Bike Sharing Data Science Task Assignment

Rohit Chowdhary

1 Part I

1.1 Task Definition

Bike sharing systems are new generation of traditional bike rentals where the whole process from membership, rental and return has become automatic. Given the end-to-end metrics from Bike Sharing data-set, such as weather conditions, and holiday schedule, the task was to perform Exploratory Data Analysis (EDA) and build a prediction model for the hourly utilization count of this data.

1.2 Data Set Information

The data set used is from the University of California Irvine's Machine Learning Repository. The raw data is a csv file with information from 17, 379 hours over 731 days with 16 features (information categories) for each hour.

1.3 Implementation

1.3.1 Exploratory Data Analysis

This is a multi-step process that was performed on the data set to discover patterns, spot anomalies, and check assumptions with the help of summary statistics and graphical representations. The top 5 rows of the data are demonstrated in Figure 1 and subsequently, the categorical and

continuous features are described in Figures 2 and 3 respectively. The data neither contains duplicate nor missing values.

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Figure 1: First five rows of the data set based on position

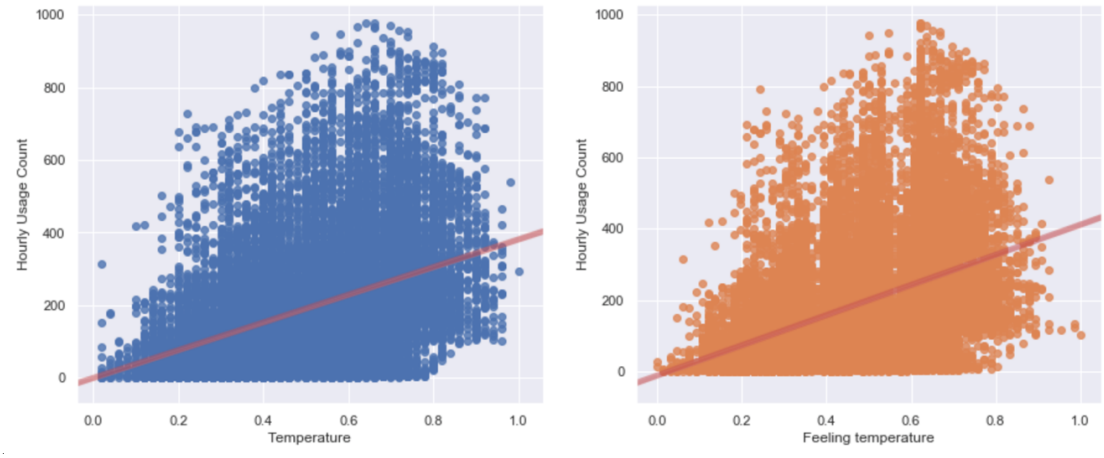

Figure 2: Scatter plot - Temperature vs Count

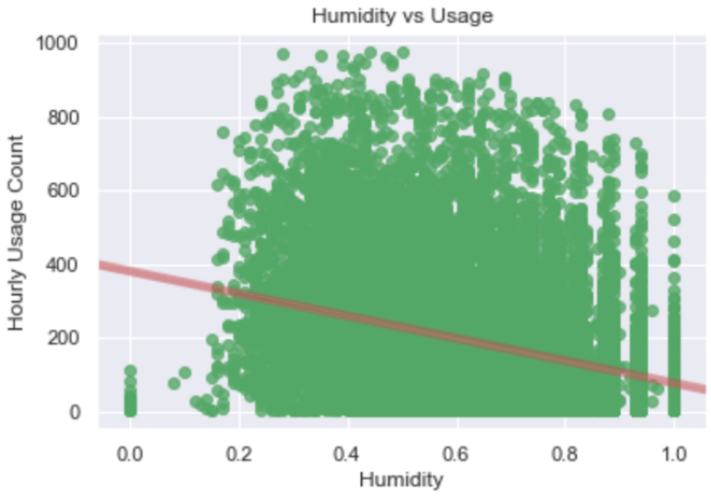

Figure 3: Scatter plot - Humidity vs Count

As seen from scatter plots in Figure 2, there is a positive correlation between both **temperature-to-usage** and feeling-temperature-to-usage, which makes sense as people are not likely to bike outside on cold weather. On the contrary, there seems to be a negative correlation between **humidity** and the usage count, with a linear fit very close to the best fit (see Figure 3). High humidity is directly related to high rainfall, so it could be hypothesized that weather situation will affect the usage, with high rainfall lowering the usage.

Scatter plot in Figure 4 support this hypothesis and indicates that bike usage significantly reduces in rainy weather (weather sit 4). It is also important to note that there are many more clear days than overcast or rainy days (weather sit 2 or 3), as illustrated by the histogram on the x-axis. Looking at the box plots (left) for each hour of the day, it is evident that the highest usage is during 7-9 am and 5-7 pm, which indicates that most of the users rent e-bikes to get to school or work, as illustrated in Figure 5. The box plot on the right reveals that the usage rate goes up from year 1 to year 2, which means the system grew in popularity over time.

