Project: Bike-Sharing Demand

Sebastian Castillo-Sanchez

Technology Fundamentals for Business Analytics

<https://www.kaggle.com/c/bike-sharing-demand>

# 1. Executive Summary

**Problem**

Bike-sharing is a growing and popular transformation in the transportation industry in this day of age. This system provides people the opportunity to travel around a city or town in a bicycle through renting through a membership or for a short period of time. As a result of rising popularity, there are currently 500 bike-sharing systems around the world. The main process of this system is the person is able to ride the bicycle anywhere and return it to the same location from where they rented the bicycle or to a different location where they also accept the bicycle. From there it then creates a way for people who are visiting a location for the first time or need a quick mode of transportation to get to a destination in a very efficient way.

**Data**

The data provided by Kaggle allows competitors to research the data to predict the bike sharing demand in Washington, D.C through different types of data. The data is very popular because bike-sharing can be seen as a sensor network which can further be analysed to forecast movement and transportation in a city. The data in its entirety allows competitors to generate bike-sharing rental predictions using weather and time data such as “holiday” and “humidity”. After running through all the fields, most of the data are integer/float types while there is one object type for the “datetime” field. Through visualizations and basic statistics using Python, I was able to view the data in a helpful way to allow me to continue my analyzation. Using the Seaborn package, I was able to create some plots that showed me what was the normal aspects of the data and what needed to be looked into deeper. In addition, incorporating Feature Creation allows to create new features based on the existing data fields as well as improving predictions in the end.

**Findings**

Through the process viewing other Kernel solutions from Kaggle as well as performing my own Modeling, there was a lot of very interesting information and predictions. **(Will be finished for final draft)**

# 

# 2. Data Description and Initial Processing

**Data Description**

These are all the data fields from: <https://www.kaggle.com/c/bike-sharing-demand/data>

|  |  |
| --- | --- |
| **Fields** | **Values** |
| datetime | object : hourly date + timestamp |
| season | int : 1 = spring, 2 = summer, 3 = fall, 4 = winter |
| holiday | int: whether the day is considered a holiday |
| workingday | int: whether the day is neither a weekend nor holiday |
| weather | int: 1: Clear, Few clouds, Partly cloudy, Partly cloudy  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog |
| temp | float: temperature in Celsius |
| atemp | float: “feels like” temperature in Celsius |
| humidity | int: relative humidity |
| windspeed | float: wind speed |
| casual | int: number of non-registered user rentals initiated |
| registered | int: number of registered user rentals initiated |
| count | int: number of total rentals |

After viewing the basic data fields from the data set, I then focused on viewing the real values from the training/testing data as well as their statistical values. The data provided spanned two years of data collection. The training set contained the first 19 days of each month while the testing set contained the 20th day to the end of each month. The purpose of this competition was to predict the total count of bike rentals for each hour in the testing set. The first thing in the EDA process was to figure out if any Data Imputation was needed to clean the data. However, there was no null values in the training data so there was no need to remove any part of the data set. Continuing onto more statistical analyses, below are two tables that can provide more information about the weather and user count for the system.

**Mean of Temperature fields by Season:**

|  |  |  |  |
| --- | --- | --- | --- |
| **season** | **weather** | **temp** | **humidity** |
| 1 | 1.424423 | 12.530491 | 56.297841 |
| 2 | 1.422978 | 22.823483 | 60.852909 |
| 3 | 1.366630 | 28.789111 | 64.123674 |
| 4 | 1.459766 | 16.649239 | 66.173738 |

