**Part 1 – Data analysis and predictive model**

The following steps were performed to analyze the Bike Sharing Dataset and build a predictive model:

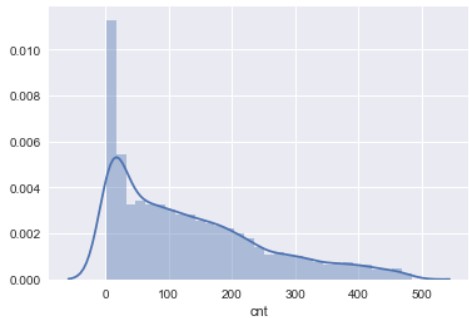
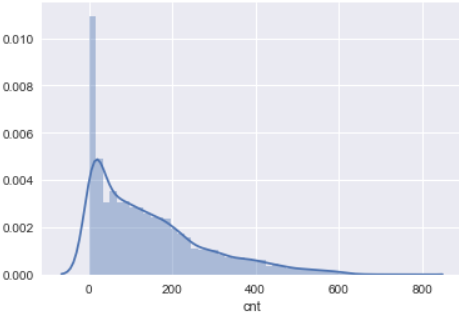
* Descriptive Analysis
* Missing Value Analysis
* Outlier Analysis
* Correlation Analysis
* Model selection
* Random Forest Training and Feature Ranking

1 – Business Case

The hourly prediction of the bike sharing count value is not only important to estimate the expected revenue of the bike sharing service, but also to provide the required amount of bicycles at each station of a distributed bike sharing service. With more information about different stations, you could predict and schedule the rebalance of given back bikes. In this task, I investigate the prediction of the hourly bike count based on the specific hour, expected weather and day information.

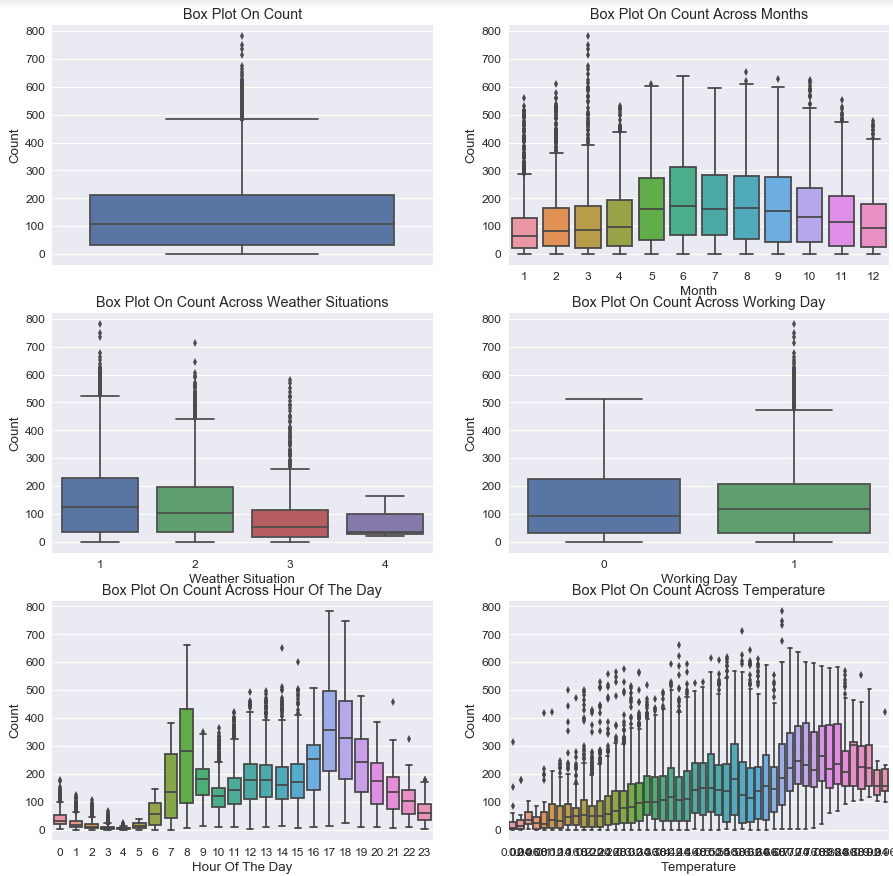
2 – Analysis and data preparation

The missing value analysis revealed the data set does not contain any not-a-number or null values which require a replacement for further processing. In the following step, the outliers of the count values were removed using median and interquartile range (IQR)  as the count values do not fit a normal distribution. This reduces the data set from 15641 to 15179 samples.



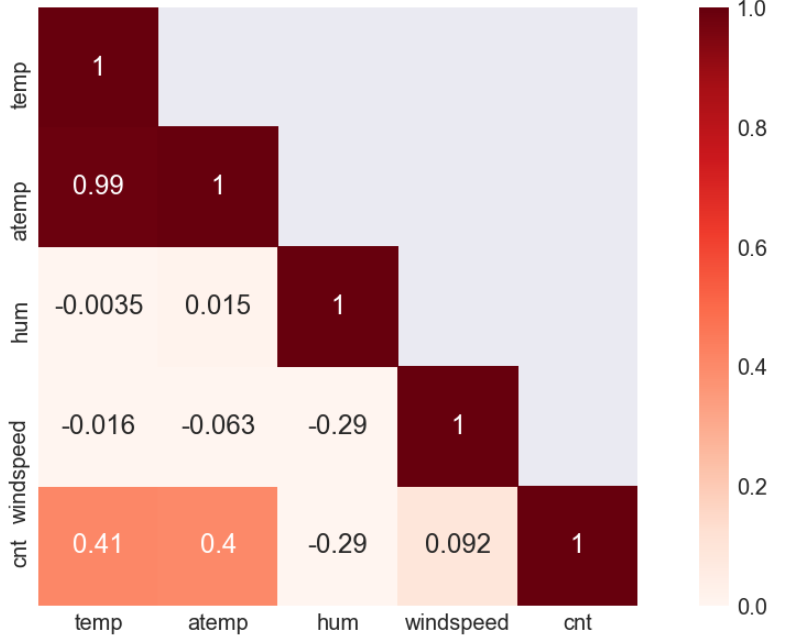
Data with outliers

Data without outliers



Box plots for different feature variables( before outlier removal).

