Introduction to Machine Learning (SS 2023)
Programming Project

Author 1
Last name: Rahm
First name: Luca
Matrikel Nr.: 12123149

Author 2
Last name: Schneider
First name: Johannes
Matrikel Nr.: 12122799

I. Introduction

In this report we will talk about the correlation between the weather and the power consumption. In the given dataset we had a lot of different weather-data for the five cities: Bedrock, Gotham City, New New York, Paperopoli and Springfield and the target label, which is the power consumption. In total we got about 40000 samples and 66 features (one of them being the power consumption of course). These features contain the weather information for different cities, some of them being wind, snow and rain. Our goal is to analyze the features and find out the given power consumption regarding a given model - so we have a regression problem.

II. Implementation / ML Process

In the beginning we had to find out the important features and extract the good features to have good results and an efficient program. We used the 'pandas' - library to find out more about the CSV-File we got:

```
import numpy as np

print(powerpredict.describe())
# Returns summary of the data
print(powerpredict.isnull().sum())
# Checks for NULL-values
print(powerpredict.corr())
# Computes the correlation matrix
```

So with these details we can now determine which features are helpful and which aren't. The correlation between each feature and the power consumption is one of the most important details. So we sorted the dataset by correlation and dropped the features that were not correlated enough to the label (so the ones, closest to 0). In the end, we then had about 20-30 features that represented our data in a good manner. Another very important thing, was to replace the so called 'DOC' function, which was there to eliminate Non-Integer columns. But these columns could also contain important information and therefore have a high correlation to the power consumption, so we encoded these columns.

A. Linear Regression

With this problem being a regression problem we chose Linear Regression as our first solution. This kind of model aims to establish a linear relationship between features and their target label. It predicts the target of a data point by averaging the target values and drawing a line through the points that averages the data. For the implementation we used the 'sklearn' - library which already contained a predefined model for Linear Regression. The one thing that had to be adjusted were the hyper-parameters, so the amount of total features we wanted to choose (already stated in the implementation-section II), the train/test - split and the amount of Polynomial features. So we benchmarked the model for different settings to find the best configuration (more about performance in the results-section III). In the end we took the best values from the result, which were the size of the test set with 20%, the amount of polynomial features which was '2' combined with the 21 features that had the best correlation to the label of the model.

B. Random Forests

For the random forest model we used the same encoding method for all the non-numeric columns, as in the first approach. To extract the important features for this model we used a recursive feature elimination algorithm from the sklearn library. The algorithm returned a ranking from which we selected the best features. To verify our selection we compared the score of the model with only the selected features, to one which used all the features, and there was almost no difference in the score. The method we used for the second model is a random forest regressor. It is an ensemble learning method that combines multiple decision trees and we make use of bootstrapping, where each decision tree is trained on random subsets of the data and features. Random forests work well for our problem, because they can handle high dimensional data and can take both numerical and categorical input by design. Additionally they can capture non-linear relationships and interactions between features and are less prone to overfitting than normal decision trees. To choose the hyperparameters we performed a grid search using cross validation on the training set, the resulting hyperparameters worked well, except for the fact, that we had to reduce the number of trees to keep the model size below the given limit of 50mb.

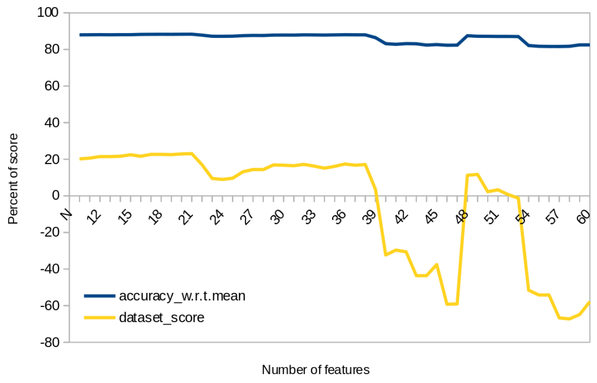
```
from sklearn.model_selection import
    GridSearchCV

grid_search = GridSearchCV(
    random_forest_regressor_model ,
    parameters)
```

The hyperparameters we selected were: the number of trees in the forest *n_estimators: 50*, the *max_depth: 17*, the amount of features to consider at each split *max_features: log2* and the *criterion: 'squared_error'*, other than that we used a train test split for the given data of 0.1 test set size, here we choose a smaller test set because the Random Forests can profit from more training data and they don´t tend to overfit as much as normal decision trees would.

III. Results

Mean squared error		
	Train set	Test set
Linear Regression:	3152.1798	3162.3335
Random Forests:	1145.2770	1944.8487

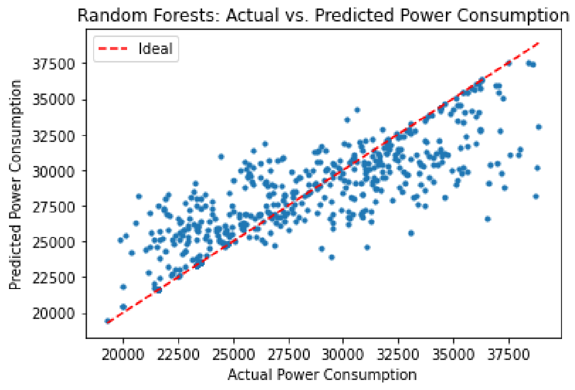
A. Linear Regression



The figure above illustrates the relationship between the score and the amount of features of the dataset in Linear Regression, considering the amount of features. The objective was to maximize the dataset_score. Similar analyses were also performed by adjusting the size of the test set and the number of features to optimize the performance of the model. As the number of features increases, the error also increases, indicating excessive overfitting. Instead, we aimed to select a model that provides the best overall fit to our data, which led us to choose 21 features for estimation.

B. Random Forests

The Random Forests model's performance we additionally evaluated by visually comparing the predicted and actual power consumption, as shown in the plot above. We can see that in general the predicted values follow the ideal line, but especially for the boundaries the model seems to generalize a bit.



IV. Discussion

Both of our implementations had a good performance and therefore a good overall score. One thing that we can see, is that the score of the training set is much lower than the test score in the Random-Forests-Implementation. This is the case, because non-parametric machine learning methods tend do overfit. Because of that, the test set is more biased then the training set. In the implementation of the linear regression model we don't have such a discrepancy, because it is less prone to overfitting and trained to perform good in general. (already stated in II-B)
At the beginning, we examined various alternative models including Ridge (Polynomial) Regression, Neural Networks, and K-Neighbors Regression. However, these models resulted in higher scores, indicating larger errors, and prone to overfitting. To enhance the performance of our models, we could explore further hyperparameter optimization or implement cross-validation on the training set to reduce variance.

V. Conclusion

For the overall Test-Set score we got the following results:

Linear Regression	3320.3084
Random Forests	3477.6875

In conclusion, our analysis reveals that estimating the power consumption for the collected data is a complex task that cannot be easily accomplished. Despite our overall performance, with scores ranging between 2000 and 3500, we acknowledge that these results are not optimal in our opinion. As highlighted in our earlier discussion (Section IV), we experimented with various models to predict power consumption, and all of them exhibited high training and test scores or a high variance. We believe that we have identified the most suitable models to fit the data. However, exploring the hyperparameters could be an additional idea to find the best model that may unlock additional opportunities to enhance our estimation accuracy and performance.