SeoulBikeData Report

**Dataset**

The dataset examined in this report was obtained from [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand). It contains 8760 instances and 14 attributes.

* Input Variables:
  + 1 - Date: year-month-day
  + 3 - Hour: Hour of the Day
  + 4 - Temperature: Celsius
  + 5 - Humidity: Percent
  + 6 - Wind Speed: m/s
  + 7 - Visibility: 10 m
  + 8 - Dew point Temperature: Celsius
  + 9 - Solar Radiation: MJ/m2
  + 10 - Rainfall: mm
  + 11 - Snowfall: cm
  + 12 - Seasons: Winter, Spring, Summer Autumn
  + 13 - Holiday: Holiday or No Holiday
  + 14 - Functional Day: NoFunc or Fun
* Response Variable:
  + 2 - Rented Bike Count: Count of Bikes rented at each hour

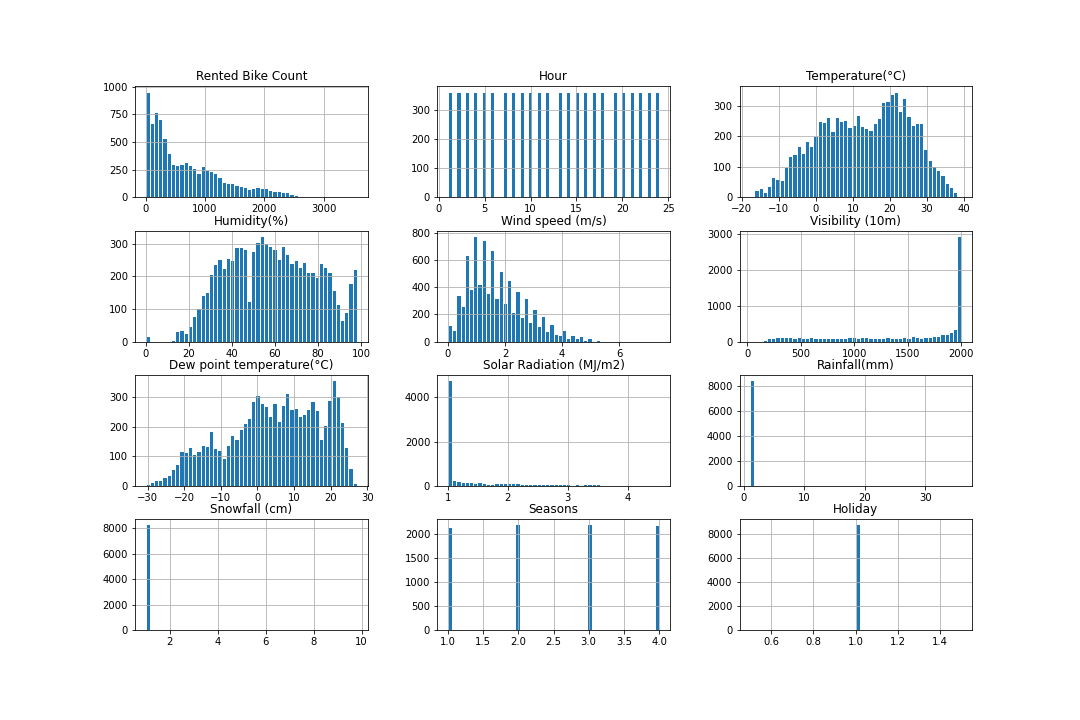
The input variable of “Functional Day” was eliminated due to every instance being ‘Fun’ and “Date” was eliminated.

**Exploratory Data Analysis**

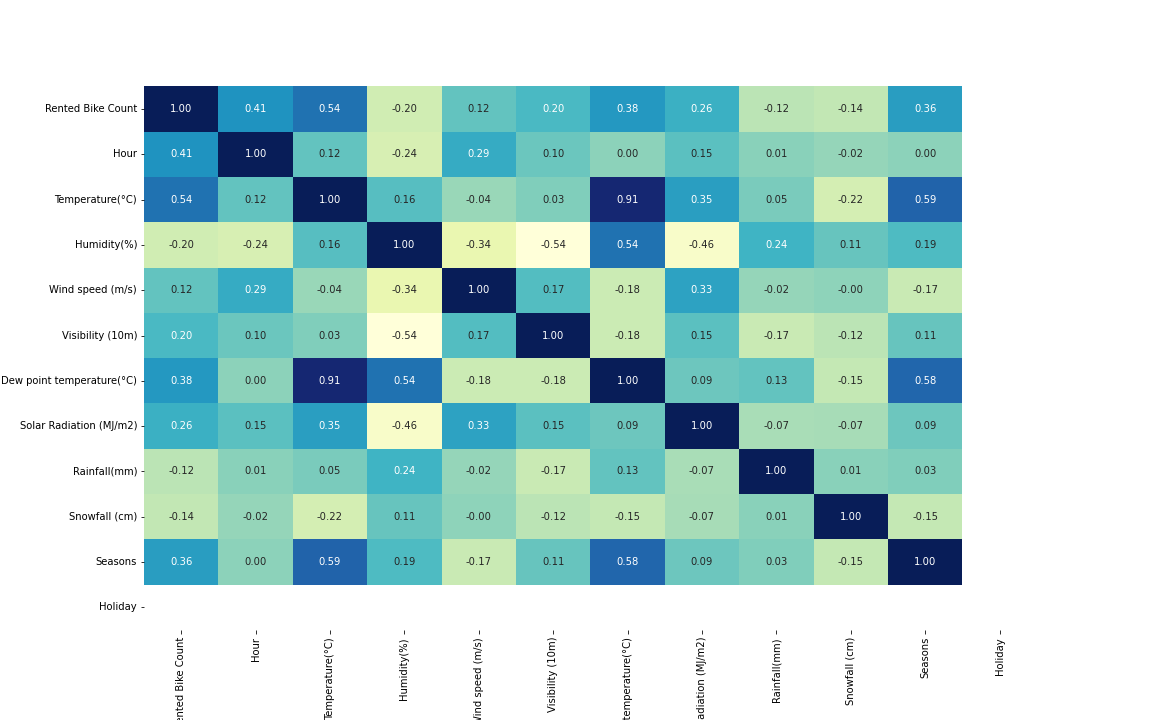
Table 1. Summary Statistics of SeoulBikeData.csv.

