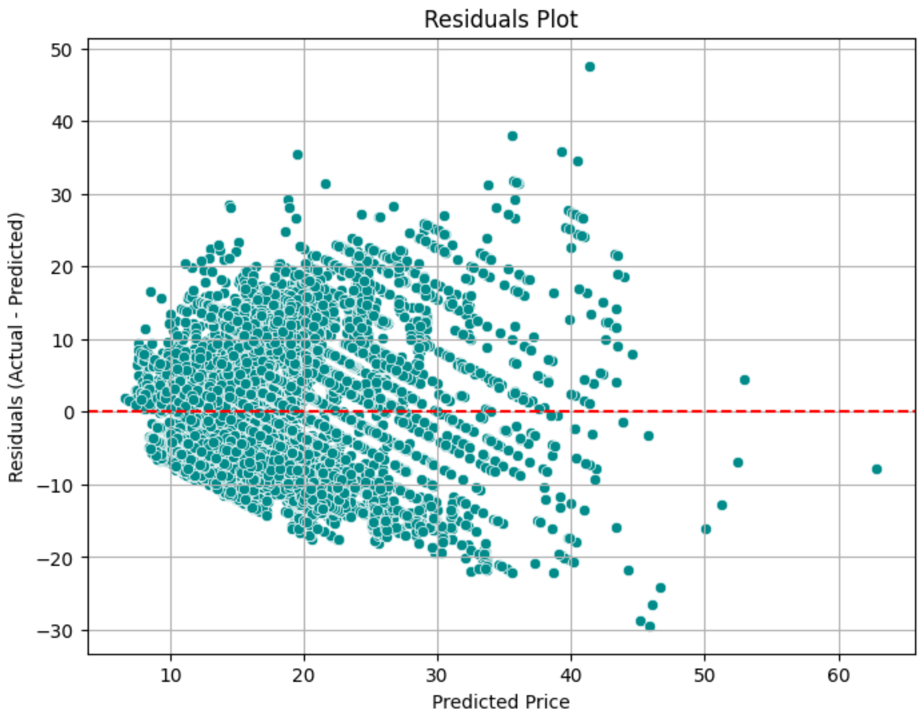


Figure8: This residuals plot shows the difference between actual and predicted ride prices against predicted prices; the pattern reveals increasing residual spread with higher predicted values, indicating heteroscedasticity and suggesting the linear regression model does not capture variance in price well across all ranges

4.2 impact on ride price based on the coefficients

surge pricing, month, and cab type appear to be the most influential factors in determining ride prices, with geographic features like latitude and longitude playing a smaller but still significant role. Weather factors, such as humidity, seem to have a modest negative impact on price.

Top 10 features with the most significant impact on ride price:

	Feature	Coefficient
13	surge_multiplier	20.754743
4	month	12.345591
9	cab_type	-8.428467
12	distance	2.803510
15	longitude	1.145469
22	humidity	-0.796864
14	latitude	0.701294
11	name	0.625509
10	product_id	0.625509
3	day	0.407807

