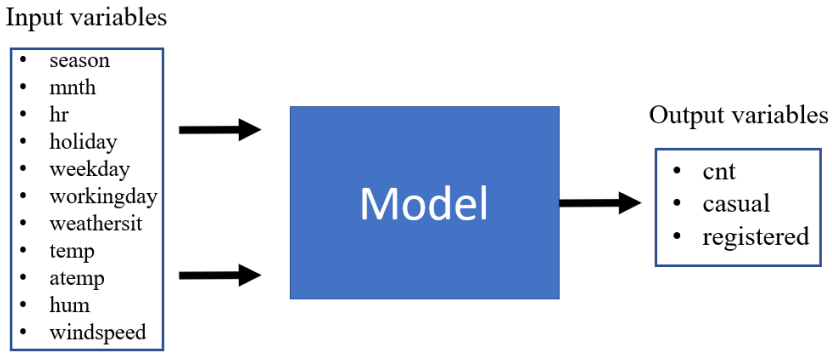
Bike Sharing Data Set Analysis and Modelling

Themistoklis Koutsellis

Introduction & Notations



All figures/plots demonstrated in section Appendix of this file.

The python code which was used to analyse/visualize the data and build up the prediction model is in *bikesharingdatamodeling.py* file. In that file, class <*BikeSharingDataModeling*> is defined. This class contains all necessary methods for data analysis and model prediction configuration.

Analysis

- No missing data: See ..*check_for_missing_data ()*.
- Unnecessary fields: I)'dteday', 'instant". They are index variables. II) "yr" field, because there are only 2 years of records, an insufficient amount of years for statistical inference in terms of years. III) "season", "'casual", "registered", "atemp". There are high levels of correlation between these and other associated fields (See Appendix Figure 10), i.e. corr(month, season) = 0.83, corr(count, casual) = 0.69, corr(registered, count) = 0.97, corr(atemp, temp) = 0.99.

A closer look of all Figures indicates the following:

- The "registered" users contribute the most to the overall count of total rental bikes (cnt).
- During summer & fall season, that is May to October, the cnt is higher (see Figure 1 & 2).
- Casual and registered users exhibit different usage pattern. Registered users mostly use bikes during the rush hours (7-8 am & 4-6pm). On the other hand, casual users tend to use bikes during the afternoon hours (12-5 pm). See Figures 3 & 7. Casual users increase their biking hours during holidays and weekends, whereas registered users decrease their bike riding hours during these periods. (see Figures 4, 5 & 6). Based on all the above, we may assume that the registered users belong to the working people (they rest on holidays/weekends at home and they move out during rush hours). On the other hand, casual users, might be people with no job moving around the town/city i.e. housekeepers, kids, students, unemployed, tourist etc. This group of people bike more during vacations, weekends and afternoon hours (non-rush hours).

Regarding the overall users cnt:

- Users tend to behave in different manner during working days and weekends. On Weekends they bike during the afternoon, whereas on working days they bike during during rush hours (see Figures 8 & 9)
- Users tend to bike less when the weather conditions deteriorate (see Figure 11). When the humidity increases (bad weather conditions) the users bike less (see Figure 12). On the other hand, when the temperature increases, the user bike more (see Figure 13). The wind speed seems to have less contribution to users' behaviour. Figure 14 indicate an almost constant usage rate [around 150 to 200 counts/hour] when wind speed is changes. This, combined with the low correlation value between wind speed and total counts, corr(count, windspeed) = 0.093 (see Figure 10), indicates that wind speed does not affect much the values of counts (cnt) and therefore we can drop this field as well.

Summary: dropped fields = [dteday', 'instant", "yr", "season", "casual", "registered", "atemp", "windspeed"].

Figures 15 & 16 demonstrate the existence of lots of outlier data points. Moreover, the PDF (probability density function) of count (cnt) variable looks like an exponential. Hence, it is preferable to train and test a nonparametric algorithm for the next step (choose a prediction model).

Prediction model

After a) data "dummification", b) dropping the data set fields with no info value, c) splitting the data set into train and test data (proportion: 0.7:03 respectively and d) then comparing a list of regression models, based on the cross_val_score (choose model with the smallest absolute value of cross_val_score), I concluded that the Random Forest Regressor model is most suitable for predictions.

Models to compare: Ridge, Lasso, LinearRegression, BaggingRegressor, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor.

All the aforementioned implemented by See model_selection() of < BikeSharingDataModeling> class in bikesharingdatamodeling.py

Below, the Mean Absolute Deviations:

Temp	0.165381
Hum	0.163957
mnth_2	0.143434
mnth_3	0.157726
mnth_4	0.151029
mnth_5	0.158814
mnth_6	0.147725
mnth_7	0.157454

mnth_8	0.154044
mnth_9	0.14662
mnth_10	0.154181
mnth_11	0.152401
mnth_12	0.155137
hr_1	0.078673
hr_2	0.078673
hr_3	0.075195
hr_4	0.078221
hr_5	0.076104
hr_6	0.080934
hr_7	0.081536
hr_8	0.082889
hr_9	0.080483
hr_10	0.081386
hr_11	0.077919
hr_12	0.080784
hr_13	0.077314
hr_14	0.079578
hr_15	0.080483
hr_16	0.080633
hr_17	0.080332
hr_18	0.084089
hr_19	0.079729
hr_20	0.081536
hr_21	0.078372
hr_22	0.077012
hr_23	0.080483
holiday_1	0.053559
weekday_1	0.243032
weekday_2	0.242089
weekday_3	0.246205
weekday_4	0.240551
weekday_5	0.248309
weekday_6	0.244445
workingday_1	0.432461
weathersit_2	0.382667
weathersit_3	0.149792
weathersit_4	0.000329

Question: Assume that the code you are writing is used in production in a daily prediction service and maintained by your colleagues (what could that mean?) Please do not spend more than a few hours on the topic. Your code and predictive model are of equal importance to us.

