UNIVERSITY OF
CAMBRIDGE

Predicting the Number of Hourly Bicycle Rentals for Capital Bikeshare in Washington D.C.

MPhil in Economics and Data Science

Modules:

D100 - Fundamentals of Data Science;
D400 - Research Computing

Candidate Number: 3381A

Deadline Date: 19 December 2024

Declaration:

I confirm that this is entirely my own work and has not previously been submitted for assessment. I have read and understood the University's and Faculty's definition of Plagiarism.
(`https://www.plagiarism.admin.cam.ac.uk/definition`)

Actual Word Count: 1993

1 Introduction

This project aims to predict hourly demand for rental bikes from Capital Bikes in Washington D.C. A better understanding of the drivers of bike rental demand allow rental-service providers to make more informed decisions on bike maintenance and placement - efficient resource allocation and increased user satisfaction. This is achieved by: (i) identifying the most influential features driving bike rentals to highlight patterns in bike usage; and (ii) accurately predicting bike usage given historical and contextual information.

The target variable, cnt represents the total number of bikes rented in a given 1 hour period in Washington D.C from Capital Bikeshare from Fanaee-T (2013). I use time-related and environmental features to predict the number of bike rental.

I use GLM (Generalised Linear Model) and LGBM (Gradient Boosting Machine) models, with a Tweedie distribution objective function for this analysis, given the data is both zero-inflated and positively-skewed. The LGBM also effectively captures non-linearity and feature interactions.

2 EDA

2.1 Cleaning Steps

- Confirmed: no missing values; all values are unique

- Extracted day (of the month) from $dteday$; then dropped $dteday$

- Revalued yr to 2011 (if equal to 0) and 2012 (if equal to 1). Easier interpretation and generalisation to other years

- Denormalised values for hum and $windspeed$ by multiplying by the max values provided

- Denormalised values for $temp$ and $atemp$ using the following formula:

$$t_{\text{denormalized}} = t_{\text{normalized}} \cdot (t_{\max} - t_{\min}) + t_{\min} \tag{1}$$

- Corroborated: $cnt = casual + registered$; dropped $casual$ and $registered$

- Dropped hum

- Dropped day and $holiday$

2.2 Explanatory Data Analysis and Feature Selection

2.2.1 Target Variable

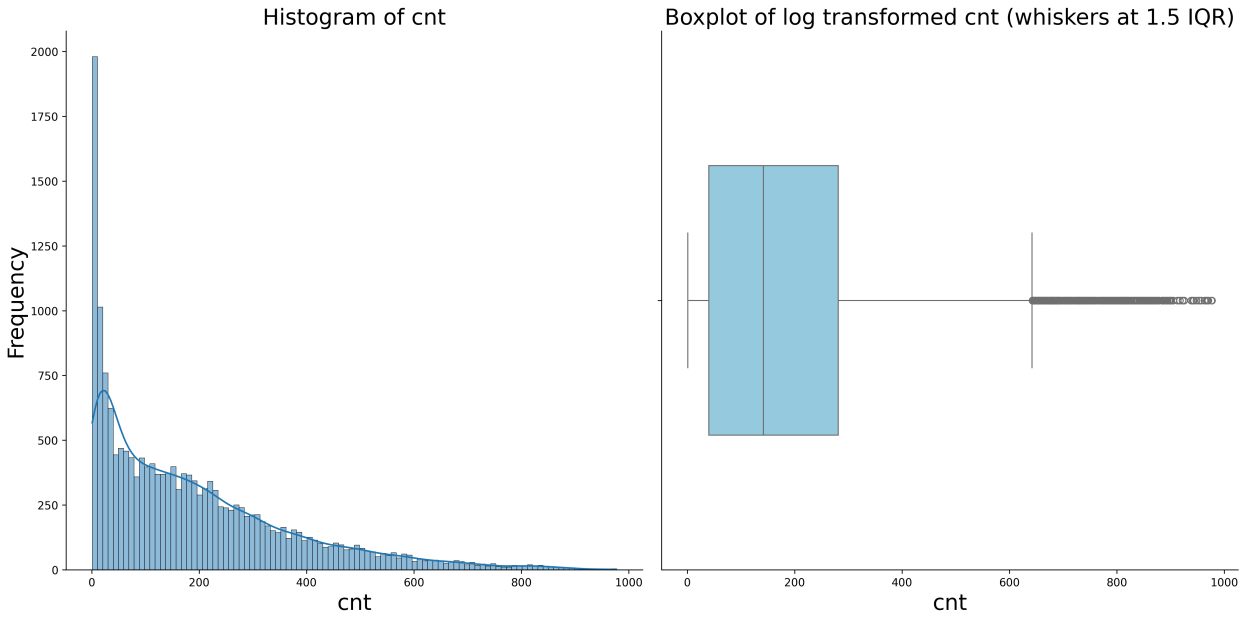

Figure 1: Side-by-side Histogram and Boxplot for cnt

The target variable, cnt, is the count of total rental bikes. It follows a distribution (Figure 1) similar to the Tweedie - zero-inflated with a long positive-skewed tail, and no negative outcomes. However, cnt has many outliers to the right which could affect the model; I tried log transforming it to address this.