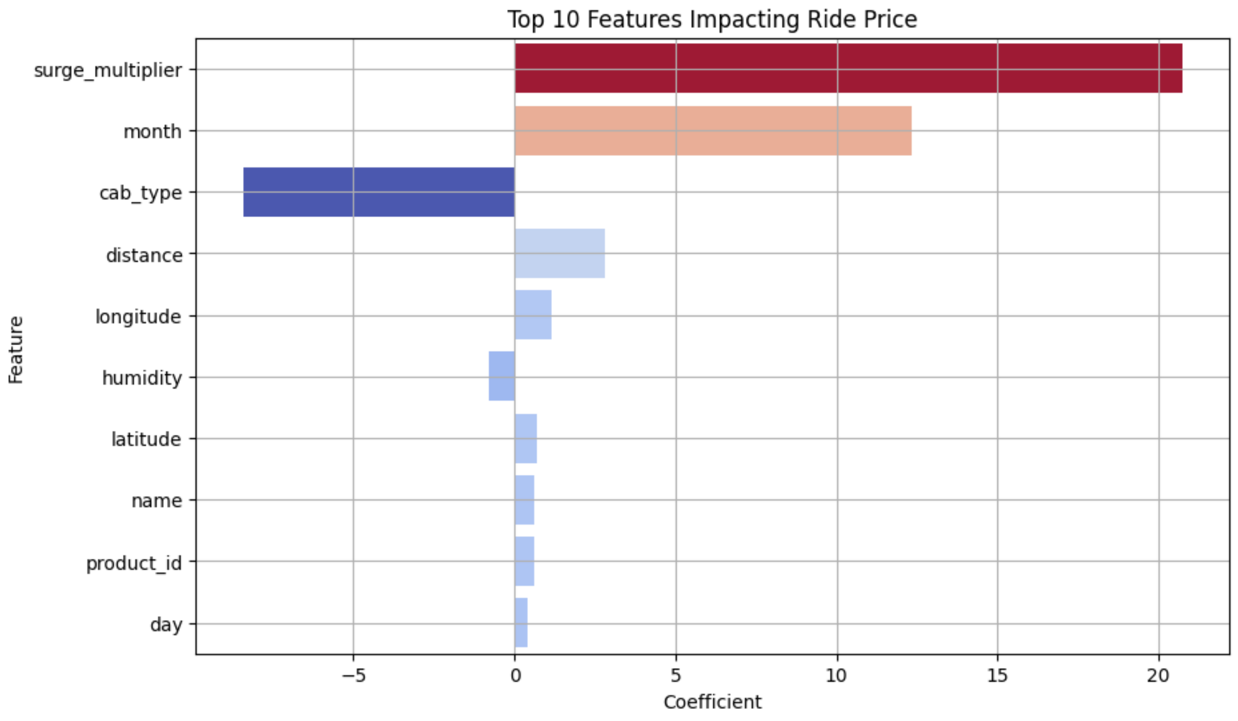


Figure8: This appearance of the plot would be a mix of positive and negative bars, with some features having stronger impacts (like surge multiplier and month) and others having weaker impacts (like humidity and day). Negative bars would typically appear below the X-axis, while positive bars appear above it.This plot helps visually communicate which features have the largest and smallest effects on ride price, making it easy to see which factors are most influential.

5 Model Variants and Nonlinear Extensions

5.1 Standard Scaler

At first StandardScaler was used to standardize the features. This means that the features in both the training (X_train) and test sets (X_test) were transformed to have zero mean and unit variance, which is important for regularized models like Ridge and Lasso. This ensures that the models don't give undue importance to features with larger scales. Secondly , RidgeCV was applied to perform Ridge regression with cross-validation. The hyperparameter alpha (which controls the strength of regularization) was tuned by testing 10 values logarithmically spaced between 0.01 and 100. LassoCV was applied similarly to Ridge regression, but here, the Lasso model applies L1 regularization. The regularization strength (alpha) was also tuned over the same range of values. Both Ridge and Lasso regression showed similar results, with low R2 scores (0.23) on both the training and test sets, indicating that the models do not perform well in explaining the variability of the target variable.

The MAE and MSE were also close for both models, indicating their similar performance.