Answer: The code/script should be self-explanatory, and the app it implements user friendly. The code must be organized in functions, classes etc. so that the colleagues can make changes to small snippets, without affecting the rest of the code. For all above, The code should follow the PEP 8 -- Style Guide for Python Code. Finally, if a git service provider is used (i.e. gitHub), the colleagues must have access to the repository where the code resides and perform certain actions, such as “fork”, “clone”, “pull”, “push”. Permission for these actions should be given according to certain criteria based to company’s policy, quality and safety standards.

Appendix

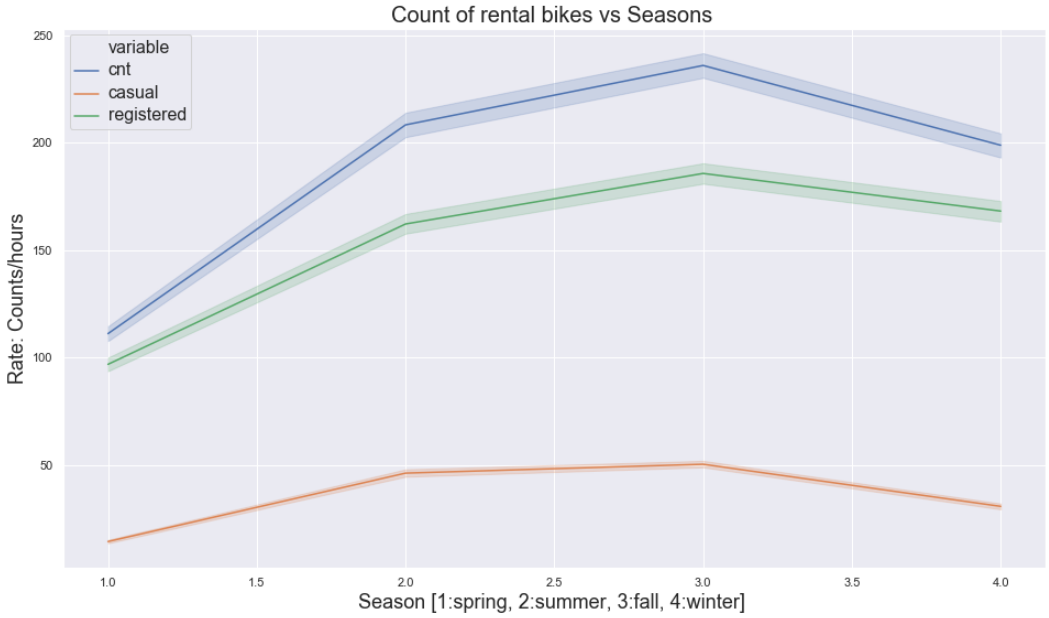


Figure 1. method: cnt_vs_season()

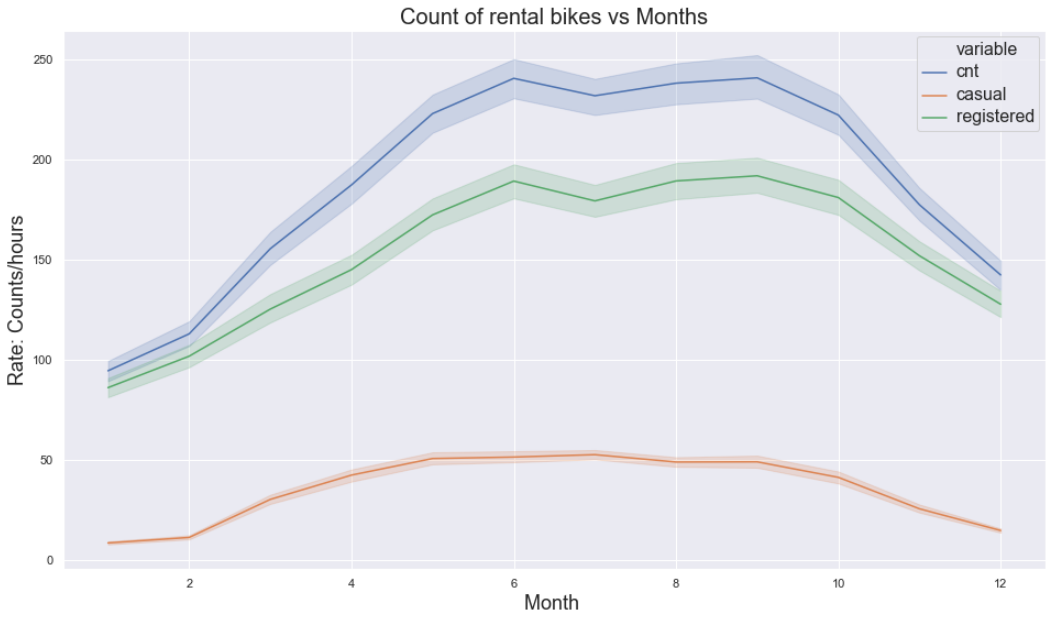


Figure 2. method: cnt_vs_month()