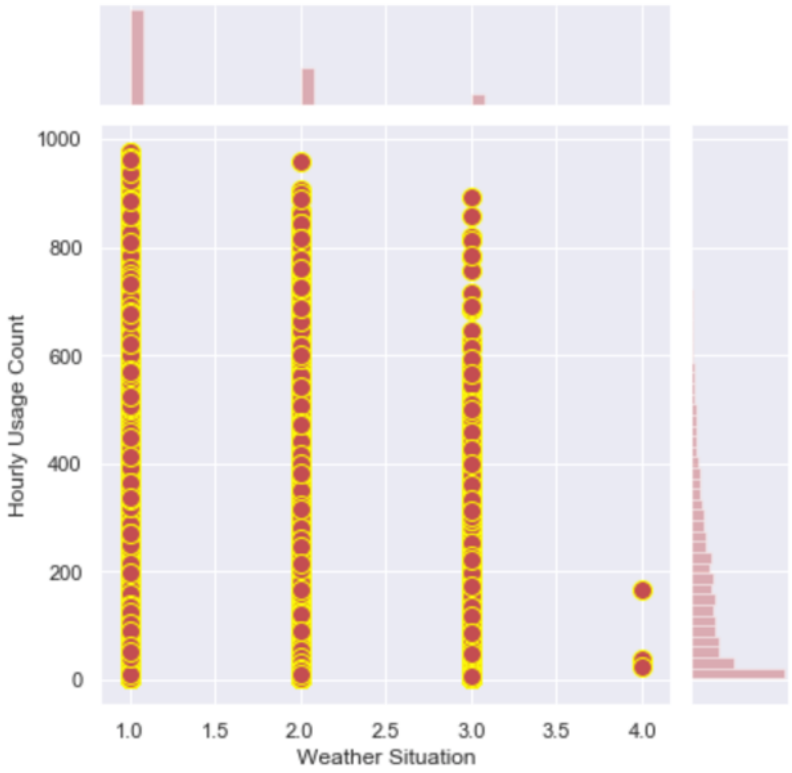

Figure 4: Marginal plot - Weather situation vs Count

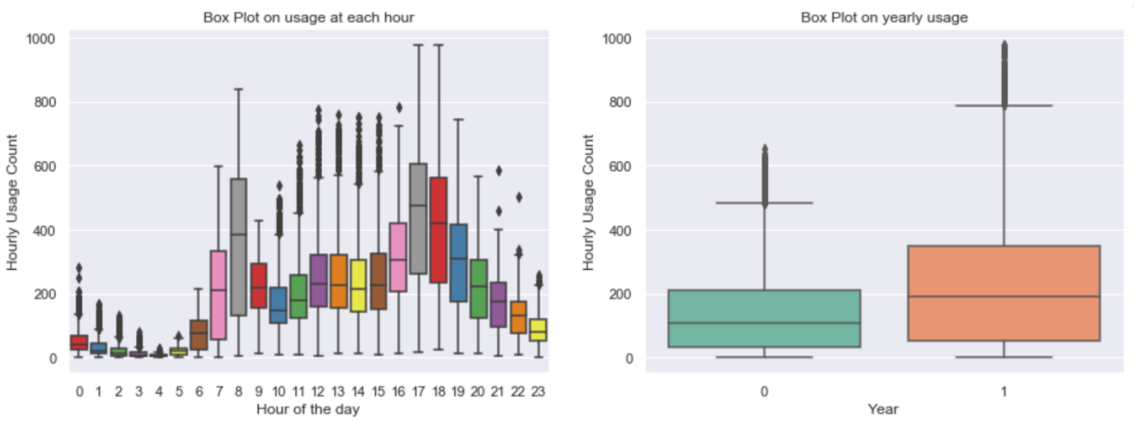

Figure 5: Box plots for different features

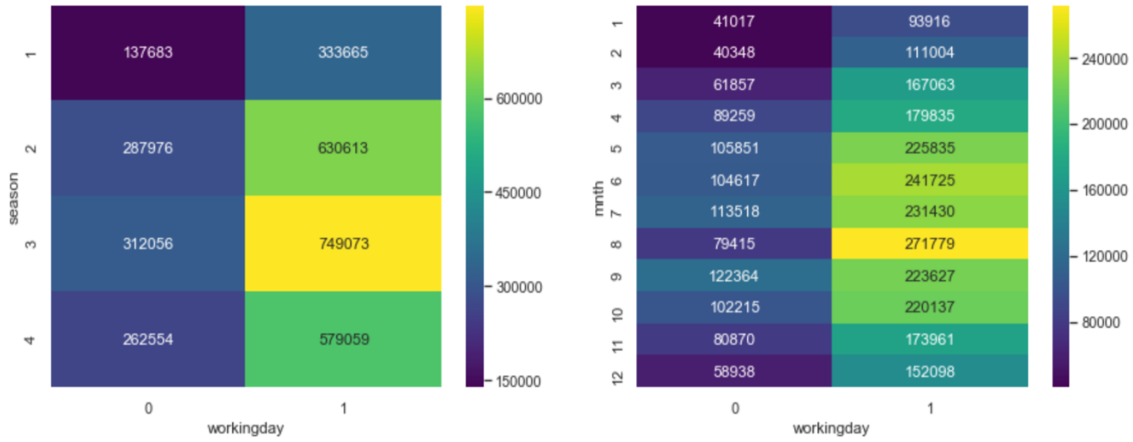

Figure 6: Heatmap - Correlation of different features with count

The above heatmap (left) demonstrates an interesting correlation with maximum usage during the summer season (3), and similar results can also be seen from the heatmap (right) with an evidently higher usage from May until October. Finally, the below correlation matrix also indicates the similar trends observed so far and shows a high degree of association between *temp* and *atemp*. Looking at the observations, it can be concluded that hour and temperature are promising features to predict hourly count value.