Ridge Best Alpha: 100.0

Ridge Train R2 Score: 0.23

Ridge Test R2 Score: 0.23

Ridge MAE: 6.93

Ridge MSE: 66.92

Lasso Best Alpha: 0.01

Lasso Train R2 Score: 0.23

Lasso Test R2 Score: 0.23

Lasso MAE: 6.92

Lasso MSE: 66.91

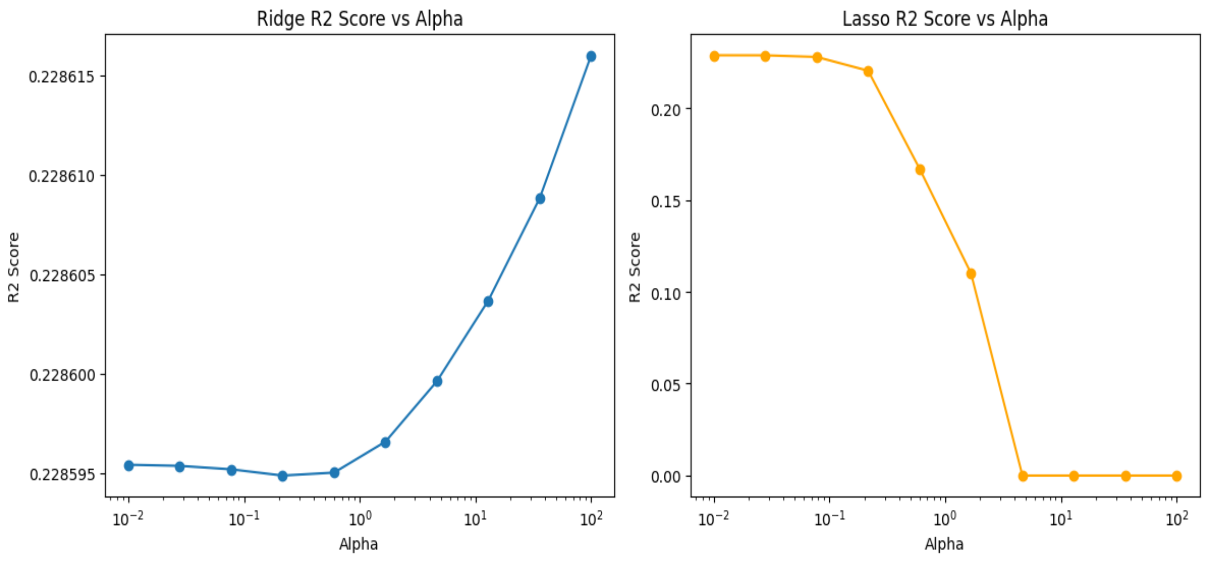


Figure9: The plots help visualize the effect of regularization strength (alpha) on model performance for both Ridge and Lasso, but given the low R2 scores, it suggests that the models may not be well-suited for this particular dataset or there may be other factors affecting performance. Ridge R2 Score vs Alpha: This plot shows how the R2 score changes as the alpha value varies for the Ridge regression model. It helps visualize the relationship between the strength of regularization and the model's performance. Lasso R2 Score vs Alpha: Similarly, this plot visualizes how the L2 regularization strength (alpha) affects the Lasso model's R2 score. Both plots use a logarithmic scale for alpha to better capture the variations in the R2 scores across different alpha values.

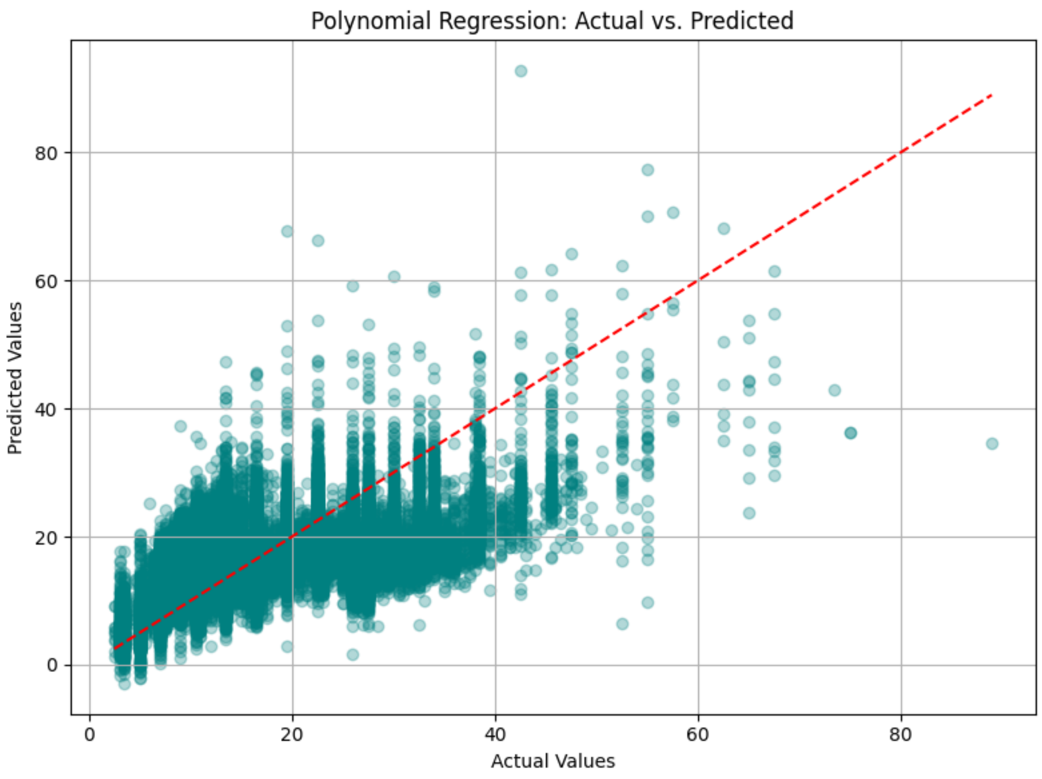
5.2 Polynomial Regression

The polynomial regression (degree 2) with linear regression does not seem to be a highly effective model for this dataset. The relatively low R2 score and moderate MSE suggest that the underlying relationships may be too complex for a simple second-degree polynomial to capture.