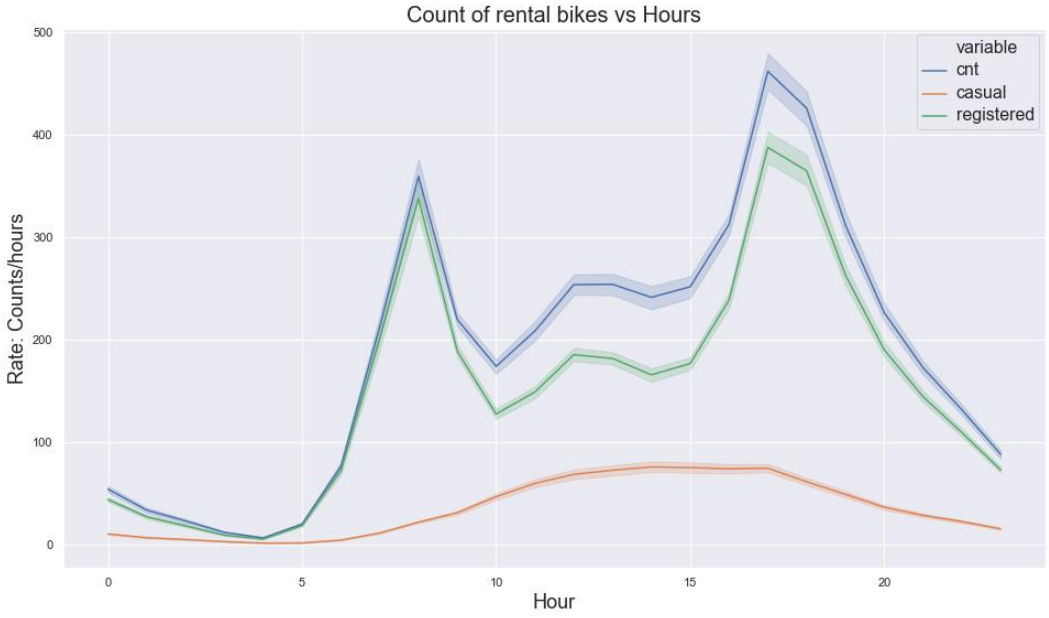


Figure 3. method: cnt_vs_hour()

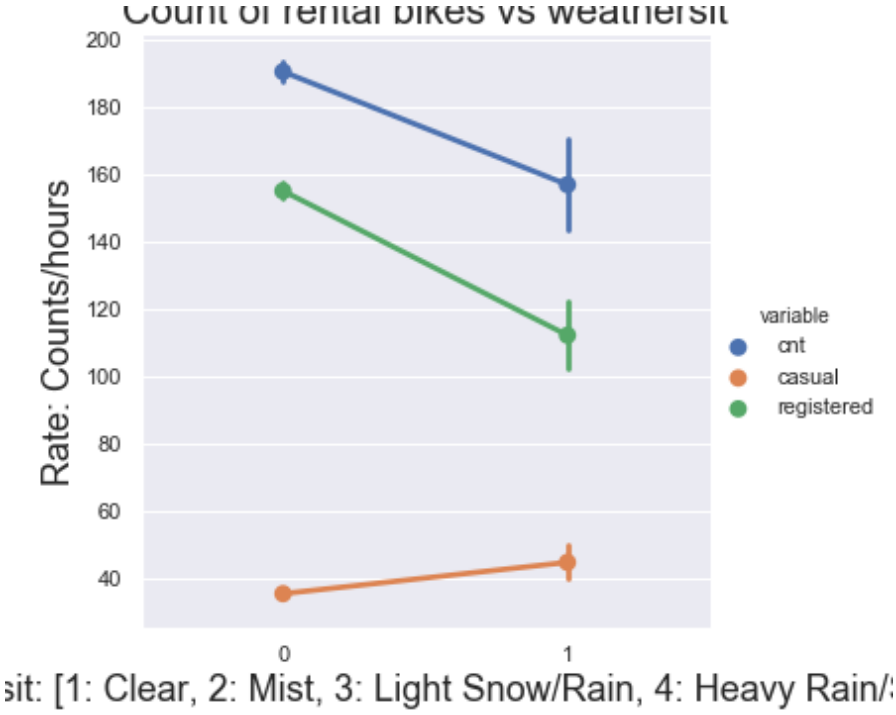


Figure 4. method: cnt_vs_holiday()