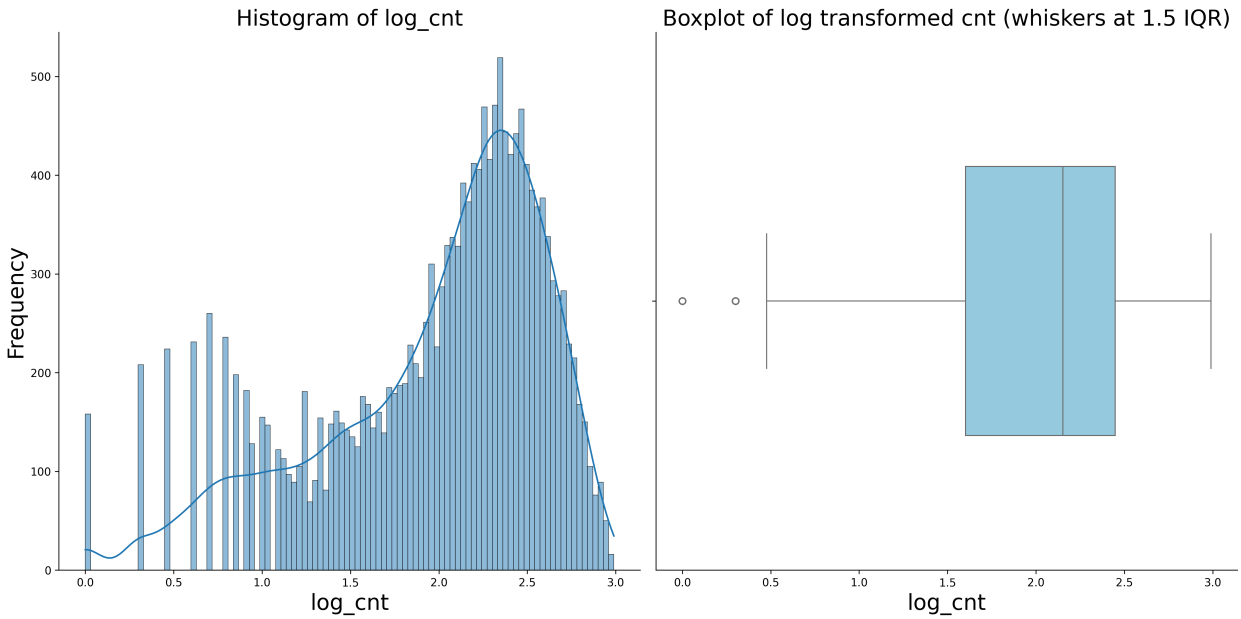

Figure 2: Side-by-side Histogram and Boxplot for $log(cnt)$

While outliers are less, $log(cnt)$ has a distorted distribution - common with zero-inflated variables (Figure 2), since the function applied is $log(x+1)$ to produce a finite output. This poses to be a bigger problem than outliers when using the GLM and LGBM models which require a specified objective function. Therefore, I opt against the transformation.

I decided against winsorising because: (i) GLM uses a Tweedie link function which, by definition, takes outliers into account; and (ii) LGBM is a tree-based method, which tend to be more robust to outliers.

2.2.2 Other Features: Cleaning

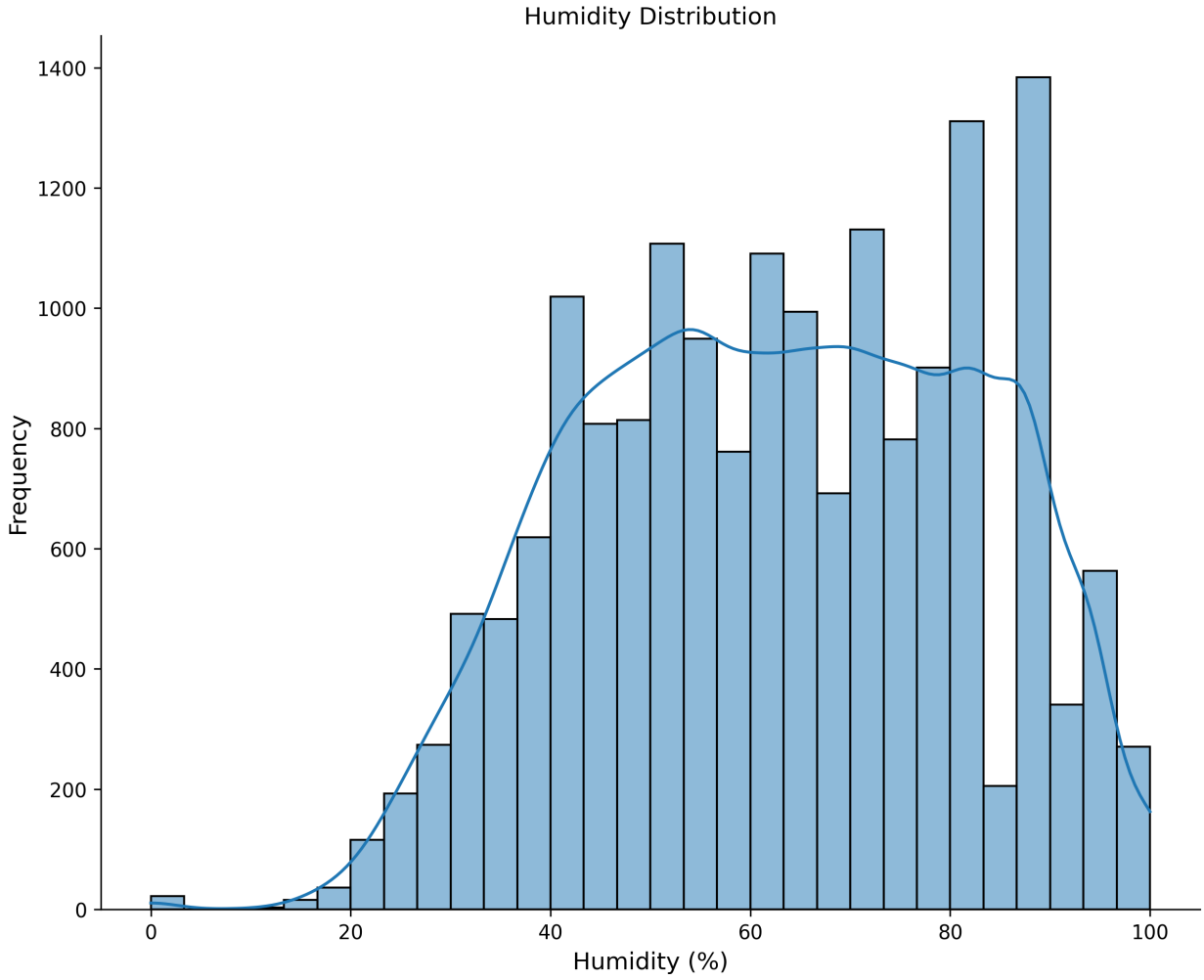

Figure 3: Histogram for hum

Figure 3 shows the 'denormalised' distribution of hum (humidity), which is scientifically impossible; low humidity values around 0% are not seen in Washington D.C.. This could be due to: (i) a problem in the data collection process; or (ii) a problem with the denormalisation process. I used the same denormalisation method for $windspeed$ which falls within the expected ranges, but hum could have used a different, unspecified (even in Fanaee-T and Gama (2013), which analyses the same dataset) normalisation method. Due to the lack of information, I drop hum.

2.2.3 Other Features: Selection

I only used manual feature selection, removing features with little variance, or covariance with cnt. I also implemented Elastic Net regularisation in my GLM model, where the tuned $l1\_ratio = 1.0$ (i.e. strictly Ridge), so there is no automatic feature selection.