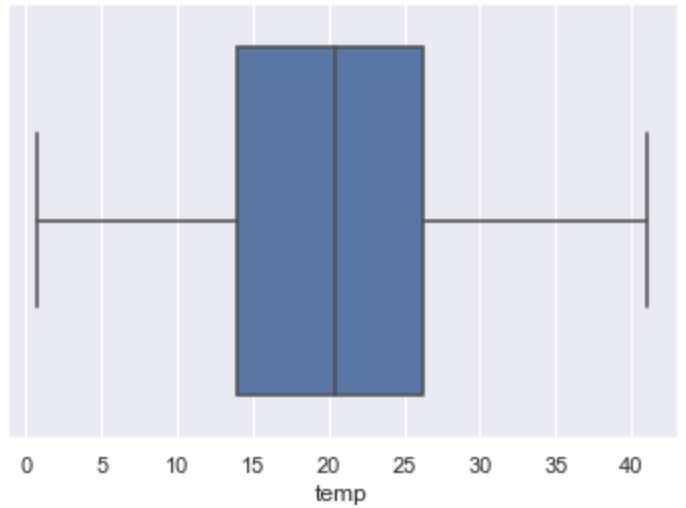
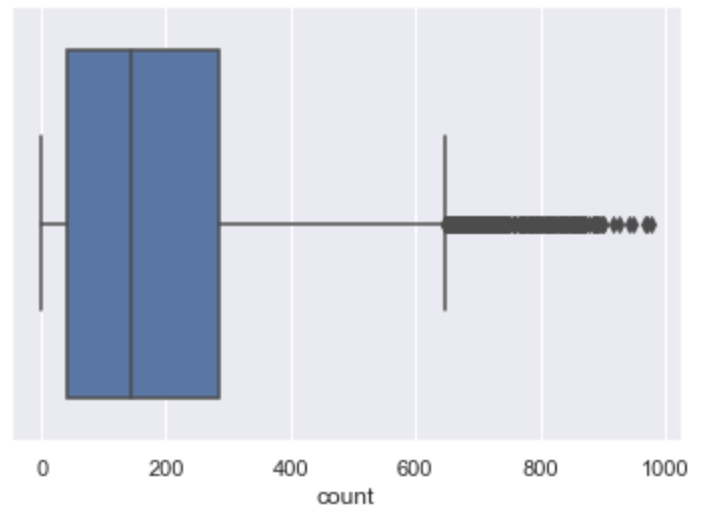
**Total of Registered Users vs. Casual Users by Season:**

|  |  |  |
| --- | --- | --- |
| **season** | **registered** | **casual** |
| 1 | 270893 | 41605 |
| 2 | 458610 | 129672 |
| 3 | 497944 | 142718 |
| 4 | 465894 | 78140 |

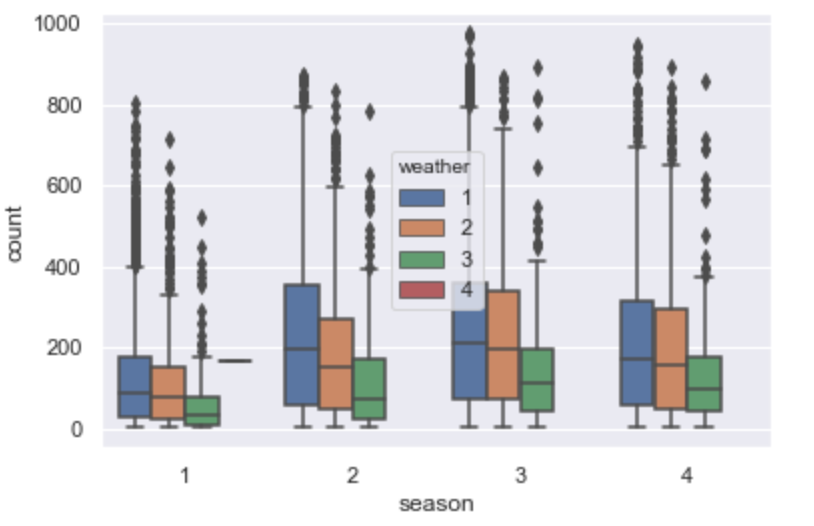
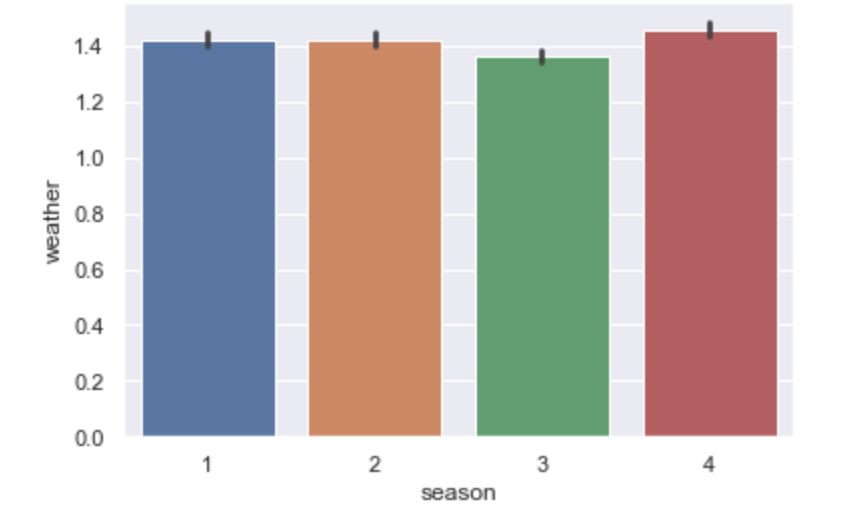
Using these two tables, we can make the conclusion that more people were registered when using this system but there was a fair amount of casual users as well. In addition, the weather shows a similar variation of each field for each season. It can also be seen that season 1 (spring) had the least count total of users both registered and casual which is interesting because I would believe that spring would have a higher user count.