If the dataset contains more intricate nonlinear patterns, you might consider exploring higher-degree polynomials, or even other nonlinear models such as decision trees, random forests, or gradient boosting models.

Polynomial R2 Score: 0.25580214171825766

Polynomial MSE: 64.55935931094976



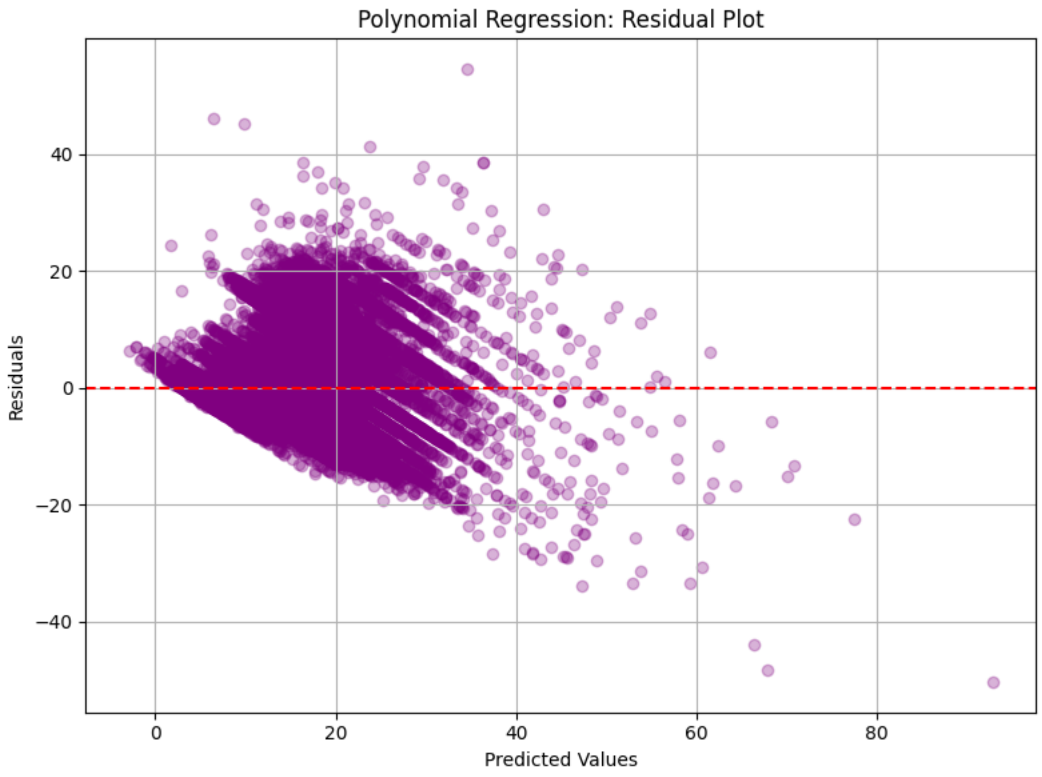


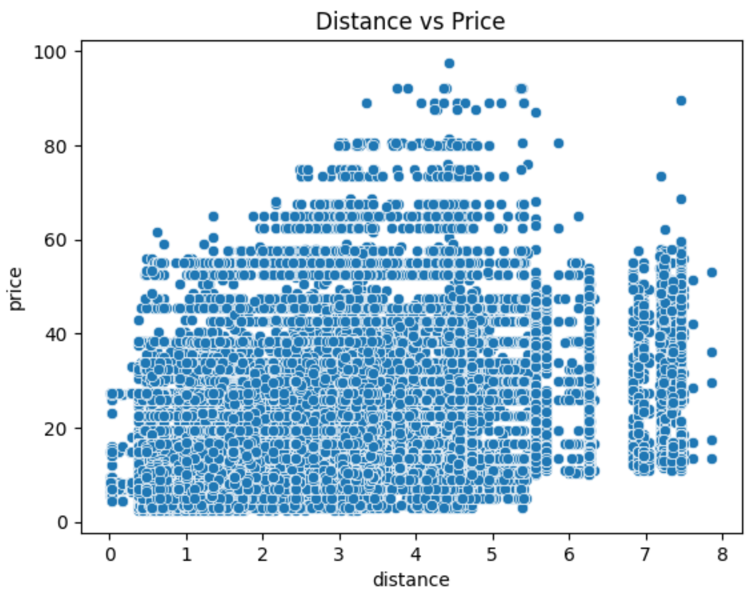
Figure10: Actual vs. Predicted Plot: This helps visually assess how well the model is predicting the target values. Significant deviation from the red dashed line indicates that the model's predictions are not highly accurate.Residual Plot: This helps evaluate if the model's errors (residuals) are randomly distributed. If they are not, it suggests that the model might be missing important patterns, and further refinement may be needed.

6 Model Enhancement through Data Preparation

6.1

The correlation matrix shows that distance and surge_multiplier have the strongest positive correlations with price, suggesting they have a moderate but noticeable impact on ride pricing.

```
price               1.000000
distance            0.345061
surge_multiplier    0.240458
source              0.096249
name                0.045805
                :
                :
moonPhase          -0.001602
long_summary       -0.001665
short_summary      -0.001892
cab_type           -0.083385
Name: price, dtype: float64
```