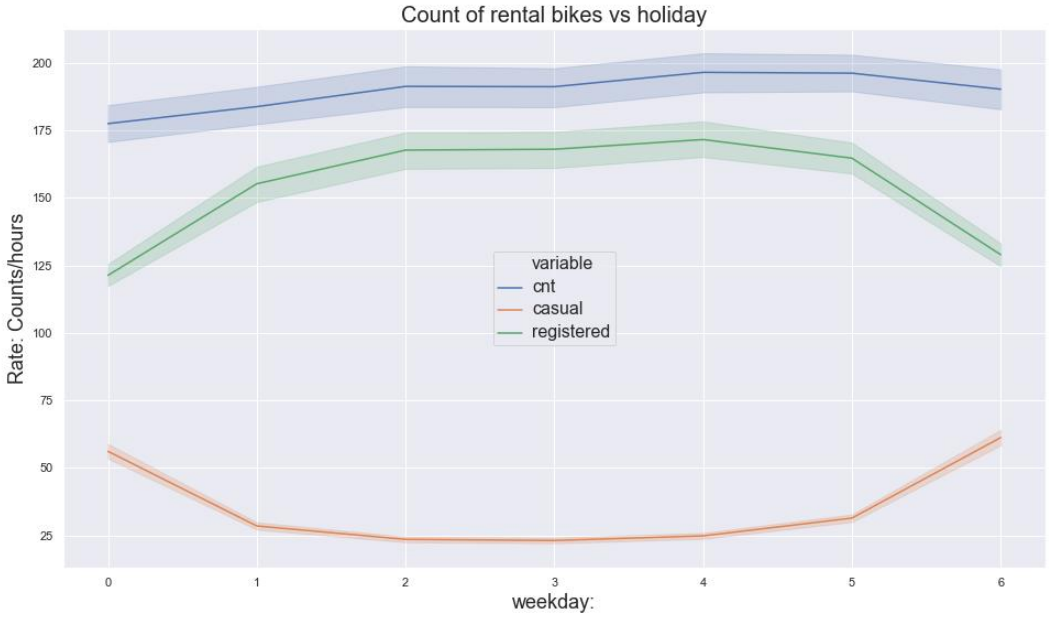


Figure 5. method: cnt_vs_weekday()

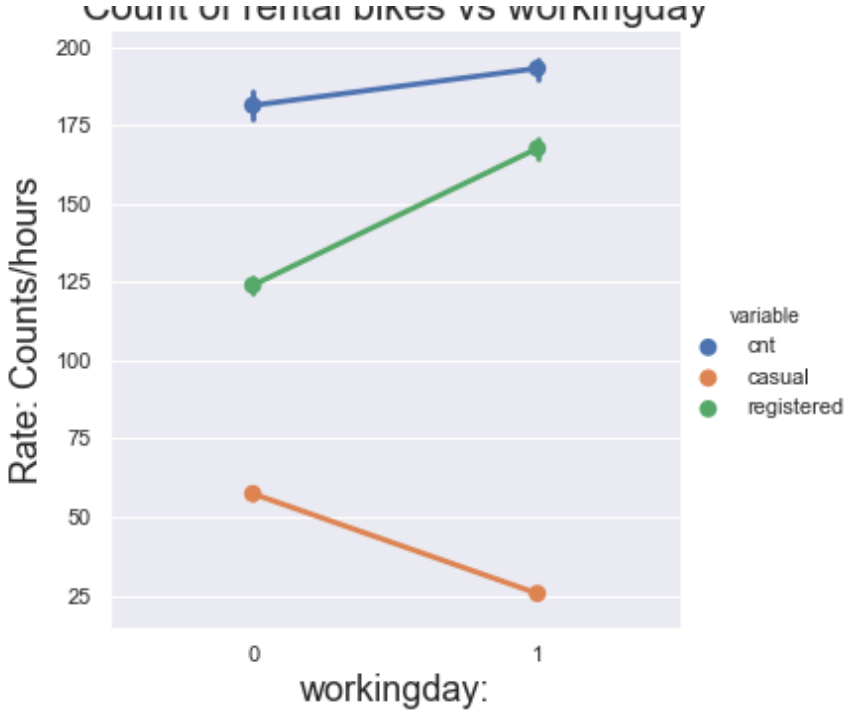


Figure 6. method: cnt_vs_workingday()

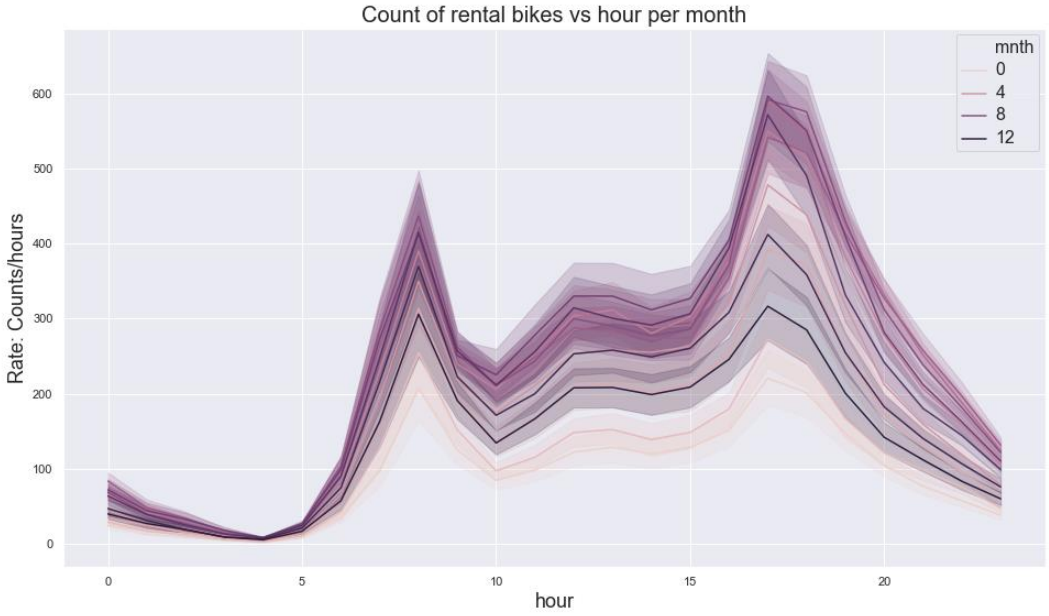


Figure 7. method: cnt_vs_hours_per_month()

