Bike sharing dataset

Part - 01

1. Objective

To do an explorative data analysis and build a prediction model for the hourly count of bike sharing data.

2. Dataset

The data set consist of 17379 observations over a period of 2 years, starting 2011 through 2012. It has 15 possible predictors and 1 target variable. Out of the 15 predictors, consist of 7 continuous , 7 categorical and 1 datetime variable.

3. Approach taken

a. Pre-processing:

- Renaming the columns
- Checking and converting data types based on categorical and numerical predictors
- Check for missing values – No missing values were found
- For better clarity mapping the values(numbers) in season, month, weekday and weather columns to corresponding description from the data description file

b. Exploratory data analysis:

- Outlier analysis and treatment:

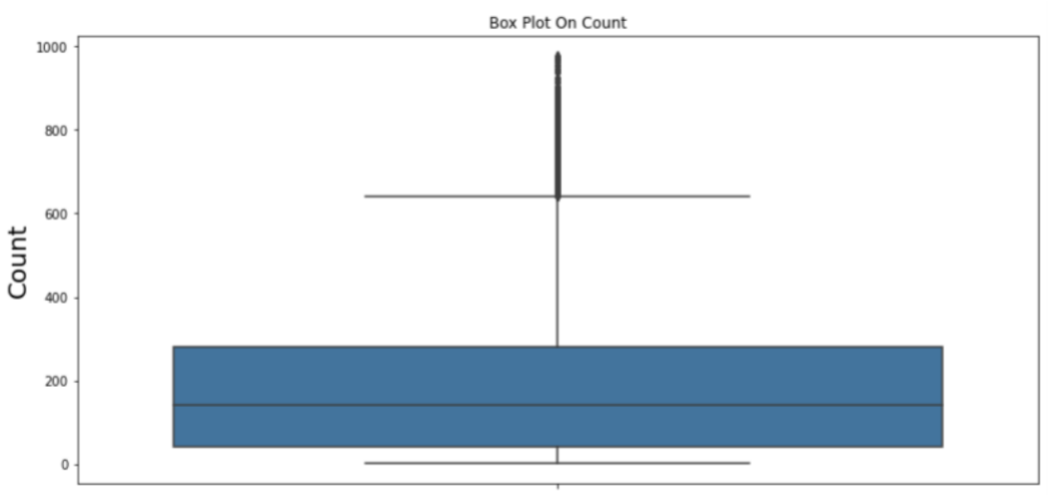

Fig 1: Box plot on count

Looking at Fig 1, we can see that the target variable 'count' has many data points outside the maximum whisker which leads us to believe the presence of outliers in the dataset. 244 outliers were removed from the dataset

- Check the distribution of the target variable – ‘count’

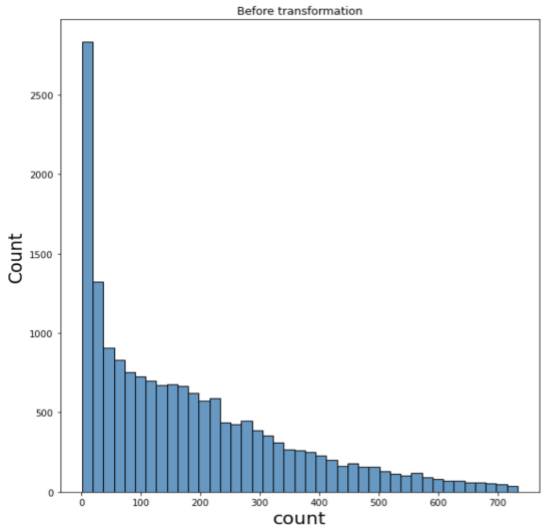

Fig 2: Distribution of target variable before transformation

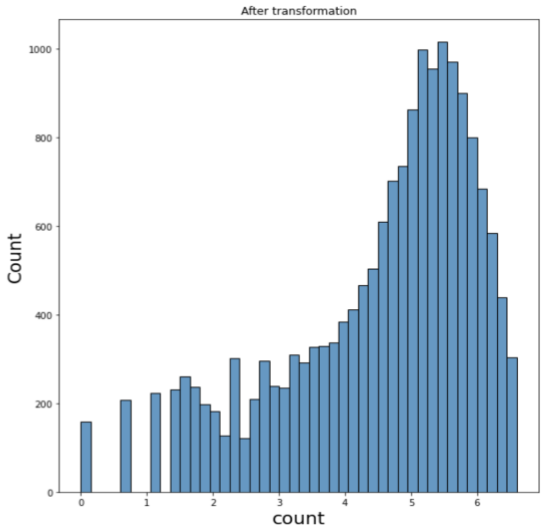

Fig 3: Distribution after transformation

From fig 2: The count variable is skewed towards right before transformation. Most of the machine learning techniques require dependent variable to be normally distributed, so the best practice would be to transform the skewed dependent variable to a normally distributed one. Fig 3: Taking log transformation on count variable after removing outliers makes the distribution much better, still not a perfect normal distribution though.