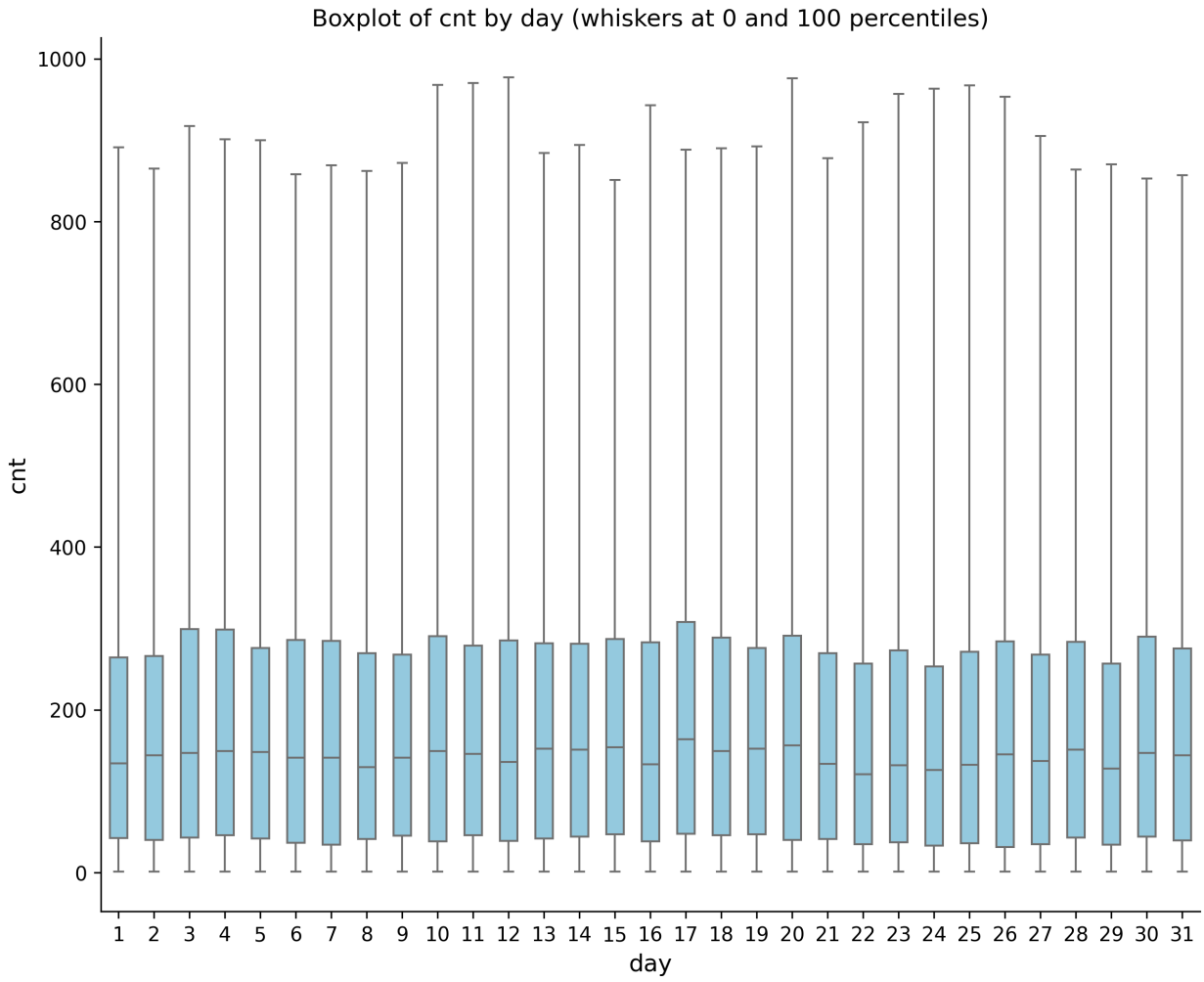

Figure 4: Boxplot of cnt by day

I created day from $dteday$ to analyse within-month variation in cnt. However, Figure 4 shows a similar distribution - first and third quartiles, mean, minimum and maximum values - between all days in the month. I drop days because of its low variance and expected explanatory power.