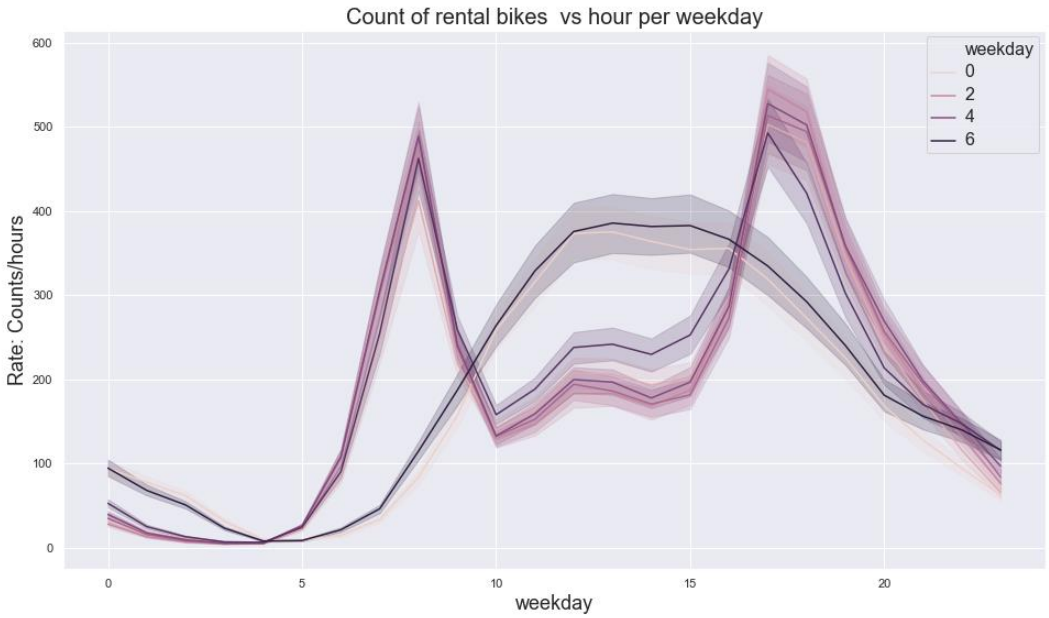


Figure 8. method: cnt_vs_hours_per_weekday()

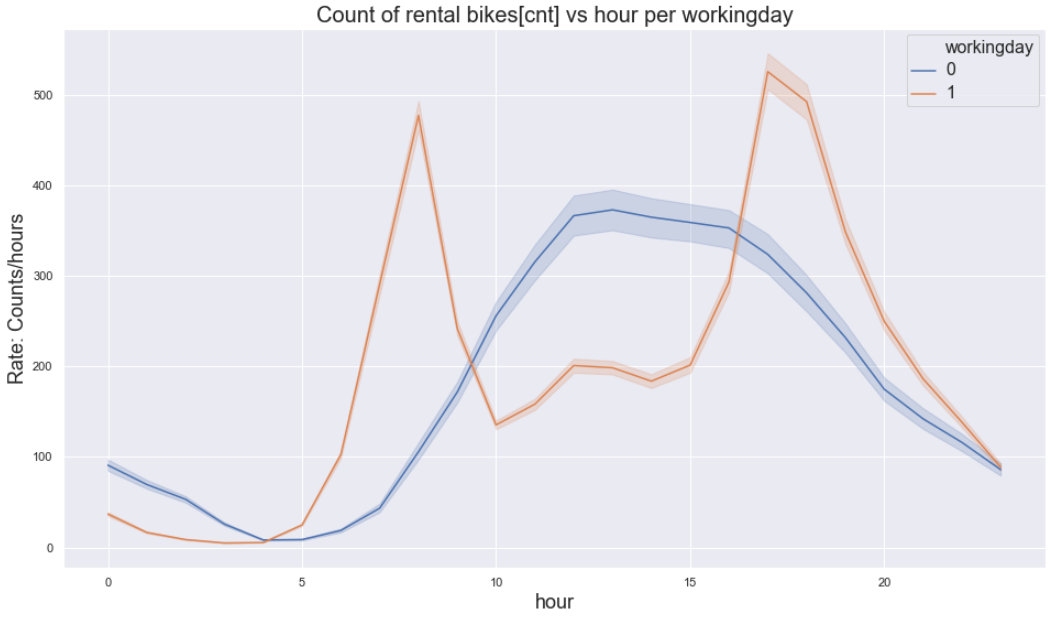


Figure 9. method: cnt_vs_hours_per_working_day()

