Datasets Report

May 2020

1 Bike Sharing

The dataset is composed of two parts, daily and hourly tables. The target column is numerical (regression problem), indicating how many bikes were rented at each day/hour ('cnt' column). In order to understand whether the daily data provides any extra information that doesn't exist in the hourly data, an initial check was made to see whether the daily table is just an aggregation of the hourly table - it was found to be true. Therefore we continued with the hourly data only, which contains approximately 17k entries.

The focus from now on was to perform a manual feature-selection and combine it with an out-of-the-box regression model (RandomForestRegressor), mainly to show that a slightly more careful examination of the data could result in better feature-selection compared to existing methods like Best-K with varying scoring functions. First, Pearson correlation were plotted between each feature and the target (Fig1).

Most notable are the "registered" and "casual" features, that according to the dataset description, represent the number of bikes rented by registered and unregistered users. Both of these features should be omitted as they certainly cause target leakage - they won't be available in prediction time.

From here on we want to pick several interesting features, while trying to keep the number of features rather low due to the magnitude of the dataset (and the fact that it was run from a rubbish laptop).

We generated a scatter pairplot of the top-5 correlating features, to get some sense of how are they connected together (Fig2):

while, 'temp' and 'atemp' came up as good correlators, it is evident that they are both correlate almost perfectly between themselves. It is reasonable to assume that people tend to rent bikes based on the "felt" temperature (represented by 'atemp') rather than the actual reported one - to which people are usually unaware. Hence we chose to omit 'temp'.

Still weather-wise, 'humidity' also came up as a decent correlators to target. 'season' and 'weathersit' both represent a similar and more general aspect of the weather, yet their correlation is not so indicative because they are simply encoded to integers. We therefore inspected the standard deviation of their target mean value per category. 'weathersit' presented slightly higher standard