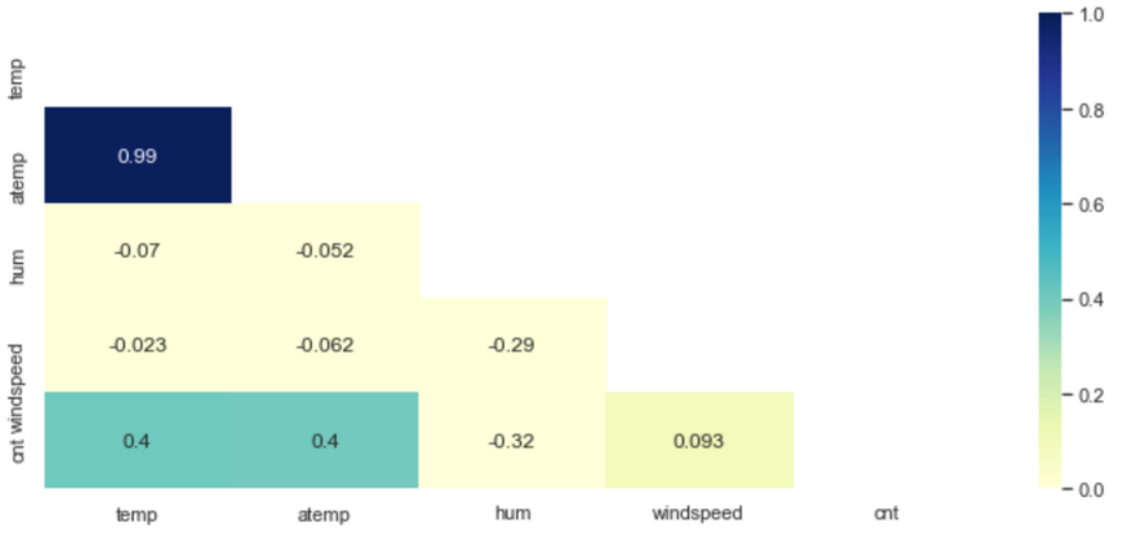

Figure 7: Correlation Matrix

1.3.2 Modelling

Model Selection: As the problem set has a continuous target variable, predictive analytics requires a regression ML algorithm. Another important characteristic for model selection is the size (in terms of total samples) of the data which in this case is less than 20k. Moreover, EDA showed that not all the features are significant for predicting the hourly usage count. Combining all these factors and task definition, I employed various regression models such as Linear Regression, Ridge, Lasso, Elastic Net, KNN, Decision Tree, and Random Forest and evaluated their performance.

Random Forest Model: Based on the evaluation of all the models, Random forest showed promising results on R2 and Mean absolute error (MAE). After fine-tuning the parameters on the validation sets, the final random forest model consists of 800 trees and gives MAE of 33.93 averaged over three splits using three-fold cross-validation.

The feature importances from the random forest are calculated based on the training set of the first split and are demonstrated below:

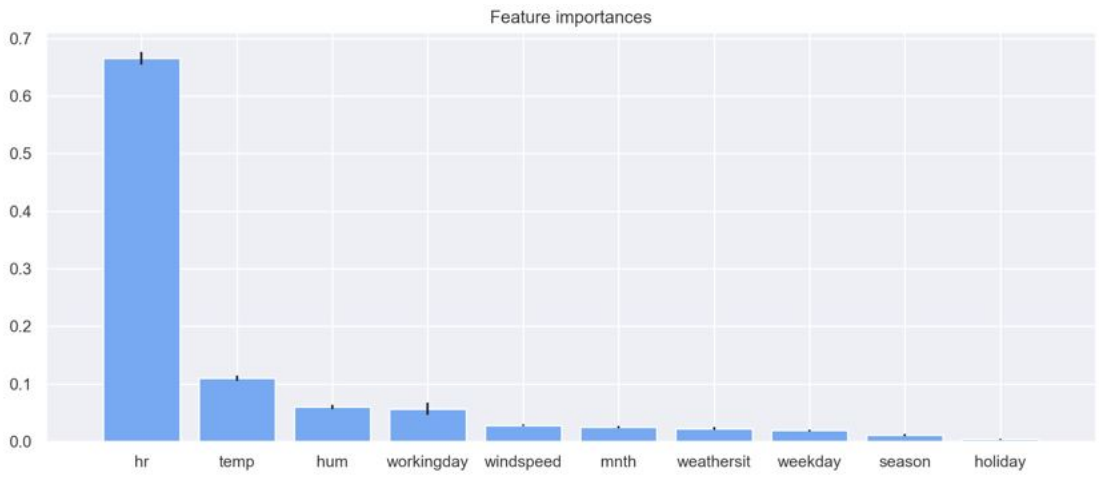

Figure 8: Feature importances

1.4 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?)