|  | **Rented Bike Count** | **Hour** | **Temperature** | **Humidity** | **Wind speed (m/s)** | **Visibility** | **Dew point temperature** | **Solar Radiation** | **Rainfall(mm)** | **Snowfall** | **Season** | **Holiday** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **mean** | 704.602055 | 12.50 | 12.88292 | 58.226 | 1.724 | 1436.82 | 4.073 | 1.569111 | 1.148687 | 1.075068 | 2.504110 | 1.0 |
| **std** | 644.997468 | 6.922582 | 11.944825 | 20.362413 | 1.036300 | 608.298712 | 13.060369 | 0.868746 | 1.128193 | 0.436746 | 1.114408 | 0.0 |
| **min** | 0 | 1.000 | -17.8000 | 0 | 0 | 27.00 | -30.6 | 1.00 | 1.00 | 1.00 | 1.00 | 1.0 |
| **25%** | 191.000000 | 6.750 | 3.500000 | 42.000 | 0.900 | 940. | -4.70 | 1.00 | 1.000 | 1.0000 | 2.00 | 1.0 |
| **50%** | 504.500000 | 12.50 | 13.70000 | 57.000 | 1.500 | 1698. | 5.100 | 1.01 | 1.000 | 1.0000 | 3.00 | 1.0 |
| **75%** | 1065.250000 | 18.25 | 22.50000 | 74.000 | 2.300 | 2000. | 14.80 | 1.93 | 1.000 | 1.0000 | 3.00 | 1.0 |
| **max** | 3556.000000 | 24.00 | 39.4000 | 98.000 | 7.400 | 2000. | 27.20 | 4.52 | 36.00 | 9.8000 | 4.00 | 1.0 |

Table 1 displays basic statistics over the dataset. No missing data points or null features were found. For “Seasons” variables, each season was given a value (1,2,3,4) for Winter, Spring, Summer, Autumn. For “Holiday”, the no or yes response was changed to 1 and 2 respectively. As well “Hour”, “Solar Radiation”, “Rainfall” and “Snowfall” all had a numerical one added to each instance in order to eliminate zeros for proper use of some of the models.



**Figure 1**: Histogram plots of the distribution of each parameter against the response variable.



**Figure 2**: Correlation ‘heatmap’ of the SeoulBikeData.csv data file.

The histograms in Figure 1 allowed for preliminary insight on what possible variables would impact the response variable the most. It seems, “Temperature”, “Dew Point” and “Humidity” show a close resemblance to a uniform distribution while the other variables are either multi-modal due to use of ordinal values or are skewed right like in “Wind Speed”.

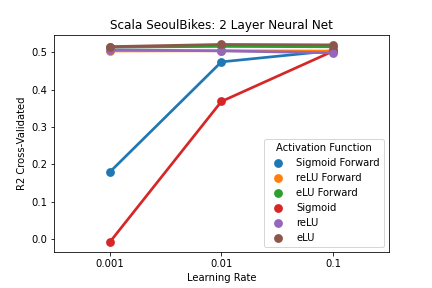
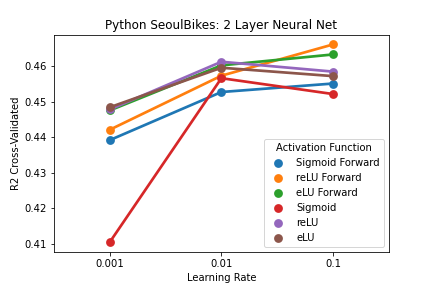
Moreover, the correlation ‘heatmap’ displayed in Figure 2 gave insights into collinearity in the data set. There are several moderately strong positive correlations between two predictor variables, such as “Humidity” and “Dew Point Temperature”. The only probable significant instance for multicollinearity is between “Temperature” and “Dew Point Temperature” which is logical due to “Dew Point Temperature” being a function of temperature itself. These observations, along with the variables that have strong positive correlations with the response variable, give insight on how feature selection algorithms used on the dataset will perform as well as how the models themselves will perform.