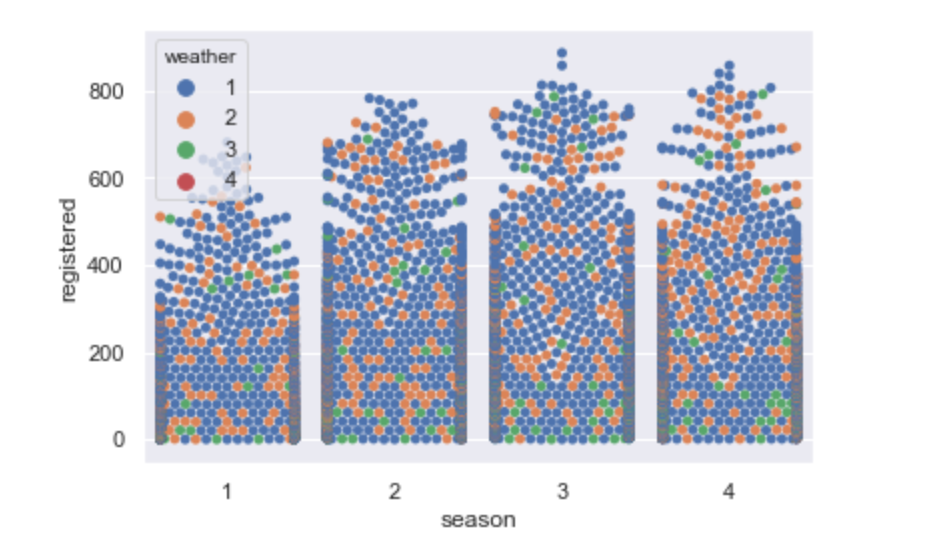
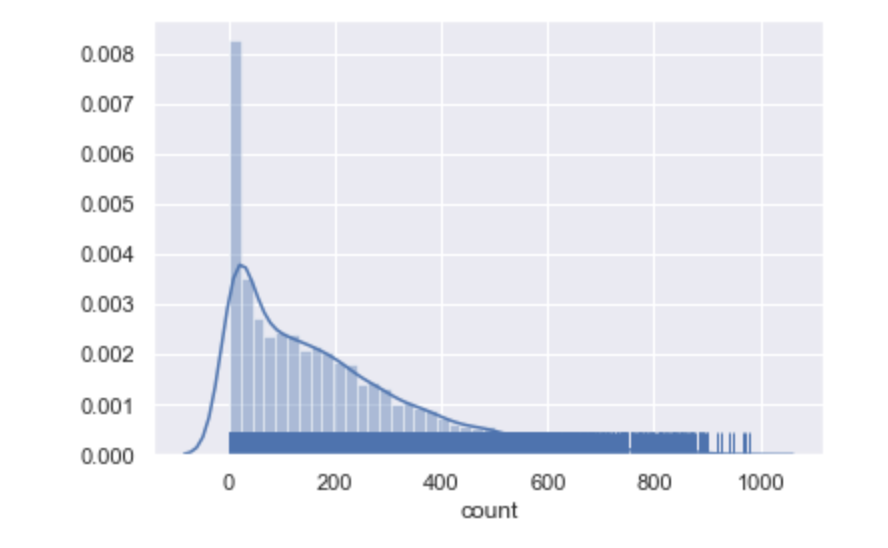
**Visualizations**

Presented below are some visualizations using the Seaborn package



Above are two Boxplots that allow to visualize the data as well as possibly view the outliers in the data. In the first one, you can see that the temperature for the entire data set was mostly in the 15-25 degrees Celsius range while there was a minimum of about 1 and a maximum of about 42. The second boxplot is for the count data field which shows the range of how many people used the system. It can be seen that the 50-300 range was the most prevalent while a minimum of 0 and a maximum of about 650. Fortunately, we can see some outliers in this plot which shows values higher than the maximum which there happens to be a good amount of.

Continuing on, here are more diverse one that combines season, count and weather. The left is a more unique box plot that shows that most the count for each season is for the 1 and 2 weather patterns while there is no data to be seen for the 4 weather pattern which is “Heavy Rain, Thunderstorm and Hail”. The barplot on the right is similar in looking at the mean of each of the weather patterns per season which seemed pretty similar in all seasons for the most part.