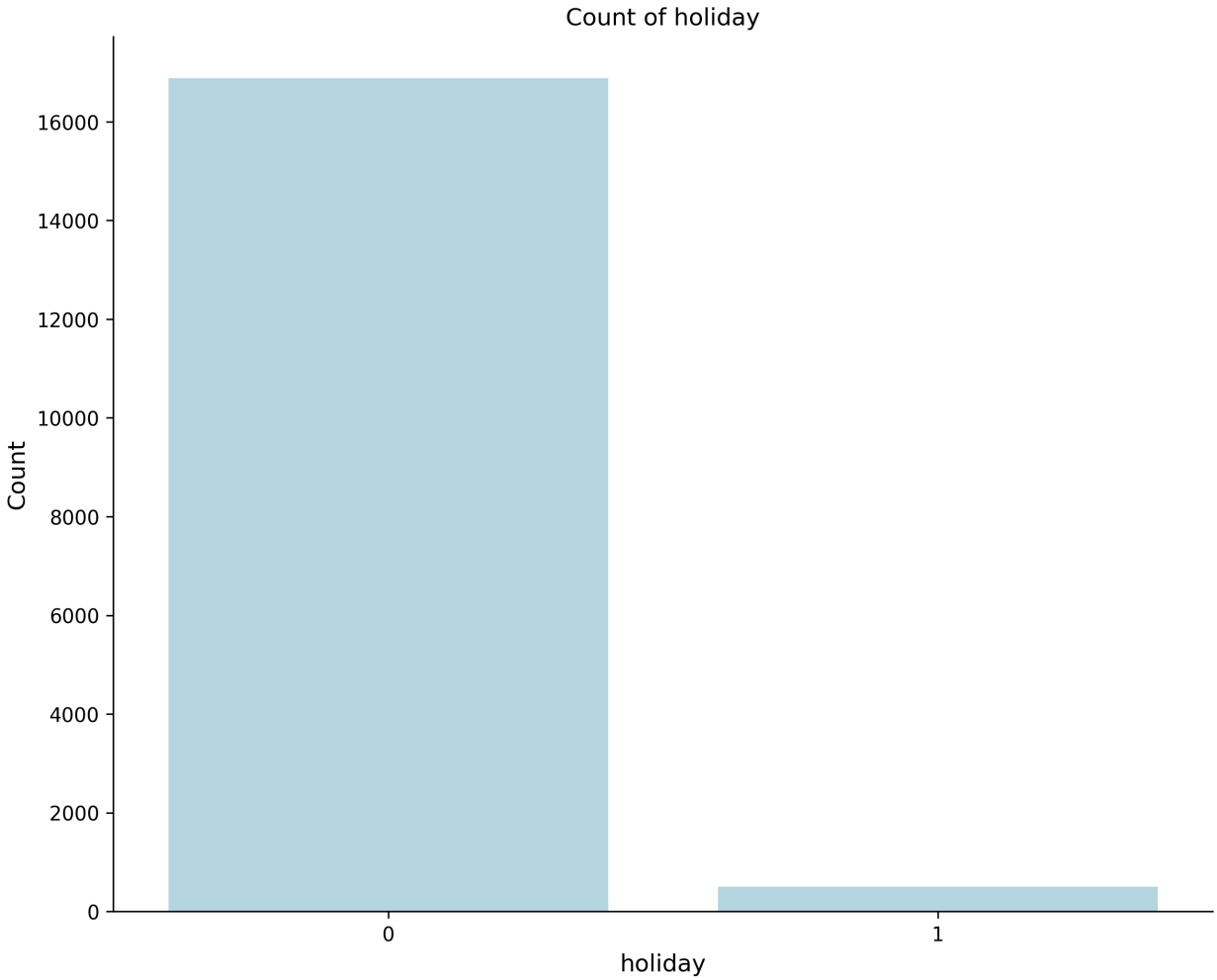

Figure 5: Count plot for $holiday$

I drop $holiday$ because it has very few values for which it is not 0 (see Figure 5), suggesting a low covariance with cnt. Additionally, any effect $holiday$ could have on cnt is likely captured by $workingday$ ($workingday = 0$ when $holiday = 1$).

2.2.4 Other Features: Hypothesis

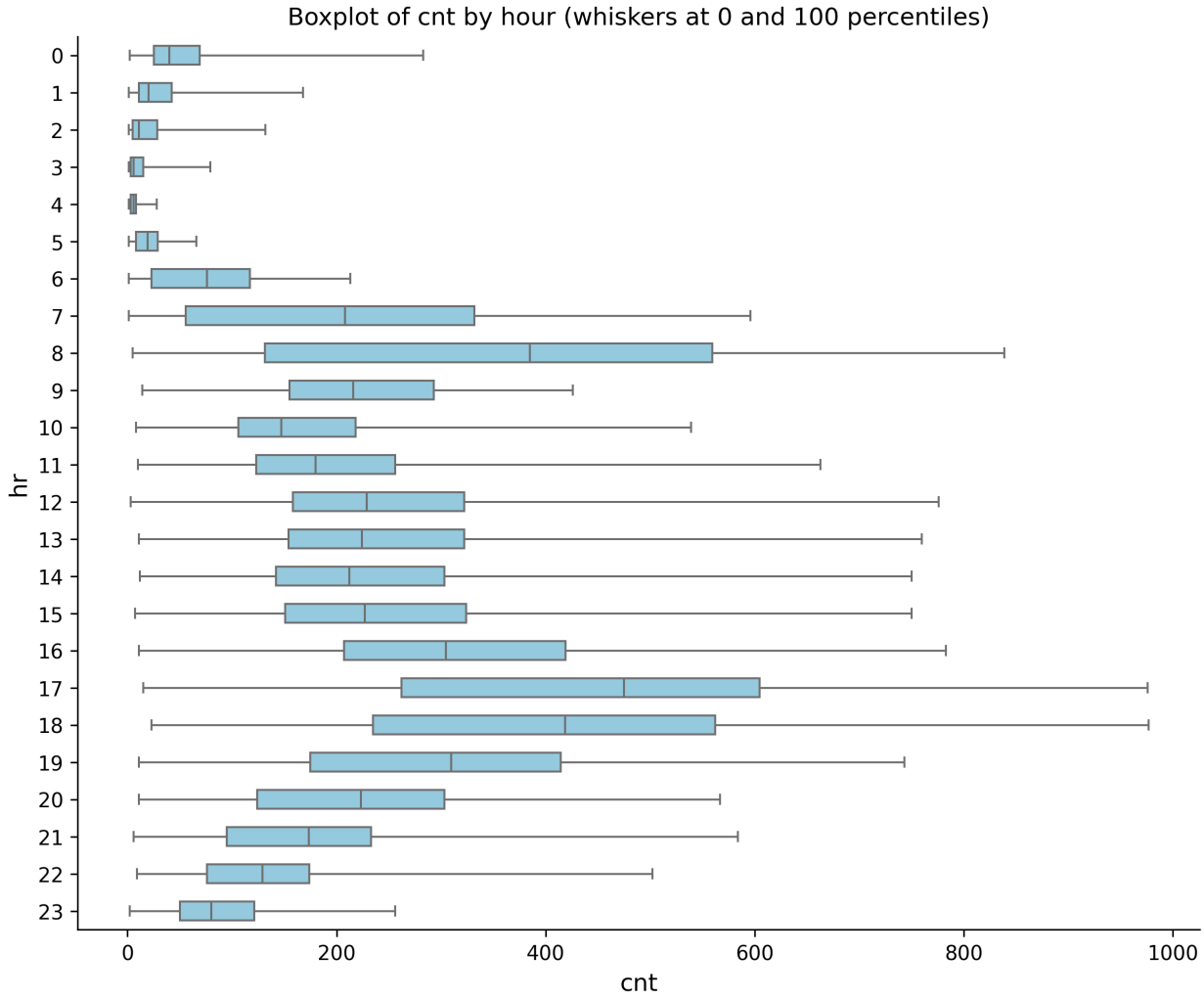

Figure 6: Boxplot of cnt for each hr

Figure 6 highlights an important feature in the distribution of cnt. There are spikes in the values of cnt at 8am, 5pm, and 6pm respectively, which could indicate a tendency to: (i) cycle to and from work; and (ii) use bikesharing systems to do so.