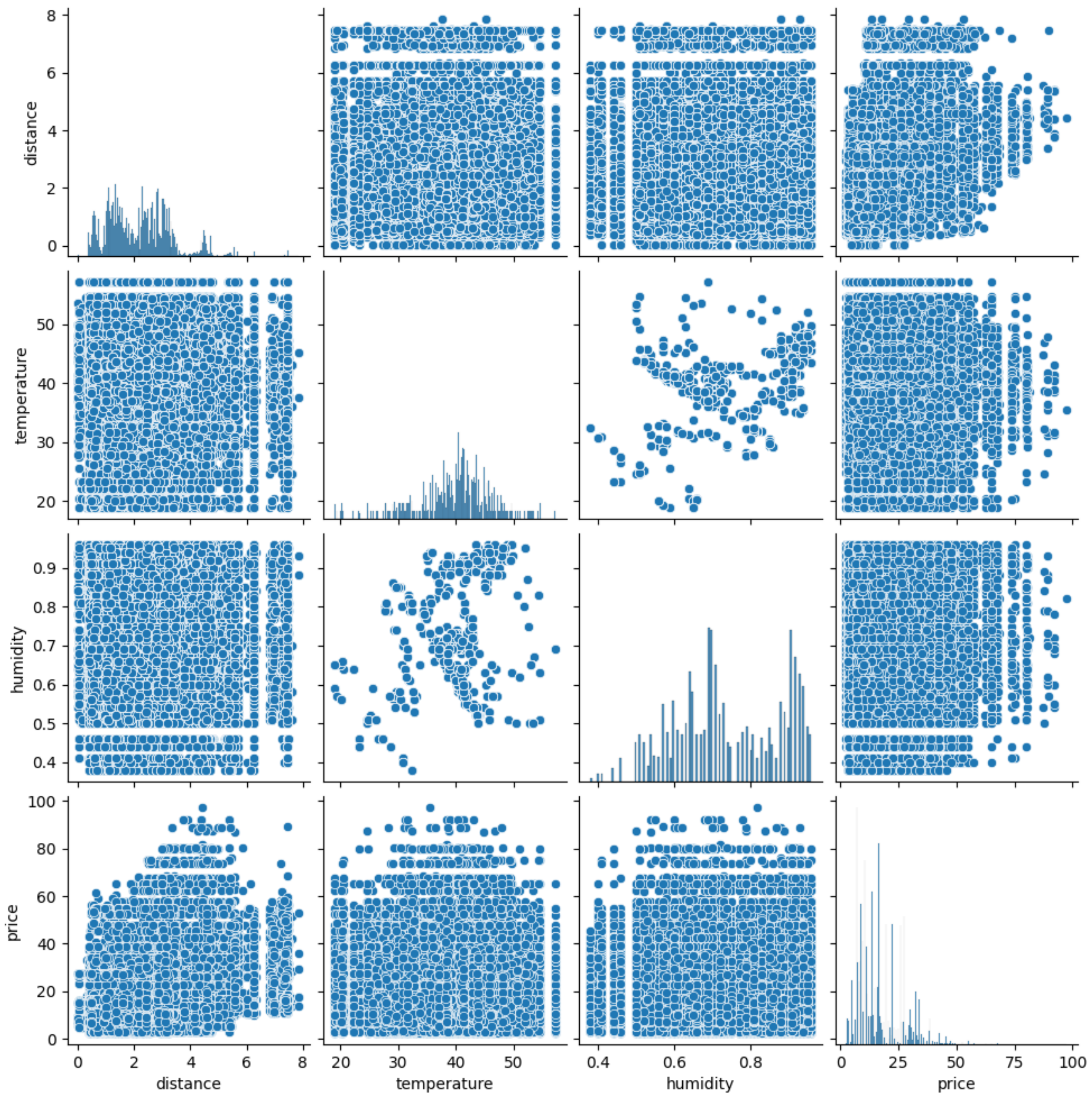


Figure11: The scatter plot between distance and price visually confirms the positive relationship between these two features.The pairplot provides a broader view of how distance, temperature, humidity, and price interact. It helps you understand potential pairwise relationships and distributions, guiding you toward important features for modeling.

6.2 Binning or encoding categorical values differently

The simplest use of qcut is to define the number of quantiles and let pandas figure out how to divide up the data. In the example below, we tell pandas to create 4 equal sized groupings of the data.keep in mind the values for the 25%, 50% and 75% percentiles as we look at using qcut directly.The simplest use of qcut is to define the number of quantiles and let pandas figure out how to divide up the data. In the example below, we tell pandas to create 4 equal sized groupings of the data.

```
Length: 637976, dtype: float64
```

6.3 Identifying and filtering noisy or low-quality samples

4 different techniques to clean data is:

Method	Target	Purpose
Z-Score	Features (scaled)	Remove statistical outliers
IQR	Features (raw)	Remove extreme values (non-normal)
Percentile Clipping	Target y	Trim outliers in the label
Residual Filtering	Whole sample	Keep only model-friendly rows

The result is:

```
result of the answer is 484861 samples
result of the answer is 55 features
```

7 Bonus (Optional)

It evaluates the performance of a Ridge regression model by hour and day. The steps are as follows:

Datetime Conversion: The datetime column is converted to a proper datetime format to enable feature extraction based on time.

