AML assignment 2 report

Roel Duijsings s1086375

Data analysis

inbound_loads.csv:

- all weight_uom values are in pounds (or NaN)
- 17756 different trailers in use. Some are used >267 times, others only once.
- During loading, the dock door opens, so the indoor temperature gets mixed with the outdoor temperature.

weather.csv

- Contain datetime, temperature and humidity.
- Weather is being measured only once every five minutes!!
- Connect datetime of demand_kW with weather.

demand_kWtrain_val.csv

- The train data ranges from 2018-12-31 21:15:00 to 2021-10-11 06:05:00.
- The validation data ranges from 2021-10-11 06:08:00 to 2021-12-13 17:59:00.
    - Therefore the validation data is only winter period, so expect lower temperatures! and therefore lower demand? Is the data representative?
- I found that the datetime objects in column 'datetime_local' are not sorted in order. For some reason time goes back 1 hour at 2021-11-07 01:59:00 (see table below). This influence the use of pd.merge_asof() which I use to merge the weather with the demand, since not all demand datetimes are represented in the weather file and therefore, I pick the nearest weather.

	datetime_local
312568	2021-11-07 01:58:00
312569	2021-11-07 01:59:00
312570	2021-11-07 01:00:00
312571	2021-11-07 01:01:00
312572	2021-11-07 01:02:00

Pallet_history_Gold_Spike.csv

-

After we connected df_demand with df_weather: df_demand_train.describe() (see table below).

- We see that demand_kW ranges from +- 300 to 3300 (a quite wide range), but most points range between 1500-2500. Pretty close to the mean, so you could say data is clustered around the mean.
- Relative humidity ranges from 9% to 100% (is 100% even possible?), with 95% of points between 28% and 100%. I expect there might be some errors in measuring relative humidity, because there are 6991 rows with 100% humidity and the second highest Relative Humidity is 97.05%, which is a big gap compared to the gaps between other measurements! Should I

remove these outliers? Perhaps this is due the fact that it was raining and the sensor got wet. I see that the Relative Humidity stays 100% for a couple of measurements in a row, on non-consecutive days. Which implies no technical error.

- Temperature also clusters around the mean pretty much.

	demand_kW	datetime	Relative Humidity	Temperature
count	117028.000000	117028	116241.000000	116877.000000
mean	2187.339130	2020-08-23 12:07:18	66.930375	70.062610
min	294.851640	2018-12-31 21:15:00	9.430000	-2.200000
25%	1806.044864	2019-11-05 10:41:15	52.030000	60.800000
50%	2252.156000	2020-09-20 16:37:30	68.340000	73.400000
75%	2572.752635	2021-06-30 23:26:15	83.130000	82.400000
max	3294.331000	2021-10-11 06:05:00	100.000000	107.600000
std	483.618044	NaN	20.114604	16.745439

Thinking process

First of all, I read through the assignment on Bright Space. Then, I scanned the data files that were provided.

After, scanning the demand_kWtrain_val.csv file, I realized that the only input is the datetime and the output is a float of the demand in energy in kW.

This datetime input can be connected to multiple pieces of information, such as:

- the weather at that datetime (temperature, humidity)
- the outbound/inbound loads
-

Problems

- What if a datetime in demand_valition is not in weather.csv? Should we take the temperature of nearby datetime? Or perhaps last years temperature? Or the average temperature for that hour? I would say, pick the nearest temperature, since that is more representative for the real temperature than last year's.
- Do we drop the

Raw validation results

This sections describes the intermediate results of the XGBoost classifier for a given set of data. The training dataset is split into 80% training and 20% validation data.

Temperature and relative humidity

Validation statistics:

MSE	85151.657
RootMSE	291.807
Mean value y_test	2405.976
RelRootMSE	0.1212 == 12.1% (good)

12.1% Relative Root Mean Squared Error is a decent score for such a small set of features, but could be improved.

Temperature, relative humidity and 5 hours load_duration

Validation statistics:

MSE	64101.71946819282
RootMSE	253.1831737461888
Mean value y_test	2403.664279856949
RelRootMSE	0.10533216966608026 == 10.5% (good)

Temperature, relative humidity and 2 hours load_duration

Validation statistics:

MSE	66095.18651509732
RootMSE	257.0898413300248
Mean value y_test	2405.495297978833
RelRootMSE	0.10687605232321142 == 10.7% (good)

Model parameters

To further improve the raw model (without any parameters), I analyzed the hyperparameters. This was done with this setup

```
model = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    max_depth=10,
    min_child_weight=1,
    early_stopping_rounds=5,
    random_state=42
)
```

With the next ranges:

Parameter	Min value	Max value
n_estimators		
Learning_rate		
Max_depth		
Min_child_weights		
Early_stopping_round		
Random_state		

Model with parameters results

Result 1

```
model = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    max_depth=10,
    min_child_weight=1,
    early_stopping_rounds=5,
    random_state=42
)
```

Training score	0.7999563254922486
MSE	48971.96376207384
RootMSE	221.29609974437832
Mean value y_test	2405.6100317815653
RelRootMSE	0.09199167646490447 == 9.2% (very good)

Result 2

```
model = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.1,
    max_depth=10,
    min_child_weight=1,
    early_stopping_rounds=5,
    random_state=42
)
```

Training score	0.80349568413666966
MSE	46112.66519529044
RootMSE	214.73859735802142
Mean value y_test	2407.6619983237088
RelRootMSE	0.08918967758245522 == 8.9% (very good)

Cross-validation

To get a better analysis of what are the best parameters to use, I use GridSearchCV and RandomizedSearchCV for cross-validation.