- Check the behavior of dependent variable with respect to categorical features:

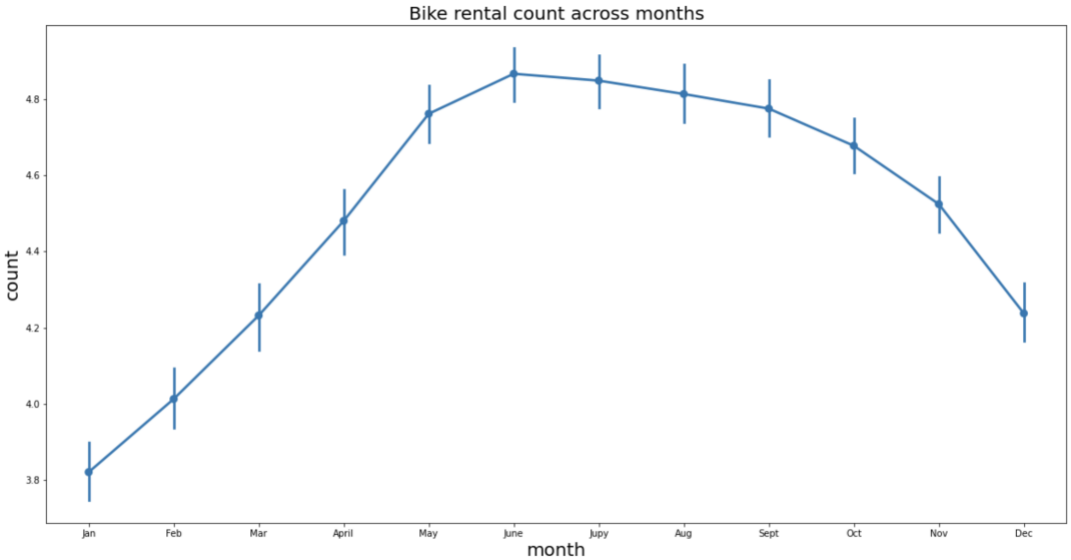

Fig 4: Bike rental count across months

Insights:

The bike rental count shows a steady increase as we approach the summer months. It peaks in June-July. and then starts falling gradually as winter approaches. This is in line with expectation of high usage of bike in summers than in winters.

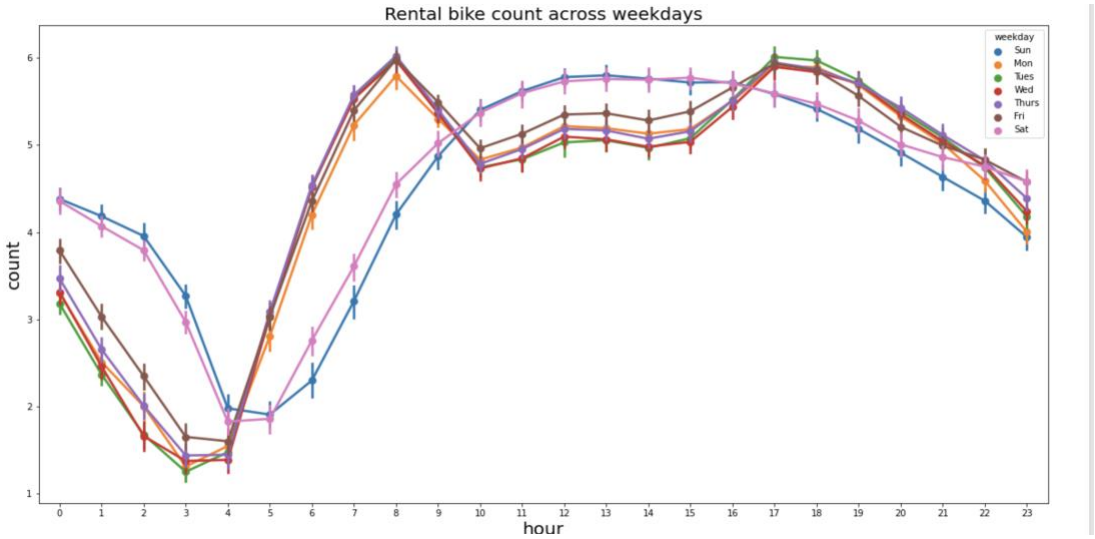

Fig 5: Bike rental count vs hour across weekdays

Insights:

Difference in the patterns on a weekend vs weekday can be clearly seen. On the weekdays, the bike rental peaks out twice in a day, around 7am -8am and then around 5 pm-6 pm. This can be attributed to the high usage of bike as a means of commutation to work or university or schools. Whereas, during weekends, there is a gradual increase in the bike rental count as the day progresses till 4pma and we see a gradual decrease as the day progresses towards night.

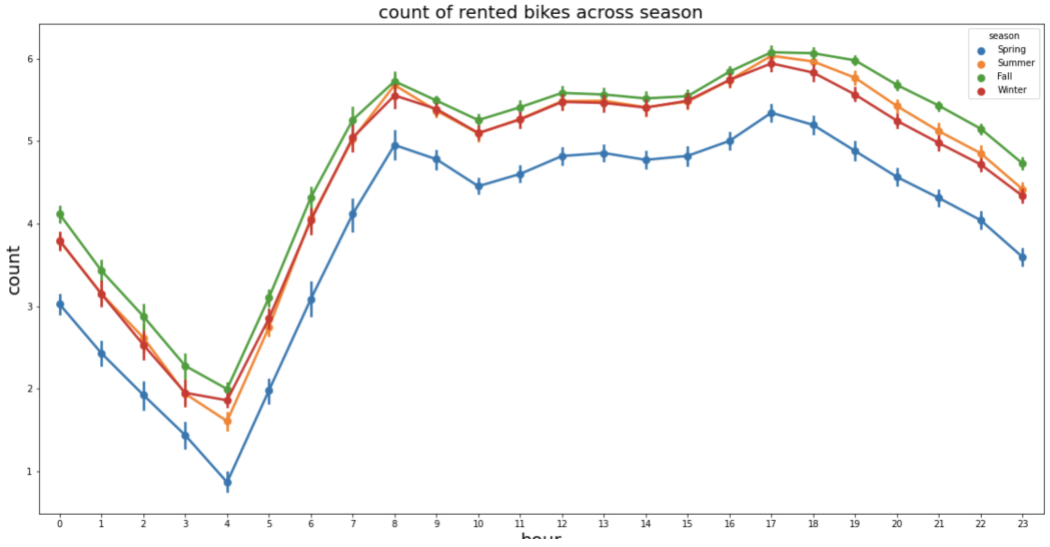

Fig 6: Bike rental count vs hour across seasons

Insights:

1) The hourly pattern of bike rental counts over hours in the day is same for all the seasons.
2) Spring is the season with the lowest bike rental counts.

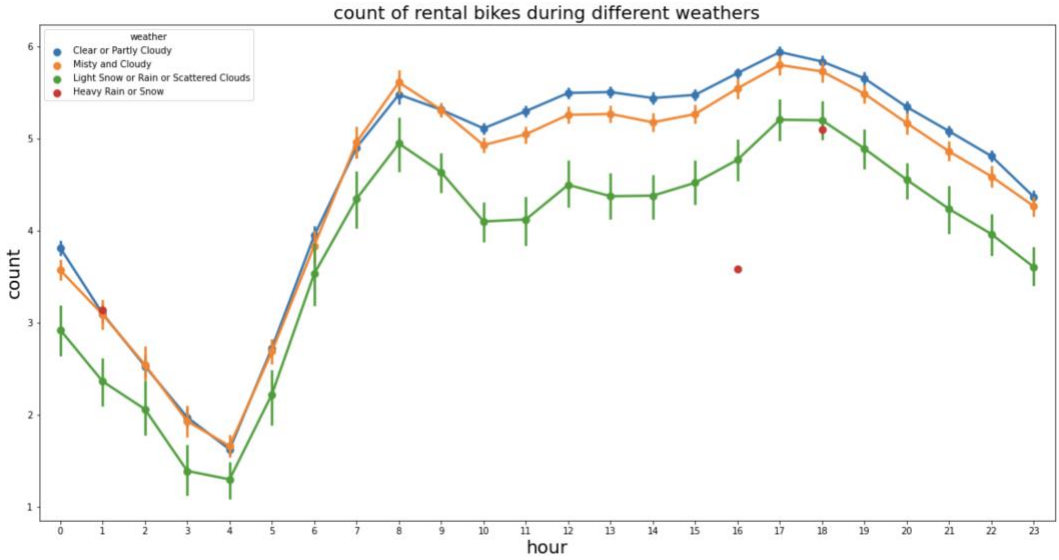

Fig 7: Bike rental count vs hour across weathers

Insights:

When the weather is very rough and severe like heavy rain and snow, the bike rental count is very close to 0 and that is as expected. Clear or partly cloudy weather conditions see the maximum bike rental count.

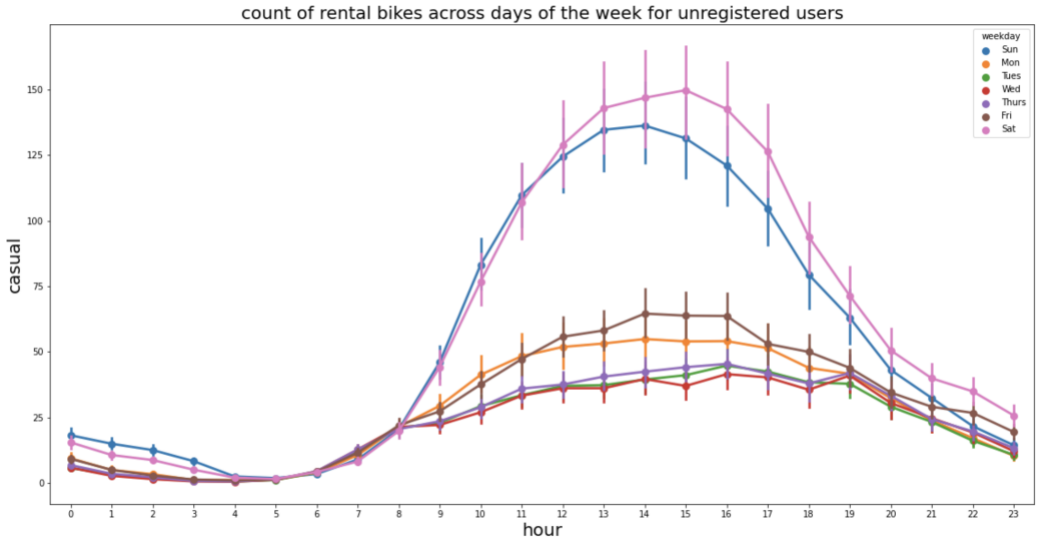

Fig 8: Bike rental count vs hour across days of the week for unregistered users

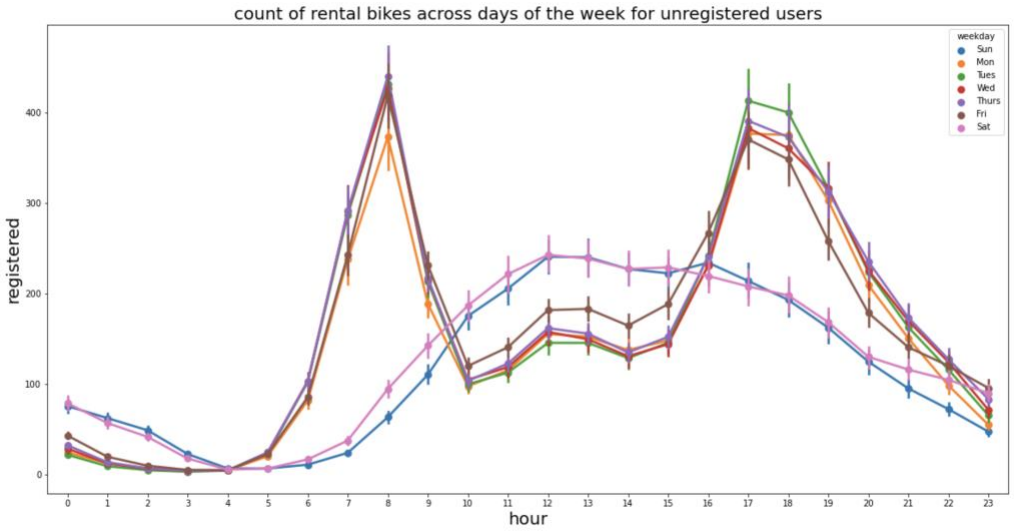

Fig 9: Bike rental count vs hour across days of the week for registered users

Insights:

1) The graph for unregistered users might point to the fact that most unregistered users are casual users and they do not show any sharp rise in count during a particular time of the day. The demand rises gradually as the day progresses until 2 pm and then gradually declines. The over-all demand is the highest on weekends.