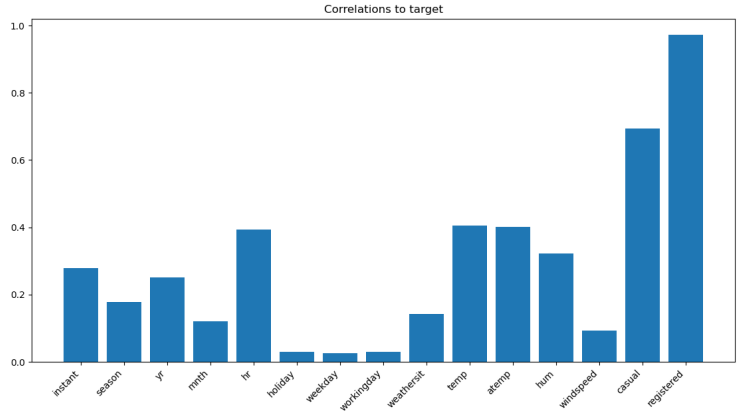

Figure 1: Fig1

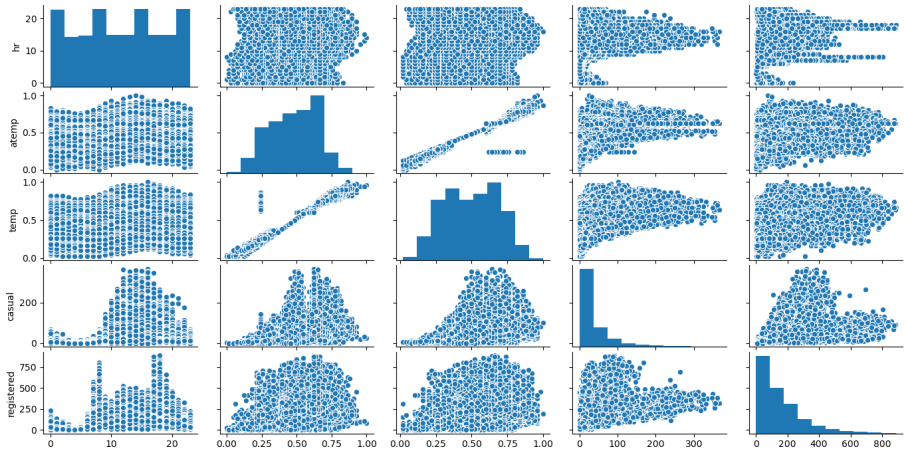

Figure 2: Fig2

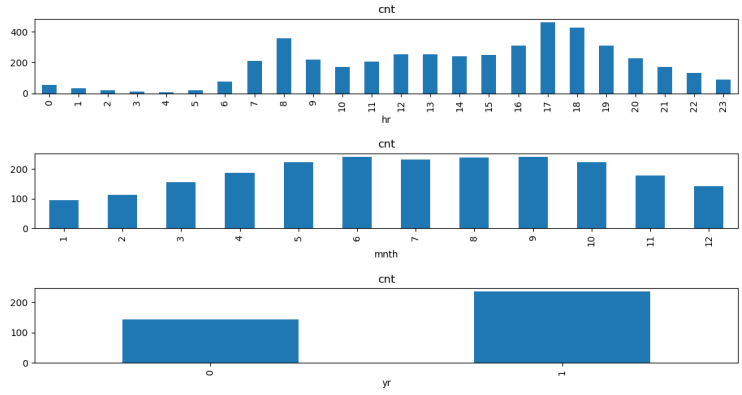

Figure 3: Fig3

deviations, and hence could be more relevant. People usually don't care of formal seasons (especially when considering season changes - to which people are unaware most of the time) but rather maintain a more vague idea of the weather that better matches 'weathersit' feature. Hence we omit 'season' and keep 'weathersit'. 'windspeed' was also omitted as we would expect it to present good inverse correlation to target had it been an indicative feature - higher winds speeds would decrease the number of rented bikes.

Turning to time based features which presented some decent correlation such as 'year', 'month', 'hour' and 'instant' (record index, the higher the index the newer the record), we first inspected the first three as they contain a rather small amount of categories. We plotted the mean target value per feature (Fig3): We can see that both 'hour' and 'month' present quite clear patterns, while 'year' being a binary feature only shows a single increase. This might explain the rather high correlation 'year' had with the target compared to 'hour' and 'month'. We also need to take into account that 'month' present a normal distribution, and 'hour' a mixture of normal distributions (probably centered at hours people go to and back from work), both aren't represented well by Pearson correlation. We also can't tell whether the increase trend 'year' presents is solid enough, as it is based on only two measured years. Therefore we decide to omit 'year' and keep 'month' and 'hour'.

'instant' on the other hand represented more gently the general trend of increasing number of rents over-time: the higher the index, the newer the rent record is. It might be suspected as target leakage, although we assume that new records still have an id. If they don't we can use their collection date instead. When using 'instant', we could omit the 'date' feature which was practically similar.

We also discarded remaining features that yielded small absolute Pearson correlation (< 0.05) - it could have been set as a hyper-parameter of the model

had we had a large number of features.

We compared our feature-selection pipeline with 3 other pipelines - f-regression, mutual-info and random as a baseline. Each was given with a same amount of features to select. Stratified train test split was used to avoid high imbalance between train and test sets (with target being split to bins), and then a 10-fold cross validation method was used to evaluate the models. MSE is reported for the chosen model on the test set.

Results (Manual is our model):

Bike-Sharing **Hourly** Data - 10-Fold CV Score			
Manual	F-regression	Mutual-Info-regression	Random
0.801220	0.791492	0.791692	0.679226

Test Set MSE for Manual (also best by CV): 6327
Other models MSEs: 6916, 6888, 27552

2 Vehicle Imports

A dataset that contains vehicles data in which the goal is to predict their price. Here we focused mainly on how to deal with missing data, which was rather prevalent, via imputation. The idea was again to do some examination of the data in order to better formulate the methods in use, compared to some well known out-of-the-box methods.

First, some early preparations were made in which we discarded entries with missing target value, and parsed the metadata file to classify which features should be treated as numerical and which are categorical.

Then, we tested for missing values in the table. The feautres which contained missing values were: "normalized-losses", "bore", "stroke", "horsepower", "peak-rpm", "num-of-doors"

In order to get some sense on how significant are the missing values in each columns, I tried to formulate a metric that compares the mean target value observed for existing and missing values in the features, weighted by the relative amount of missing values in that feature.

"normalized-losses" showed a quite large missing importance compared to the other inspected features (0.16 vs 0.03 for all others)

In all of these features with missing values, much lower missing-importance values were observed. Further exploration showed that in case of all of these features, a very small number of values were missing (between 2 and 4), where "normalized-losses" contained a rather large number (37). Another interesting point was that the pairs ('horsepower', 'peak-rpm') and ('stroke', 'bore') were missing in the exact same entries, decreasing the number of entries with missing values even further.