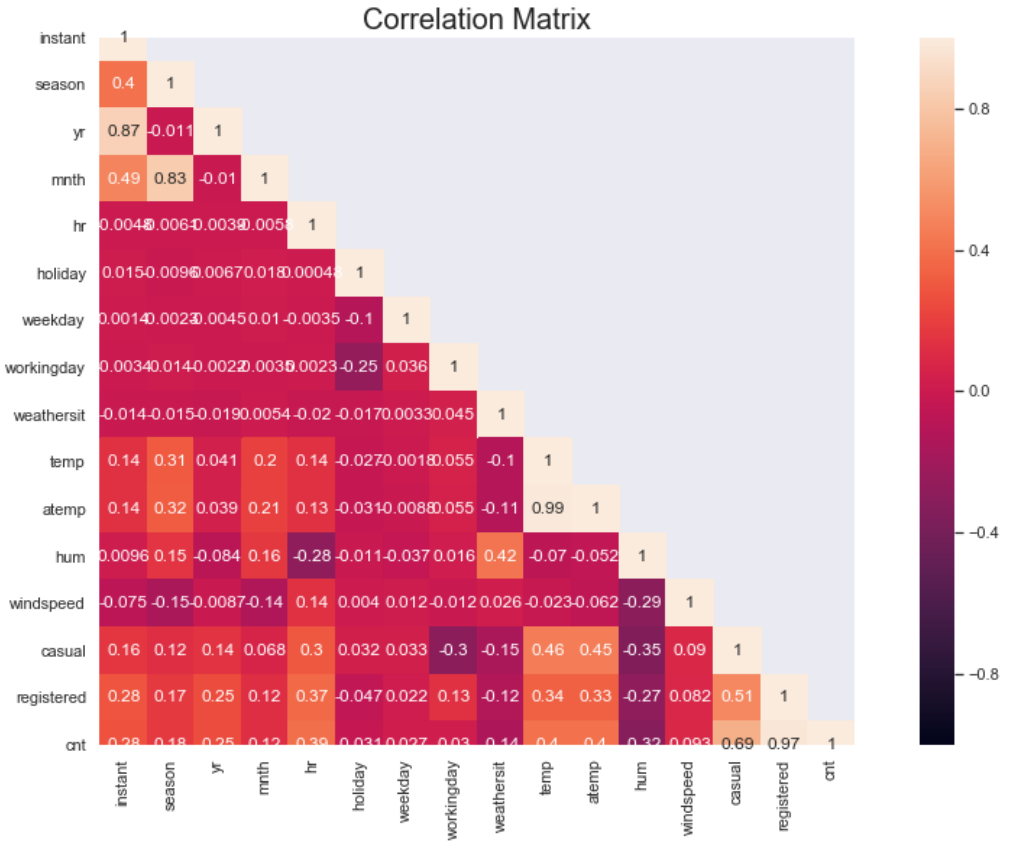


Figure 10. method: corr_matrix()

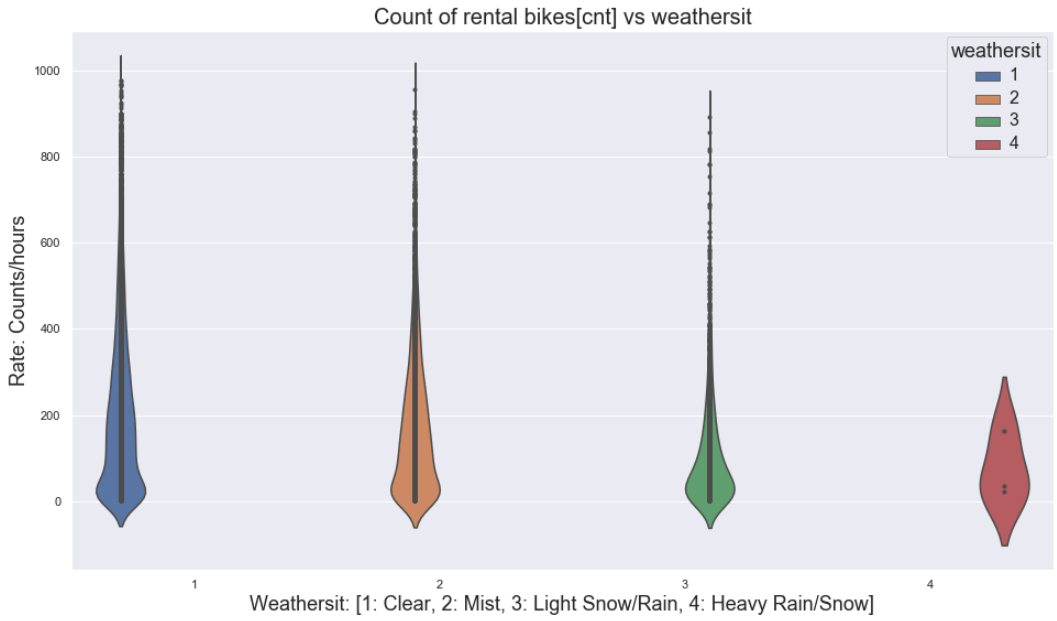


Figure 11. method: cnt_violin_weathersit()