2) The graph for registered users might point to the fact that most registered users are regular users. There is a peak in the demand at 7am - 8am and again at 5pm - 6pm on. weekdays probably. This could be attributed to the need of commutation to work place, university, schools during the weekdays.

- Correlation analysis and feature selection on numerical features:

Fig 9: Correlation matrix

Insights from the correlation matrix above:

a) 'atemp' and 'temp' are strongly correlated with each other. It is quite expected as the apparent temperature depends on the actual temperature. Hence one of the features has to be dropped since they exhibit multicollinearity in the data. 'atemp' was dropped.
b) 'temp' has noticeable positive correlation with count. Hence, we keep this feature.
c) 'humidity' has a noticeable negative correlation with count. Hence, we keep this feature as well.
d) 'windspeed' has a very low correlation value with count. This feature was dropped.
e) Casual' and "Registered" will also be dropped as they are leakage variables because sum of casual and registered is equal to total count.

c. Deciding the algorithm:

3 algorithms were considered – Linear Regression, Random Forest and XGBOOST.

XGBOOST was chosen as the final algorithm for the following reason.

a) XGBOOST works well when the data satisfies the following conditions –
   i. Number of features < numbers of observations in training data set. (Krishna)
   ii. When the dataset has a mix of categorical and numerical features.(Krishna)
b) It is based on gradient boosting decision tree algorithm. It is an ensemble learning method that helps reduce variance and bias.
c) **Mean Absolute Error = 0.2818141**. It was the lowest among all the 3 algorithms.

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?)

It means the following –

1) The code should be clean and well documented. Possibly checked using pylint
2) The code should be maintained properly for any changes or bug fixes in the due course. The changes should be documented and maintained using a version control system like Git.
3) Possibly a handover documented should be prepared and handed over to the support team with the steps to re-run the code in case of failure. It should also contain errors and issues faced in past.
4) The code should be built in a way that is scalable for foreseeable future. However, monitoring mechanism to keep a track of runtimes should also be established.

Part - 02

What are the scaling properties of your model, if you assume that the amount of data you need to handle go up to several terabytes? Do you see any problems?

Since the pandas dataframes are stored in memory, there is a limit to the amount of data that can be processed at a time (GeeksforGeeks). If the size of the dataset is greater than the available RAM, it will lead to memory errors.

How would you address these problems? Are there technologies for data storage/predictive modelling you can build upon?

To overcome this, following solutions can be implemented –

a) Use of parameter 'chunksize' to load data in chunks (GeeksforGeeks).
b) Use of Dask – Dask is a library that makes use of parallel and distributed computing (Dask).
c) Use of vaex - high-performance DataFrame library in Python, primarily built for the processing, exploration and analysis of datasets as large as the size of hard-drive and on a single machine(Veljanoski)
d) Use of Pyspark – Pyspark can significantly speed up the processing by combining local and distributed data processing. Benefits of using Pyspark:
    - Swift processing – Usage of PySpark will likely get a high data processing speed of about 10x faster and 100x faster in memory.
    - In memory computation – PySpark has DAG execution engine that helps in-memory computation . The data is cached, preventing data fetch from the disk every single time. (Technologies)

What are the limits and drawbacks for your new approach?

a) No real time data processing
b) No file management system
c) Spark provides fewer algorithm as compared to python
d) Difficult implementation due to the need to convert numerical problems to map/reduce problems

Do you have hands-on experience with such technologies?

I don't have hands on experience on these technologies.

References:

Dask "https://docs.dask.org/en/stable/."

GeeksforGeeks "https://www.geeksforgeeks.org/bypassing-pandas-memory-limitations/."

Krishna, H. "https://www.kdnuggets.com/2020/12/xgboost-what-when.html." from https://www.kdnuggets.com/2020/12/xgboost-what-when.html.

Technologies, B. "https://www.besanttechnologies.com/pyspark-programming." from https://www.besanttechnologies.com/pyspark-programming.

Veljanoski, J. "https://towardsdatascience.com/dask-vs-vaex-a-qualitative-comparison-32e700e5f08b."