Due to the low missing-importance value together with the absolute low number of missing values, we make an assumption that all the mentioned feature

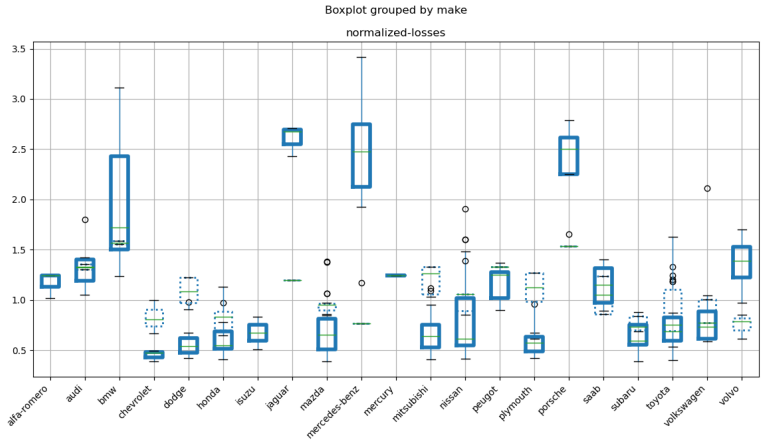

Figure 4: Fig4

expect for "normalized-losses" can be considered as MCAR, and can be treated with listwise deletion.

To better understand the missing pattern of "normalized-losses", we inspect its average values aggregated by the car 'maker' - which definitely has a strong indication of the car's value. We plotted box-plots of the normalized prices and losses of all cars, categorized by each 'maker' (Fig4). The solid lines are prices and the dotted ones are the losses.:

It can be seen the highest branded makers such as "prosche", "jagur" have very large gaps between the reported losses and the cars' prices, reaching even lower losses compared to lower tier cars manufacturers. As we've mentioned, the average car price of entries with missing values was extremely high compared to entries with reported losses. This can be perhaps explained by high-branded makers trying to obfuscate the value loss of their cars. We therefore concluded that missing values in "normalized-losses" cannot be avoided, as it doesn't seem to be MCAR but rather MAR, meaning that it has different miss probability for different values of 'make'.

To handle the missing values in this column, we impute by the average loss value of each 'maker' (ClassMeanImputer). For makers with non-determined mean value (due to their absence in the training data), we impute with the overall average loss.

For label encoding, we take two different approaches based on the number of distinct values in the feature. For features with low number of values (< 5), we perform one hot encoding to fully preserve the data. For features with a greater number of values, we use target mean encoding (TargetEncoder), where we compute the mean target value for each category, and by that form a mapping of these categories to numbers. This technique would be more relevants in large datasets, since one-hot-encoding a feature with n categories adds $m * n$ data cells to the table, where m is the number of records. We also constructed a

pipeline that use only one hot encoding.

Finally we use the same comparison setting as in the previous dataset, where we now compare to pipelines which use different simple imputation strategies ('mean', 'most-frequent' and 'median') and one-hot encode all categorical features. Due to the small size of the dataset, we perform repeated stratified split to test and train until a low mean difference was reached, to avoid high imbalance between the sets. The validation scheme to 20-fold cross validation to try and produce more reliable validation - sacrificing train time.

Imports Data - 10-Fold CV Score				
Manual1	Manual2	Mean	Median	Most-Frequent
0.845397	0.801093	0.824393	0.798315	0.819656

Test Set MSE for Manual1 (also best by CV): **3868661**
Other MSE results for comparison: 4082835, 4553740, 4631006, 4589046

As I saw it, the cardinality of this dataset made it very hard to produce stable results overtime, although most runs I've encountered ended up with an advantage towards the first model (Manual1). Even with high fold CV, the validation scheme wasn't reliable enough and didn't always correlated with the MSE results. I would either look for more data, or explore methods that specialize in working with small datasets such as this.

3 Further Work

- Working with more advanced models such as gradient boosting algorithms could produce better results. This would also increase the requirement for hyper-parameter tuning.
- In the bike-sharing dataset, it could be interesting to compare regression models that rely on the hourly data such as the one we used, to predict the daily data by aggregating the predictions, and then compare it to models built only using the daily data. This would really give an insight on whether the decomposed nature of the hourly data provides additional information.
- Follow the feature selection methodology used in the bike-sharing dataset in other large datasets, and see how far we can go using linear correlation based inferences, while still trying to maintain a very small portion of the feature space.
- Perform deeper inspection of the features in the vehicles dataset, and try to look for additional data online to verify the assumption of faulty loss reports in high branded cars.
- Try to work with ensemble models in the vehicles dataset, either on the whole dataset or even further by dividing the set to manufacturers tiers and then train a model for each tier.

- Inspect problematic samples (in our case, those which produce relatively high MSE), and look for patterns. Using these patterns we could improve the model by trying to obtain more data that conforms to these patterns, or reproduce similar data using random noise from our existing data, and let the model train on it and essentially strengthen its weak spot (while maintaining proper evaluation technique to avoid overfitting).
- Look for existing models that were trained on identical or similar data, and combine them with the methods introduced here to look for improvements.