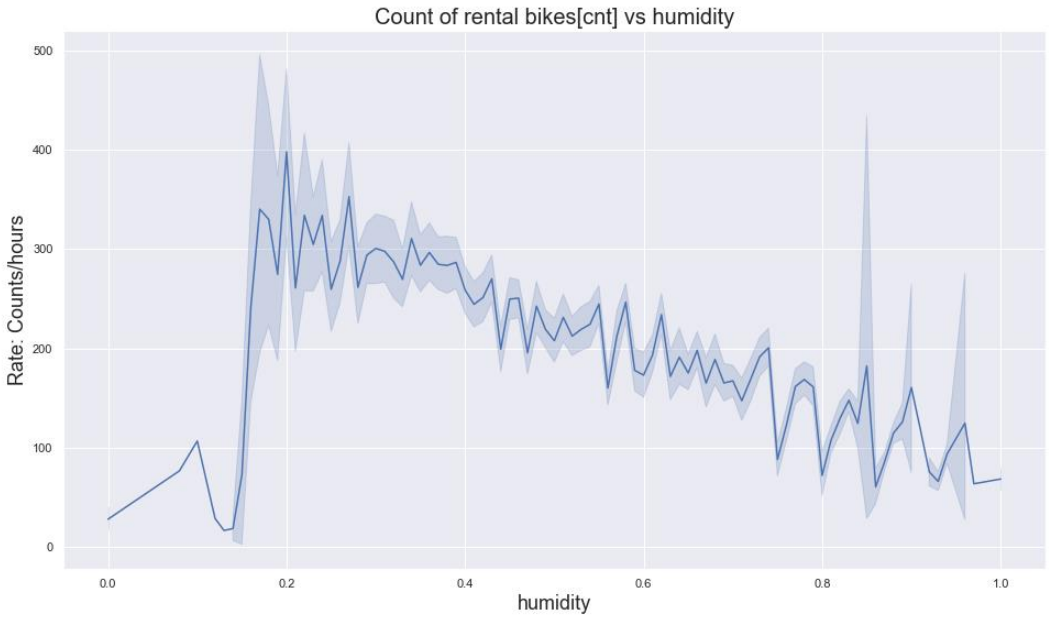


Figure 12. method: cnt_vs_hum()

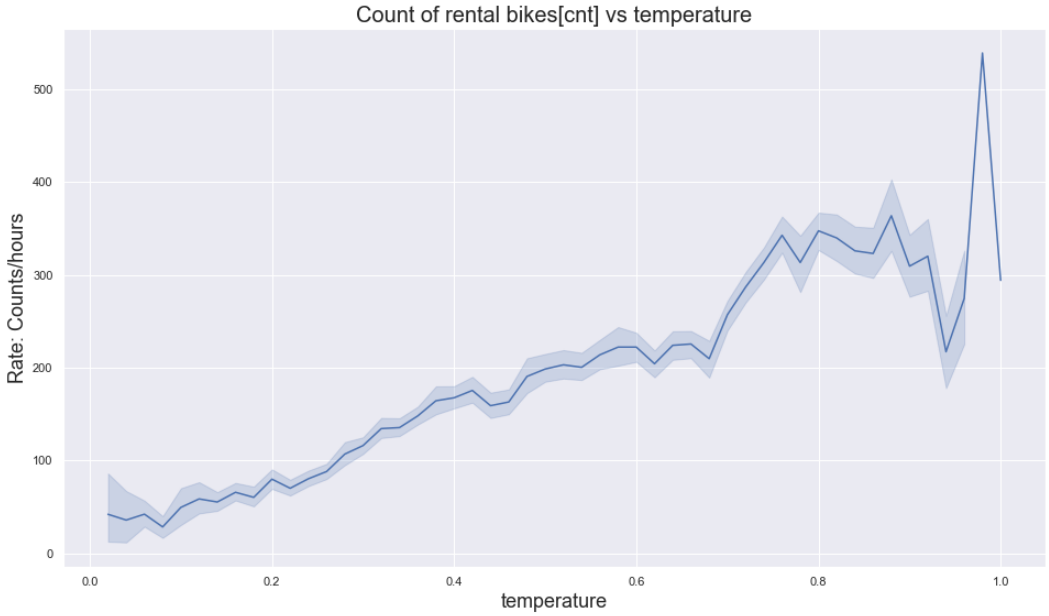


Figure 13. method: cnt_vs_temp()

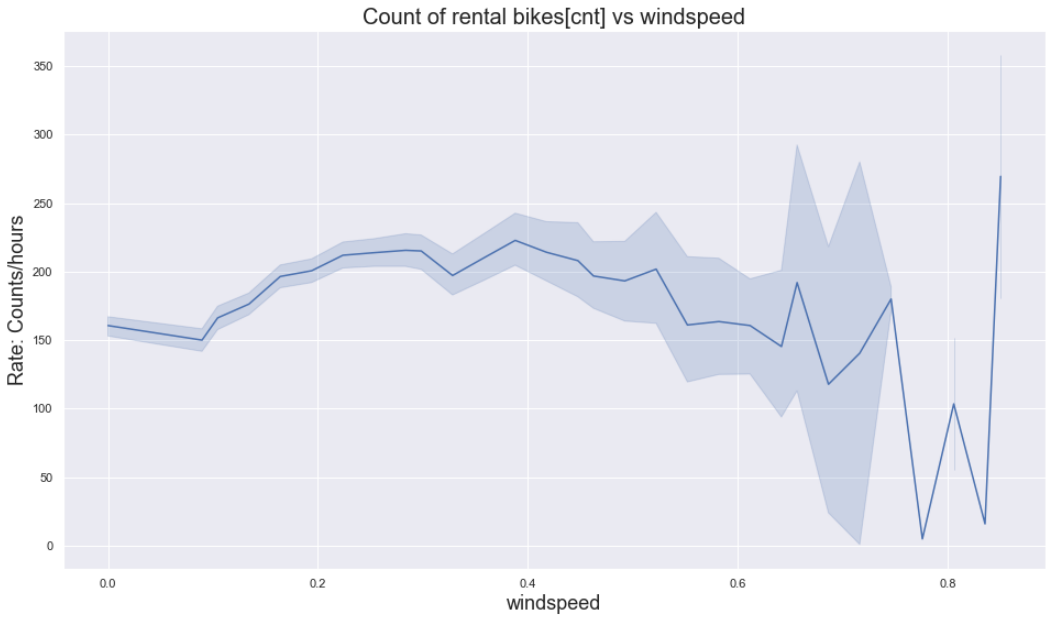


Figure 14. method: cnt_vs_windspeed()