2.3 Summary

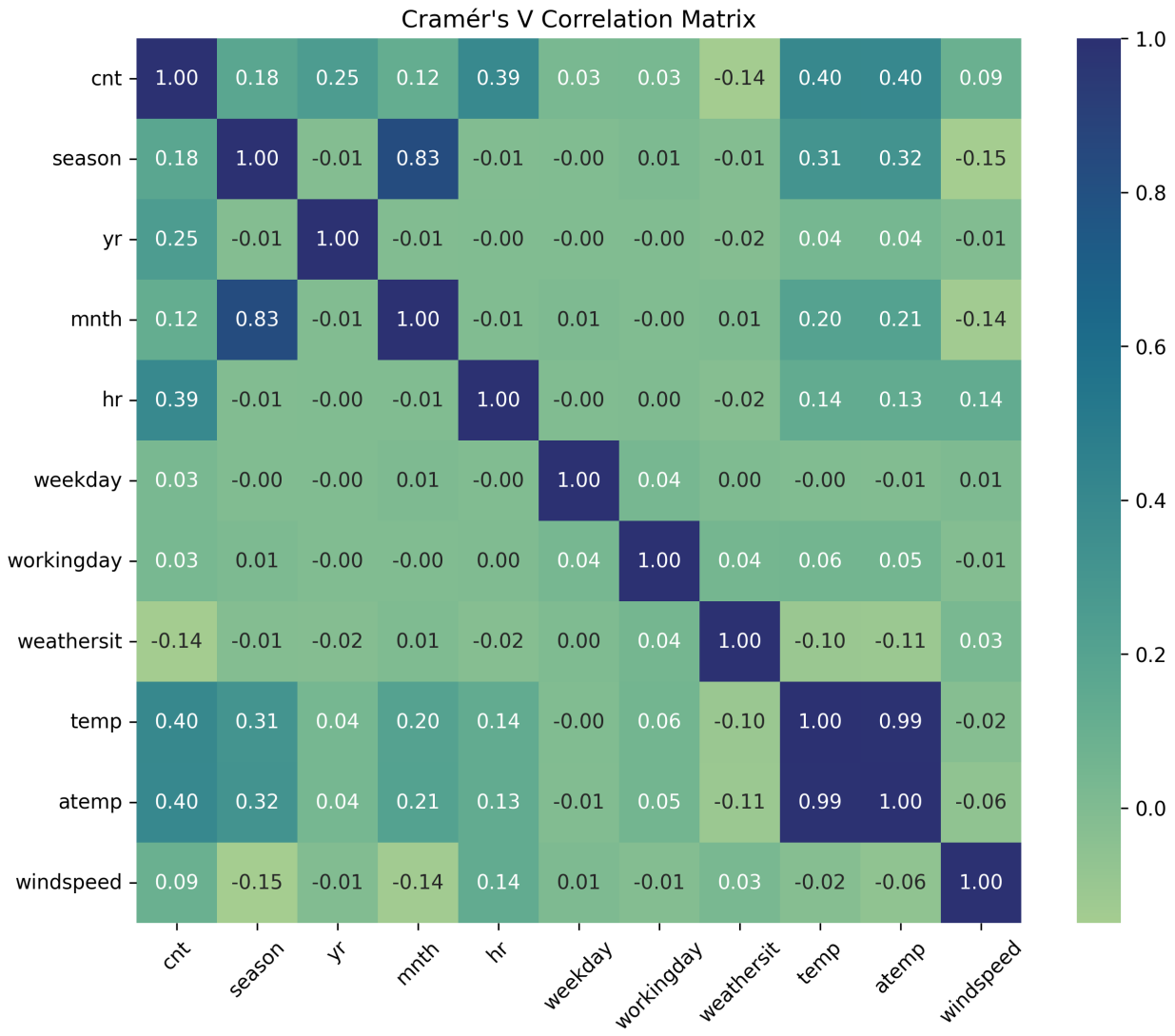

Figure 7: Cramer's V Correlation Matrix

Figure 7 displays the correlation between all the features in the cleaned dataset (uses Cramer's V value for categoricals). Post cleaning, all explanatory variables have non-zero correlations with cnt; hr, $temp$, and $atemp$ specifically have high correlation with cnt, explaining why they have higher feature importance in the evaluation.

3 Feature Engineering

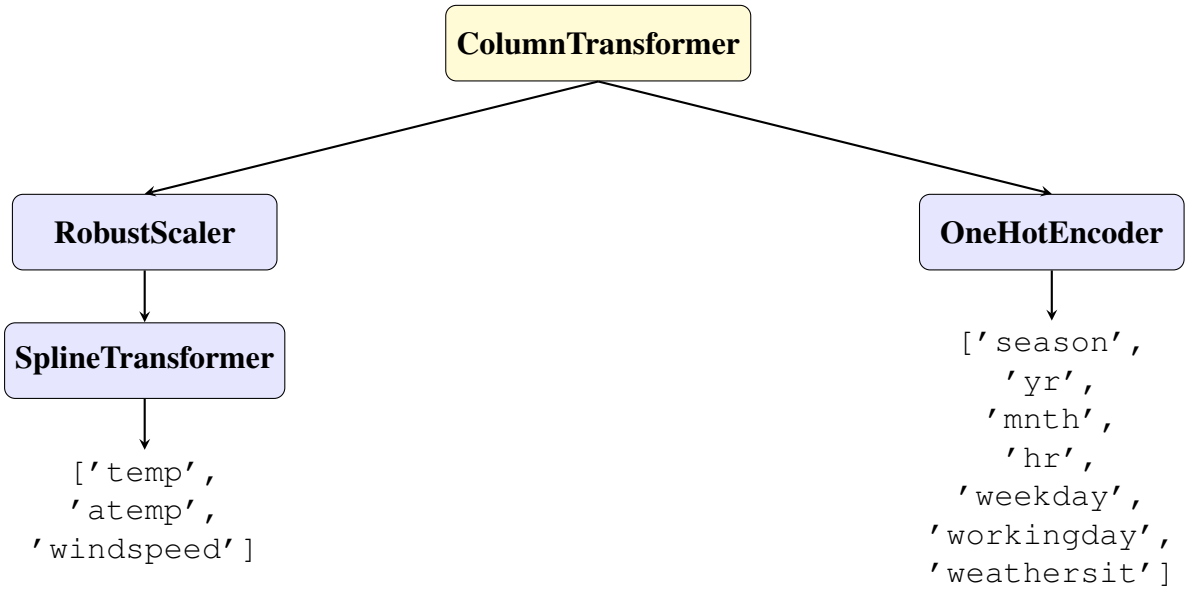

Figure 8: Visualization of Full Preprocessing Pipeline: ColumnTransformer, RobustScaler, SplineTransformer, and OneHotEncoder

Figure 8 is the pipeline I use to preprocess my data before feeding it to the model. First, I split the sample into test and train datasets, using a train ratio of 0.8. I also stratified the split to ensure a consistent distribution of cnt between test and train sets; it also avoids misleading evaluation metrics due to imbalanced train/test sets.

I use separate preprocessing pipelines for numerical and categorical columns. I one-hot-encoded all my categorical variables, creating a dummy variable for each observed value.

Numerical columns were first scaled using $RobustScaler$, which scales the observations according to the mean and standard deviation of observations between the 1st and 99th percentile, to decrease the effect of outliers. Scaling also ensures that no feature wrongly dominated the objective function just because of a larger absolute variance. Next, they are split into splines, with knots at each quintile. Splines, like binning, prevent overfitting, but splines additionally allow for non-linear relationships. Creating knots at quintiles, rather than uniformly, ensures a more even split of observations between each spline.

The GLM and LGBM models are each trained on the preprocessed training datasets. Both models use a Tweedie loss with a power of 1.5, which effectively addresses the zero-inflation, continuity, and positive skewness of cnt.