**Proposed Differences in Feature Selection**

The crux of the analysis on the neural nets used with this dataset do not only lie in comparing different activation functions and learning rates but as in comparing if the usage of a feature selection prior to the neural net would have any impact on the results obtained as the neural itself also utilizes some feature selection method. For the analysis done, forward feature selection was chosen and the hypothesis that these forward feature selected models would perform better was made due to the better understanding of how the feature selection algorithm works compared to the “black box” of what occurs inside a neural net. When forward feature selection was used, the features given to the neural net were that of “Temperature”, “Visibility”, “Rainfall”, “Snowfall” and “Holiday” in order to predict “Rented Bike Count”.

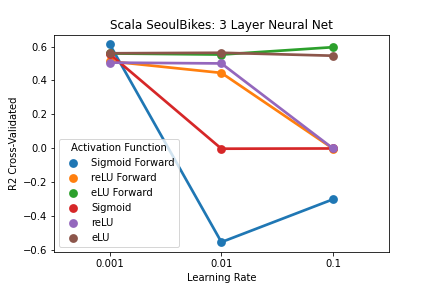
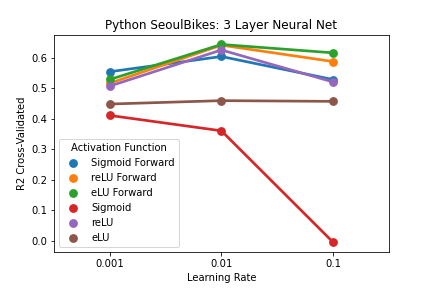
**Results**

The full results (R2 adjusted, R2 cross validated, feature selection and final MSE) can be found in the accompanied summary file.



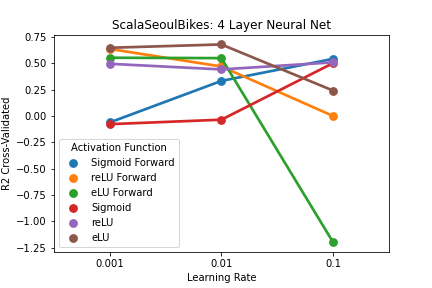
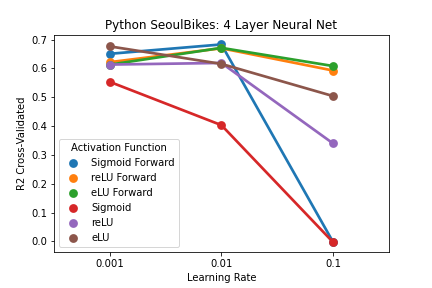
**Figure 3**: R2 cross validated values for the 2 Layer models in both Python and Scala

Comparing the R2cv values in both languages, the Scala generally performed better with most of the activation functions across both usage of forward selection and at the tested learning rates performing almost the same. As well, in both languages, the sigmoid activation function performed the worst. Overall, at the 2 layer model, the conclusion that the learning rates tested do not provide a significant difference in the R2 received nor the activation function used (with the expectation of sigmoid).

****

**Figure 4**: R2 cross validated values for the 3 Layer models in both Python and Scala

Like in the two layer models, the sigmoid function seemed to provide the worst R2 values. Interestingly, in both languages, the models in which forward feature selection was used outperformed the direct feeding of the data to the neural net. Moreover, eLU activation function performed the best in both languages. The usage of 0.01 learning rate seemed to be the most optimal. In regards to the acquisition of the zero and negative R2s with the sigmoid function, it can be inferred that at too high of a learning rate, the neural net is not able to viable update the parameters and just fails to predict the output.



**Figure 5**: R2 cross validated values for the 4 Layer models in both Python and Scala.

The similar trend of the forward feature selected models outperforming the regular neural net models continues at the four layer level. In the Python code, the sigmoid function with forward feature selection at 0.01 slightly outperforms the eLU function with forward feature selection. In the Scala code, the eLU function performs significantly more than any of the other activation functions. Both the sigmoid functions in the Scala code go against the trend that the higher learning rate usually correlates with lower R2, with both gradually increasing as the learning rate increases.

**Discussion**

In both languages, it was illustrated that the more layers the neural nets had, the higher R2 value obtained. This was an expected result as at a higher number of layers more updates, or rather more appropriate updates, between each epoch can be made which leads to better predicting of the response variable. As well, the proposed idea that the use of forward selection prior to the input into the neural net would lead to better performance, mostly holds true. Understanding why this is the case would prompt more investigation into how the neural net itself updates parameters and chooses to drop or add features as well. If more time was given for the project, more testing into different feature selections prior to the neural net could be done to see how different modes of feature selection compare to that of the neural net.

Overall, across all the models, the eLU activation function and the 0.01 learning rate seem to give optimal prediction power despite the number of layers in the neural net. Despite that, the four layer model did perform the best as expected.