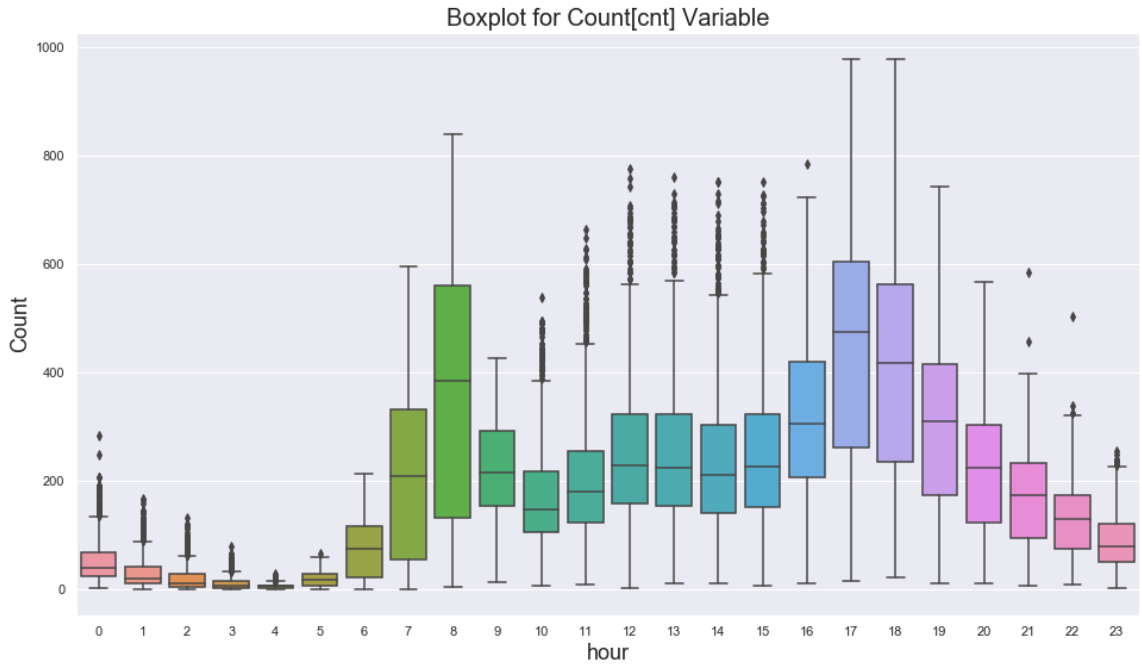


Figure 15. method: box_plot_cnt_vs_hours()

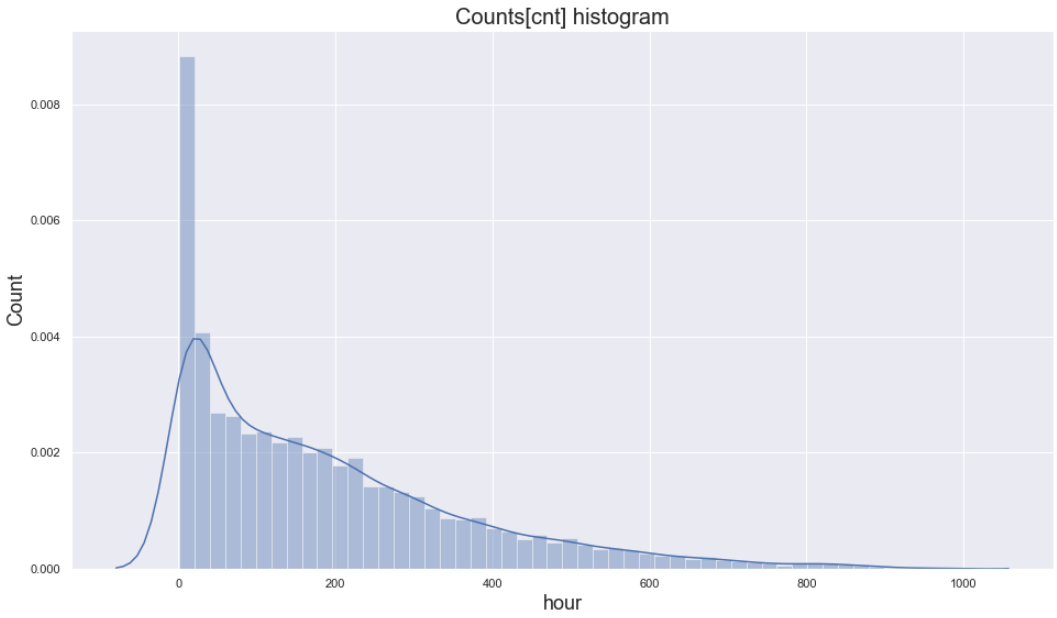


Figure 16. method: histagram_cnt_vs_hours()