Skewed Feature Transformation: A function is applied to handle skewed numerical features. It uses Box-Cox transformation or log transformation to normalize features with skewed distributions, replacing infinities and missing values with appropriate values to avoid errors during modeling.

Model Evaluation by Group: The code evaluates the model performance for different groups based on the hour of the day and the day of the week. For each group, the features are preprocessed and scaled, and a Ridge regression model is trained. The performance is measured using R^2 and Mean Squared Error (MSE).

Visualization of Results: Performance metrics (R^2 and MSE) are plotted for each hour of the day and each day of the week to examine how the model's accuracy varies over time.This approach helps assess if the model performs differently depending on the time of day or week, which can be useful for understanding temporal patterns in the data.

```
   day        r2        mse
0    1  0.281029  62.888712
1    2  0.275676  63.366344
2    3  0.276286  61.821613
3    4  0.287218  63.171142
```

```
4    9  0.285210  61.150225
              :
              :
19   19  0.282709  61.693093
20   20  0.279483  63.747732
21   21  0.282215  62.860253
22   22  0.278911  62.941499
23   23  0.275058  62.399169
```

<u>Day-wise Plot:</u>

The R² scores are relatively consistent, ranging from approximately 0.26 to 0.29. The highest R² score is seen on day 4 (0.2872), and the lowest on day 10 (0.2628).The MSE values show a similar pattern, with values ranging from around 61.15 to 64.08. The lowest MSE occurs on day 9 (61.15), and the highest MSE is observed on day 18 (64.08).This suggests that, while performance slightly varies by day, the model's predictive accuracy does not drastically fluctuate across days.