4 Final Models and Hyperparameter Tuning

Model	Hyperparameter	Range	Optimal Value
GLM	α	$0.001, 0.01, 0.1, 1, 10, 100, 1000$	0.01
	$l1\_ratio$	$0, 0.25, 0.5, 0.75, 1$	1.0
LGBM	$learning\_rate$	$0.01, 0.02, 0.03, 0.04, 0.05, 0.1$	0.1
	$n\_estimators$	$50, 100, 150, 200$	200
	num_leaves	$31, 63, 127, 255$	31
	min_child_weight	$1, 2, 5, 10$	1

Table 1: Hyperparameter Tuning for GLM and LGBM Models

Table 1 displays the hyperparameter grid over which I tuned the GLM and LGBM models respectively, and their optimal values. Starting with the GLM, $alpha$ represents the regularisation strength, which penalises the model complexity to prevent overfitting; a higher $alpha$ means coefficients values are shrunk more. Too high regularisation can cause overfitting but too low regularisation can result in underfitting. Regularisation is multiplicative, and its effects are non-linear; I address this by tuning across an evenly spaced values on a log scale. Next, I tune the $l1\_ratio$, which determines the type of regularisation. $l1\_ratio = 0$ corresponds to L1 (Lasso) regularisation, which is linear, which leads to automatic feature selection by setting some coefficients to 0; this can make the model more interpretable. $l1\_ratio = 1$ corresponds to L2 (Ridge) regularisation, which is non-linear, shrinking coefficients more smoothly; L2 is efficient in reducing the effects of multicollinearity. $0 < l1\_ratio < 1$ is an Elastic Net, combining both types of regularisation. The optimal $l1\_ratio = 1$ makes sense, because there is high correlation between $season$ and $mnth$, and $temp$ and $atemp$ (Figure 7), making them highly multi-collinear.

LGBM hyperparameter tuning involved 4 parameters: $learning\,rate$, $n\_estimators$, num_leaves, min_child_weight. $learning\_rate$ controls the optimisation function's step size, i.e. how much the model adjusts with each boosting iteration when training; slower learning rates are more precise and tend to be more accurate, while higher learning rates have a faster training speed. LGBM performs well with small learning rates, so I tune over incrementally spaced values, small in magnitude. $n\_estimators$ is the number of trees (boosting rounds) - higher values increase precision at the cost of training time; num_leaves is the maximum number of terminal nodes. Too high or low values of both $n\_estimators$ and num_leaves may lead to over- or under-fitting respectively. min_child_weight is the minimum amount of observations required in a child node to split. Converse to the other hyperparameters, higher values of min_child_weight lead to more general splits, reducing overfitting.

I use $RandomizedSeachCV$ to decrease computation time when hyperparameter tuning, by randomly sampling a fixed number of combinations from the parameter grid (seeded for reproducibility), which is more computationally efficient than $GridSearchCV$ which checks every combination.

$RandomizedSearchCV$ splits the train dataset into 5 subsets (folds), called K-fold cross-validation, where one is used as validation and the other four as training; $RandomizedSearchCV$ iteratively tunes across each fold, using each fold as validation once, averaging the results to obtain more reliable estimates.

5 Evaluation

5.1 Comparing Models

Metric	GLM	LGBM
Mean Predictions	187.54	187.47
Mean Outcome	189.56	189.56
Bias	-0.01068	-0.01102
MSE (Mean Squared Error)	8009.83	1964.25
RMSE (Root MSE)	89.49	44.32
MAE (Mean Absolute Error)	59.81	27.65
Deviance	9699.80	3050.81
Gini Coefficient	0.4588	0.4958
Normalized Gini	0.9015	0.9742

Table 2: Model Evaluation Metrics for GLM and LGBM (tuned)

Table 2 displays evaluation metrics for the tuned GLM and LGBM models. Both models possess high predictive accuracy, seen through the mean outcome and low bias. However, when considering the error metrics - MSE, RMSE, MAE and Deviance - GLM is severely outperformed by LGBM. Figure 9 illustrates this; GLM has much a much larger error-term variance, exacerbated as the value of cnt increases, but LGBM errors consistently remain low.