This means that the machine learning model we built is integrated into an existing production environment to make practical business decisions based on data. Deployment is one of the last phases of any data science or software engineering project life cycle, where the developed solution is moved to a server to be put in action and generate measurable business value. However, the ability to write a production-level code is one of the sought-after skills for a data scientist role and should include elements such as Modularity, Logging, Unit testing, Version control, and Readability (Docstring and Comments).

Machine Learning models are powerful tools to make predictions based on available data. To make these models useful, they need to be deployed so that others can easily access them through an API (application programming interface) to make data-driven decisions. One such deployment solution is to use Flask and Heroku - Flask is a powerful python microwebserver framework that allows us to build REST API based web-services and Heroku is a cloud platform that can host web applications. For this, the trained model is first stored as a pickle file which is a serialized format for storing objects.

Another scalable option is to deploy machine learning models as microservices in Docker which is a containerization platform that packages an application all its dependencies into a container. Kubernetes along with Docker provides an orchestration platform to automate, scale and manage multiple containerized applications.

1.5 Project Execution

The python code to reproduce the provided analysis, plots and models is provided including documentation and unittests. This guarantees the reliability of the code securely, facilitates maintenance and allows potential colleagues to extend the code.

Create a python environment using conda and the 'environment.yaml' file:

```
conda env create --file=environment.yaml
```

All analysis steps are provided in more detail in the 'Bike Sharing Data Science Task.ipynb' jupyter notebook. Run it with the following command from the source folder:

```
jupyter notebook
```

The random forest is implemented in the 'predict.py' file. Run it with the following command from the source folder:

```
python predict.py
```

To run the unit tests for the random forest code, you can run the unit_test detection of python:

```
python -m unit_test
```

2 Part II - Scalability

The implementation of random forest regression using sklearn will slow down the predictions when data goes up to several terabytes. Both the amount of data and number of trees in random forest take up memory and it may not even be able to read all the data into memory. So, in the worst case, it will crash the implementation and become non-functional for big data.

To address this problem, **Apache Spark** can be utilized to distribute both data storage and processing. Along with that, python provides a **wrapper-based construction framework** i.e. Woody for building large random forests for hundreds of millions of training instances.

The key idea in Woody is to use top trees to partition all the available training data into smaller subsets associated with the top trees' leaves and to build bottom trees for these subsets (Gieseke & Igel, 2018). On the other hand, Apache Spark is a powerful unified analytics engine for large-scale distributed data processing and machine learning. Using PySpark, one can take advantage of working with MLib, which is a machine learning API from Spark for faster computation. Spark ML can run on Hadoop, Apache Mesos, Kubernetes, etc., and can access data from both SQL and NoSQL databases such as MySQL, MongoDB, etc.

Nonetheless, Sparks' in-memory capability can become a bottleneck as it requires a lot of RAM to run in-memory. Notwithstanding the fact that Spark provides a plethora of computational power, it is not cost-efficient in all the use cases. Also, the runtime of the pre-classification could slow down the data distribution.

I have hands-on experience of working with Docker, MongoDB, and Apache Spark. In my previous experience, I integrated all these technologies to implement a Self-Service analytics solution in a production environment. Apart from that, I worked on the development of a mini-digital factory to process real-time data by integrating Apache Kafka and Spark. During my academia, I developed a prediction model in the Hadoop ecosystem i.e. HDFS and MapReduce.

References

Gieseke, F., & Igel, C. (2018, February). Training Big Random Forests with Little Resources. *arXiv:1802.06394 [cs, stat]*. Retrieved 2019-12-28, from http://arxiv.org/abs/1802.06394 (arXiv: 1802.06394)