<u>Hour-wise Plot:</u>The R² scores by hour show minor fluctuations, with values ranging from 0.267 to 0.285. The highest R² score occurs at hour 0 (0.2853), and the lowest at hour 7 (0.2675).The MSE values are mostly between 61.68 and 64.08, with the lowest occurring at hour 10 (61.68) and the highest at hour 11 (63.98).The performance by hour is more variable compared to day-wise evaluation, indicating that time of day may have a more noticeable effect on model performance.

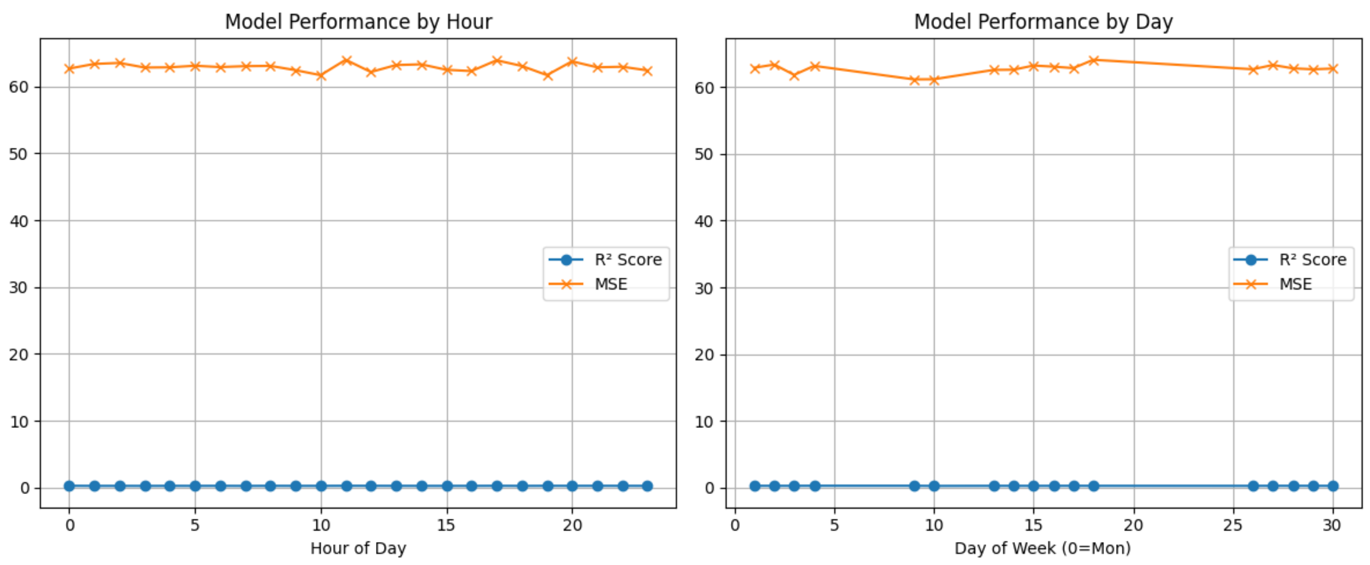


Figure12: The plots display the R² score and Mean Squared Error (MSE) of a Ridge regression model evaluated by day of the week and hour of the day. the model's performance shows some fluctuations depending on the time of day and day of the week, but the differences are not drastic, suggesting relatively stable predictive power over time.

8 Conclusion

This analysis aimed to evaluate and improve the predictive performance of a Ridge regression model by performing several transformations and evaluations on the dataset. Below are the key findings and conclusions drawn from the analysis. The Ridge regression model demonstrated consistent performance, with slight fluctuations observed in the results based on time-of-day and day-of-week. While these variations were present, they were not drastic, indicating that the model's performance was relatively stable.The transformations applied to handle skewness and outliers, along with the feature engineering steps, were effective in improving model accuracy.Future improvements could include exploring other regression techniques, such as Lasso Regression, or experimenting with non-linear models like decision trees or gradient boosting, to further improve predictive power.

In conclusion, this analysis successfully transformed raw data, removed noise, and applied appropriate modeling techniques to predict ride prices, with consistent model performance observed across different time-based groupings.

References

[1] BM, "Uber and Lyft Dataset Boston, MA" 2018.
https://www.kaggle.com/datasets/brllrb/uber-and-lyft-dataset-boston-ma