The last couple of visualizations are the most interesting because they show very unique data in several different plots. The distribution plot is for the count data field and also displays the KDE, Rug and Histogram. This shows the variance and distribution of the count of users in the system. It can be seen that most of the distribution is below 400 which is expected from the data. The Swarmplot on the right draws a categorical scatterplot using non-overlapping points. In this plot, we are visualizing the registered users depending on the season and the weather. There seems to be a greater amount of points in the 1-2 weather range that can be seen as better “biking weather”.

The next step is the Feature Engineering/Creation which allows to create new features from already created data fields that will allow for better predictions and visualizations before performing modeling. This will be explained in further detail in Modeling (Section 4)

For more visualizations and respective code as well as the Feature Engineering, view the “project.ipynb” Jupyter Notebook in the GitHub Repository.

# 3. Modeling & Evaluation of 3 other solutions

After performing my own statistical analyses and visualizations on the training data, now there are several solutions on the Kaggle competition that can be seen and evaluated.

**EDA & ML on bike-sharing** (<https://www.kaggle.com/fredkron/eda-ml-on-bike-sharing>)

This first solution provided EDA that focused on the total count rather than dividing up based on registered and casual users. They then went forward to split the datetime field into specific feature fields such as month, week and day. A very interesting approach to the data is that they split the analysis for each year (2011,2012) which allows more specific data being presented. They did this for every data field and then presented visualization to show the difference for each field depending on the year the data was collected. Another new feature that was created was Temp by weather which calculated the temperature by the weather value

Modeling-wise, they took their approach to split the testing and training data as should be done before performing any modeling. They then decided to test the data using several different models which were:

* Decision Tree
* Linear Regression
* Lasso
* KNeighbors
* Gradient Boost
* SVR
* XGBRegressor

As a result, the best score (through cross validation scoring) for the data set was from the XGBRegressor which they then used for the testing set. They fitted the data and then created the prediction which was then used for the submission file for the competition with a final score of 0.40231.

**EDA & Ensemble Model (Top 10 Percentile) (**<https://www.kaggle.com/viveksrinivasan/eda-ensemble-model-top-10-percentile>**)**

This solution followed EDA similar to the last solution where they provided some Feature Engineering to create new features from the datetime and then create features from already created data fields. After some visualizations, they provided some analyses for Missing values as well as some outliers to ensure the best prediction for the data set. They did this through the “missingno” package and through Seaborn visualizations.

Modeling-wise, they started to build their models by first using Random Forest to fill in 0 values for the Wind Speed column which I believe was an interesting way to improve predictions. They then proceeded to split the train and test data before moving onto the actual modeling. The different types of modeling they used were:

* Linear Regression
* Lasso
* Ridge
* Random Forest
* Gradient Boost

They provided a function that would compute the RMSLE score for each model. For each type of model, they fit and predicted the model and then produced the score. As a result, the best (no major overfitting) model was the Gradient Boost ensemble model with a score of 0.41

**BIKE SHARING DEMAND [RSMLE:: 0.3194]** (<https://www.kaggle.com/rajmehra03/bike-sharing-demand-rmsle-0-3194>)

The third and final solution that I viewed was structured similar to the past two solutions but at the end provided the lowest (and therefore best) RMSLE. After visualizations and missingno analysis, they moved forward to Feature Engineering by creating dummies for the weather and season variables for deeper analysis. Like the other two solutions, they created specific date field features while dropping the necessary fields since they are not needed anymore. One interesting thing they did with the date features was to visualize the count of bike rentals depending on the hour of the day which is something I will look into more when doing my own Feature Engineering.

Modeling-wise, they first split the training and testing data to begin the modeling process. The models they then chose to use were:

* Random Forest
* AdaBoost
* Bagging
* SVR
* KNN

After fitting and predicting the models for each of the types, they proceeded to calculate the RMSLE which resulted in Random Forest Regressor model being the lowest with a score of 0.3194

**Final Modeling Result Table for 3 solutions:**

|  |  |  |
| --- | --- | --- |
| **Solution** | **Modeling** | **RMSLE (Performance)** |
| EDA & ML on bike-sharing | XGBoost  (XGBRegressor) | 0.40231 |
| EDA & Ensemble Model (Top 10 Percentile) | Gradient Boost  (GradientBoostingRegressor) | 0.41 |
| BIKE SHARING DEMAND [RSMLE:: 0.3194] | Random Forest  (RandomForestRegressor) | 0.3194 |

In conclusion, as you can see from the results table, the last solution received the lowest RMSLE with a Random Forest Regression model. However, the other two solutions were also low with their own specific models and presented very unique evaluations of the data set.

# 4. Modeling

# 5. Summary

# 6. Appendix

**GitHub:**

<https://github.com/scastillosanchez/TechFundamentalsProject>

**Resources:**

* **Seaborn Documentation**
* **Class Notes/Exercises**
* **Kaggle Competition**