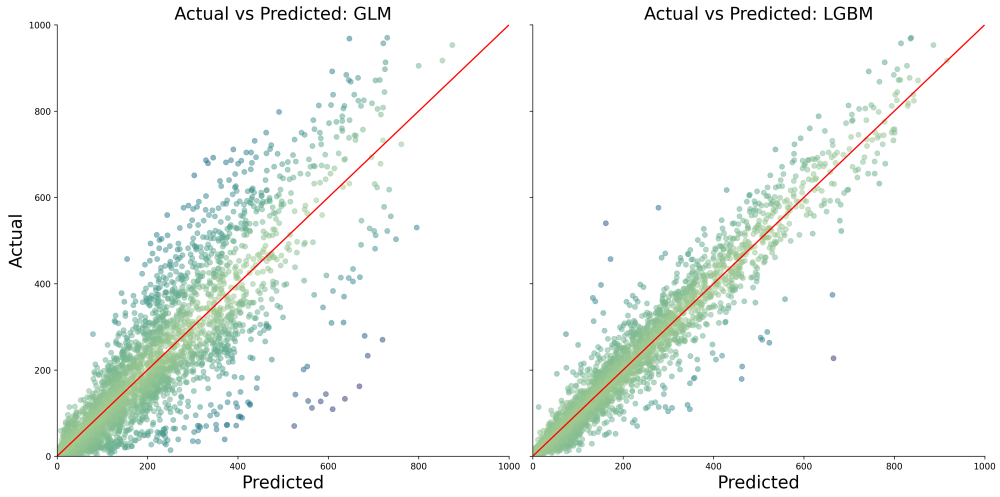

Figure 9: Predicted vs Actual

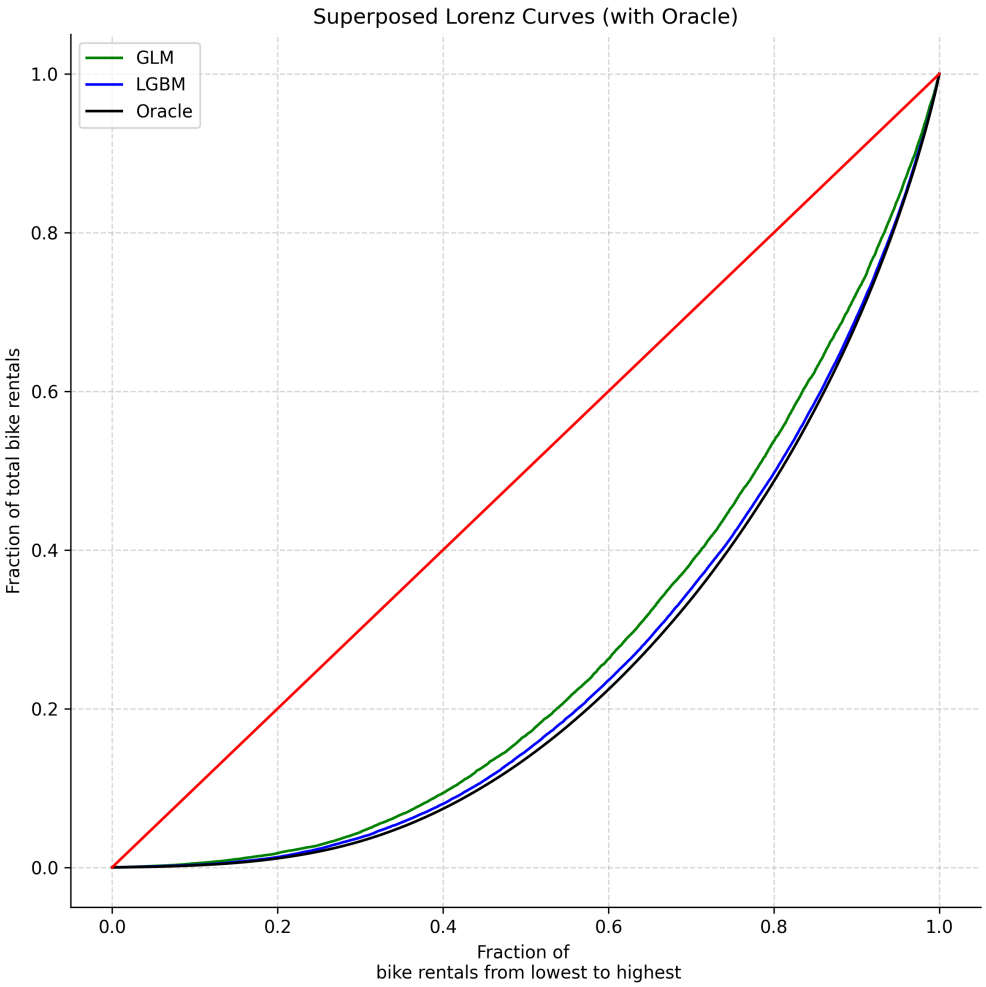

Figure 10: Lorenz Curve

In terms of discriminatory power, LGBM performs better on the test set, with predictions closer (than GLM) to the Oracle (Figure 10); this is also reflected in the Gini and normalised Gini coefficients.

5.2 Feature-specific Insights

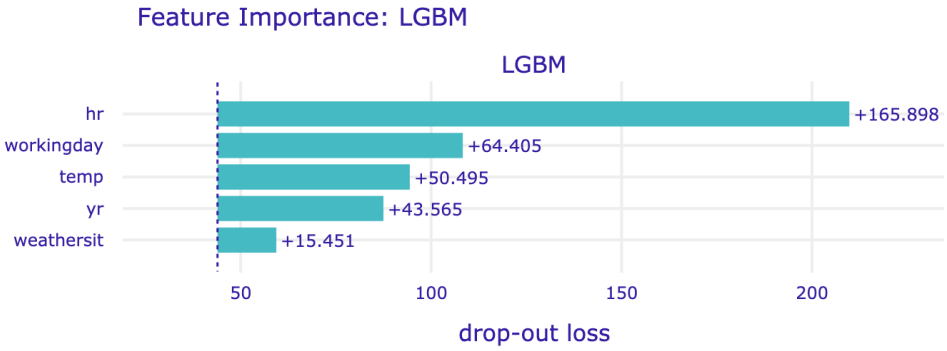

Figure 11: Feature Importance: LGBM

Analysing individual feature importance, Figure 11 ranks features by the change in loss if they were removed. Both LGBM and GLM have similar (not the same) features, but I focus on LGBM as it performs better. The results for hr, yr, and $temp$, are consistent with them being the features most correlated to cnt (Figure 7).

Figure 7 also notes high correlation between some of the features, so I use ALE plots instead (Figure 12), which analyse feature effects locally, avoiding impossible feature combinations. Figure 12 is contradictory of Figure 11 (for both models), finding that individual changes in hr, $workingday$, yr, and $weathersit$ have no significant effects on prediction. Partial dependence plots (not shown) find a similar insignificant effect. The reason for this contradiction is that Figure 11 measures the overall contribution of a feature, rather than changes in feature values, to the predictive power of the model. Figure 11 accounts for feature-interaction effects as well, which does not occur with ALE (or PDP) plots, because of the ceteris-paribus analysis.