The box plots and the correlation matrix of the numerical features revealed that the hour and temperature are some promising feature variables to predict the hourly count value. The correlation analysis also revealed that temperature and feeling temperature are highly correlated. To reduce the model complexity and avoid collinearity, the feeling temperature features were dismissed.



Correlation matrix for numerical features.

3 – Model selection

The prediction of the count values requires a regression algorithm based on categorical and numerical features. The dataset is quite small with less than 20K samples and the analysis steps revealed that a few features could be particularly significant. Based on these characteristics of the task and the data, I evaluated a set of possible algorithms: Lasso, Elastic Net, Support Vector Regression with different kernels, Ridge Regression and Random Forests.

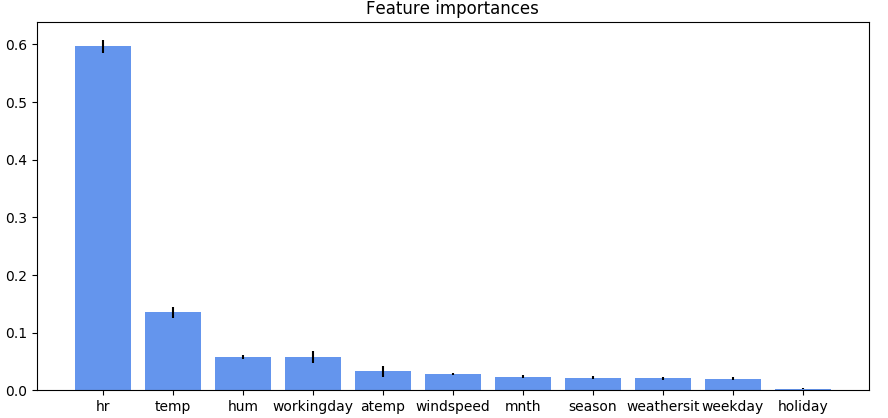
4 – Random Forest

The random forests showed the most promising results on the Bike Sharing Dataset and were picked for the final result. The final random forest model consists of 200 decision trees trained on various-subsamples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. An internal needs at least four samples to split and the mean squared error was used to estimate the quality of a split.

The final model receives a mean absolute error of 44.30 averaged over three splits using three-fold cross-validation:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Split | Mean Squared Error | Mean Absolute Error | RMSLE | R² Score |
| RandomForestRegressor | 1 | 4489.95 | 43.72 | 0.41 | 0.86 |
| RandomForestRegressor | 2 | 4636.60 | 44.60 | 0.41 | 0.86 |
| RandomForestRegressor | 3 | 4691.67 | 44.57 | 0.41 | 0.86 |
| **RandomForestRegressor** | **Mean** | **4606.07** | **44.30** | **0.41** | **0.86** |

Feature ranking of the decision trees in the random forest regressor on the training samples of the first split:



5 – Code

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 ‘env.yaml’ file:

conda env create --file=env.yaml

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

jupyter notebook

The random forest is implemented in the ‘random\_forest.py’ file. Run it with the following command from the source folder:

python random\_forest.py

To run the unittests for the random forest code, you can run the unittest detection of python:

python -m unittest

**Part 2 – Large-scale dataset**

**The used sklearn python implementation of random forest regression will extremely slow down for large scale datasets (> 10 Mio. samples). This is caused by costly computations and because the data can not completely be stored in the main memory anymore. In the worst case, the sklearn implementation will crash and is not usable for large-scale datasets.**

**A good solution would be the combination of state-of-the-art top tree pre-classification and distributed computation e.g. with Apache Spark ML.**

**The python woody implementation uses top-trees for a coarse pre-classification and distributes the samples to bottom random forests which are implemented in pure C and highly optimized. In a direct comparison (see arxiv-paper) woody shows good results in runtime performance and accuracy. Apache Spark ML is a machine learning framework highly optimized for distributed computation and would allow the utilization of a computation cluster. Spark ML can run on Hadoop, Apache Mesos, Kubernetes, etc. , can access data from popular Apache databases like Apache Cassandra and claims to be 100 times faster than classic algorithms.**

**One idea could be the distribution of data with coarse top-tree pre-classification on distributed machines and to learn the bottom random forests on these machines. Such a solution makes only sense if a large random forest is required to model the underlying data properly. The success of this solution depends on the following factors:**

* **Pre-classification runtime: The runtime of the pre-classification could slow down the data distribution.**
* **Network delay: Especially in a computation cluster the network could be the bottleneck and slow down the computation.**
* **Hardware: A computation cluster provides a multitude of computational power, but is not affordable in every use case.**

**I have theoretical knowledge about distributed computation and cluster architectures. I have no hands-on experience with frameworks like Hadoop or Apache Spark ML so far.**