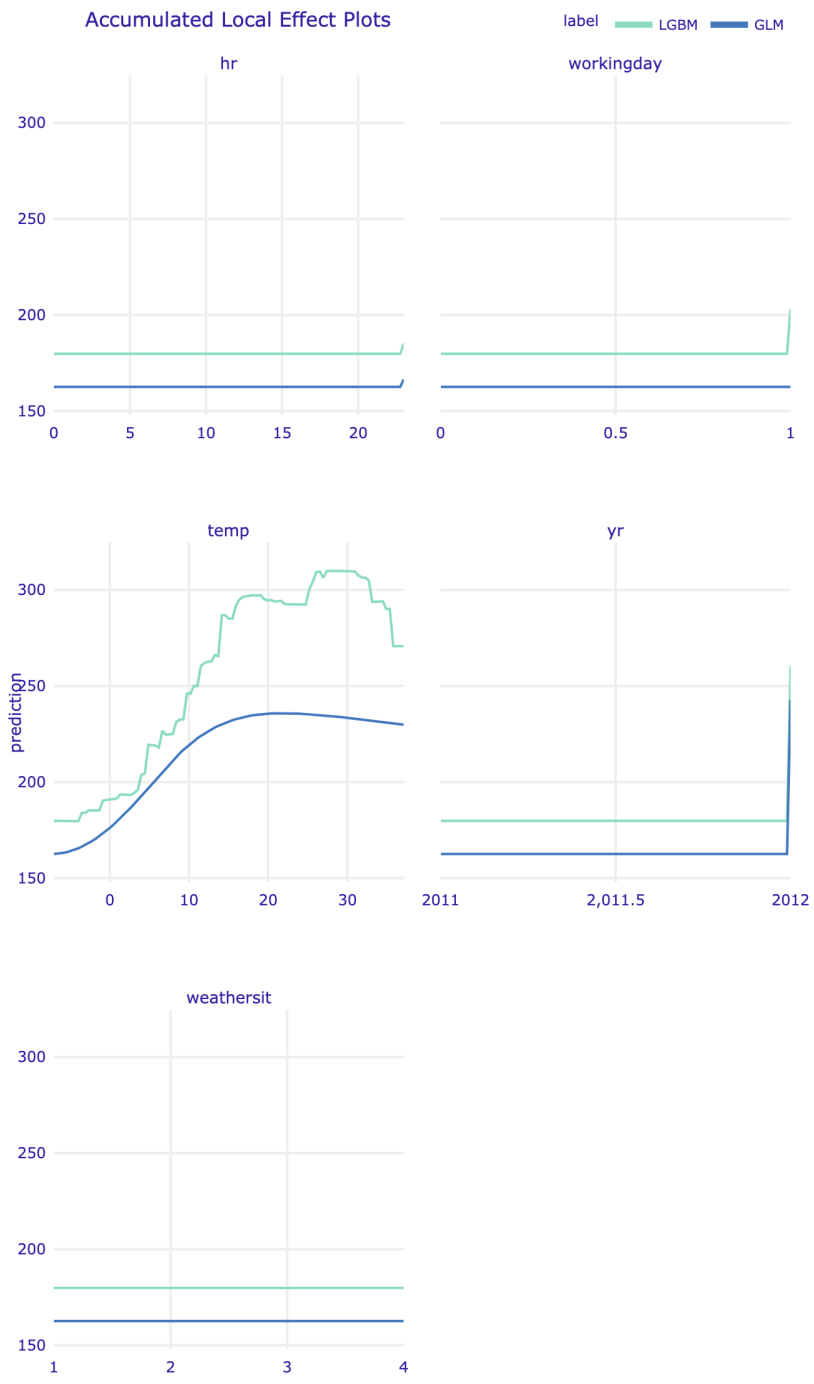

Figure 12: Accumulated Local Effects (ALE) Plot

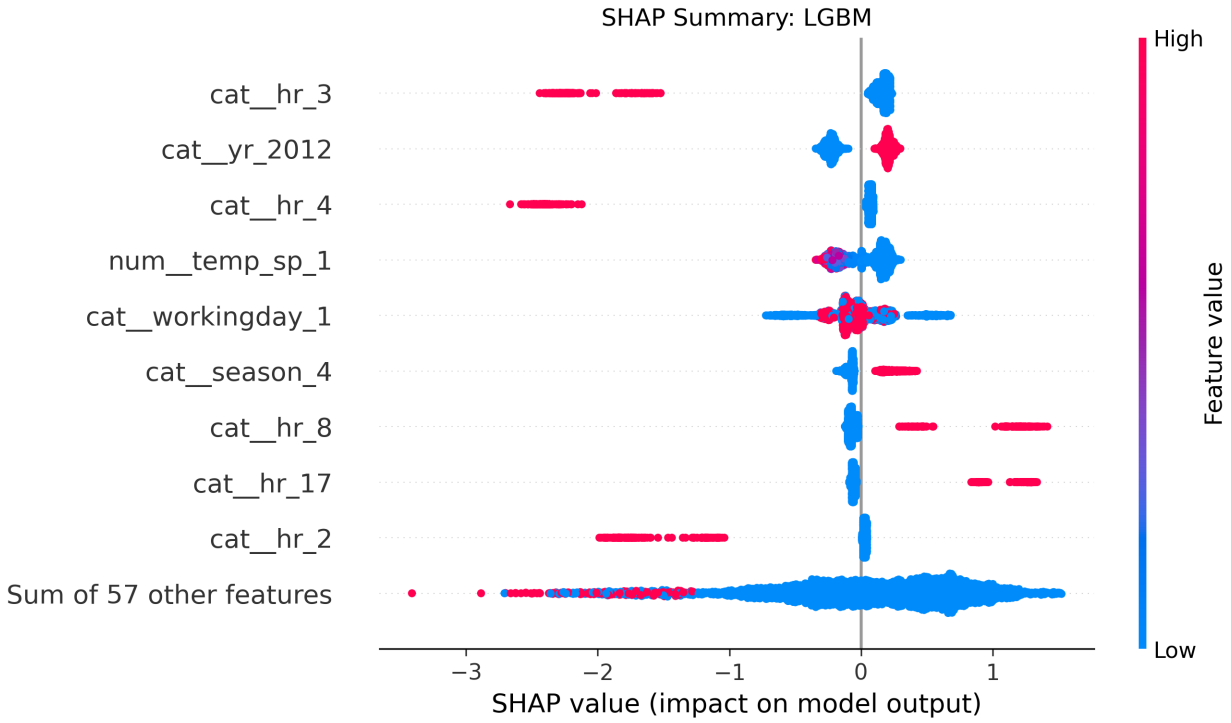

Figure 13: Beeswarm Plot of LGBM Shapley Values

Figure 13 plots Shapley Values for the top nine contributing features, which are better at analysing interaction effects; the plots in Figure 13 represent the changes in Shapley values caused by different values of a feature. Note that the values are plotted for processed features (so augmented feature names are presented).

When $workday = 1$, the variance of Shapley values are decreases, despite the mean seeming the same. Similarly, $hr = 8$ and $hr = 17$ both have strong positive effects on Shapley values. 8am and 5pm are the times at which work starts and finishes; it is likely that bikes are used to commute to and from work, which explains the consistency, and therefore lower variance in Shapley values when $workingday = 1$ - this is also consistent with Figure 11, suggesting larger interaction effects, and smaller individual effects.

6 Conclusion and Improvements

The objective of this project is to accurately predict hourly bike rentals from Capital Bikeshare in Washington D.C. I use a mixture of time-related and environmental features to do so. I use GLM and LGBM models, tuned using k-fold cross-validation. The tuned models are evaluated and compared against each other across a number of metrics. I also analyse feature-specific insights. I conclude there exists relatively larger interaction effects than individual effects, which is why feature effects are better explained by Shapley values than with Accumulated Local Effect plots. The most prominent trend is the correlation with work: cnt sees more consistency in its values during workdays, and has spikes at 8am and 5pm - bikes are likely a form of commuting.

There are two main points to be addressed to improve the model. First, $yr = 2012$ has a large, positive effect on values of cnt (Figure 11; Figure 13), but since data only exists across 2 years, this effect (and therefore the models) cannot be generalised. A larger training dataset is an obvious solution, but increased domain knowledge could also lead to the identification of other between-year features which could have a more causal effect.

Second, the models could be tuned iteratively; evaluation metrics can be used to identify features which are likely superfluous that can be removed, and performance can be assessed again. Furthermore, more attention could be paid to time-related variables, modelling for a time trend, or cyclical encoding, rather than just using fixed effects encoding.

References

Fanaee-T, Hadi (2013) “Bike Sharing,” Published: UCI Machine Learning Repository.

Fanaee-T, Hadi and Joao Gama (2013) “Event labeling combining ensemble detectors and background knowledge,” *Progress in Artificial Intelligence*, 1–15, 10.1007/s13748-013-0040-3, Publisher: Springer Berlin Heidelberg.