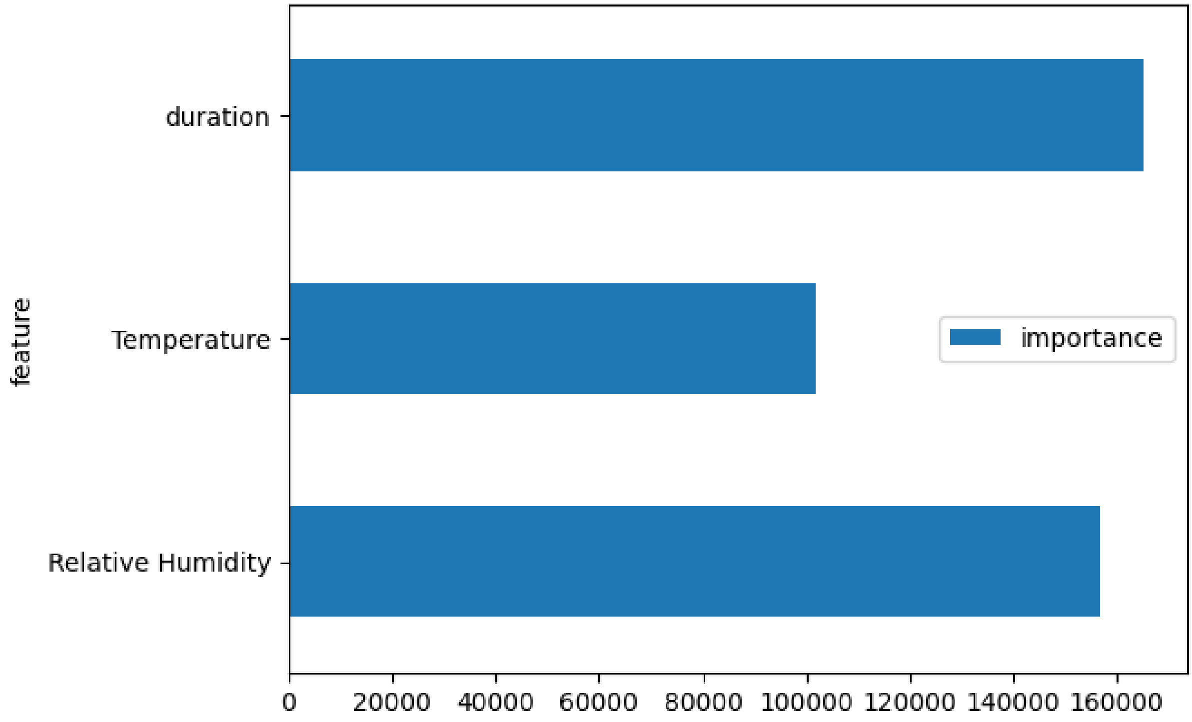
```
param_grid = {
    'n_estimators': [500, 1000, 1500],
    'learning_rate': [0.1, 0.3],
    'max_depth': [10],
    'min_child_weight': [1]
}
```

Which returns the parameters with the highest training score:

```
Best parameters: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'n_estimators': 1000}
Train score: 0.8293708324328936
```

Feature importance

After fitting the model, we can get the importance of the features of the training data.



In this graph we can see that load duration and relative humidity are way more important than the outdoor temperature. This is something that I did not expect.

ChatGPT prompts

Given this dataset, what models should I consider and why?

Based on the given dataset, it appears that you are working with a time series data, where you want to predict the demand in kW based on the datetime, relative humidity and temperature.

Here are some models that you can consider for this problem:

1. ARIMA (AutoRegressive Integrated Moving Average): ARIMA is a popular model for time series forecasting. It takes into account the past values of the dependent variable, and predicts the future values based on that. It can handle trends and seasonality in the data.

2. SARIMAX (Seasonal AutoRegressive Integrated Moving-Average with eXogenous regressors): SARIMAX is an extension of ARIMA that can also handle seasonal patterns in the data. In addition to the past values of the dependent variable, it also takes into account the past values of the exogenous variables (in this case, relative humidity and temperature).

3. Prophet: Prophet is a time series forecasting model developed by Facebook. It is based on a decomposable time series model with three main components: trend, seasonality, and holidays. It can handle missing data and outliers in the data, and also incorporates the effects of external regressors.

4. LSTM (Long Short-Term Memory): LSTM is a type of neural network that can handle sequential data like time series. It is particularly good at capturing long-term dependencies in the data. LSTM can be used for both regression and classification tasks.

5. XGBoost (Extreme Gradient Boosting): XGBoost is a popular machine learning model that can handle both regression and classification problems. It is based on gradient boosting and uses decision trees as base learners. It can handle missing data and is known for its high accuracy and speed.

The choice of the model will depend on various factors such as the size of the dataset, the number of features, and the nature of the data. You may try multiple models and compare their performance based on some evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared score.

Why should I use XGBoost?

You should consider using XGBoost because it is a powerful and highly flexible machine learning algorithm that is well-suited for a wide range of regression and classification problems, including time series forecasting. Here are some reasons why you might want to use XGBoost for your dataset:

1. High accuracy: XGBoost is known for its high accuracy and has been used to win many machine learning competitions. It uses an ensemble of decision trees and gradient boosting to iteratively improve the model's performance.

2. Handles missing data: XGBoost can handle missing data by automatically learning how to best impute missing values during training.

3. Feature importance: XGBoost provides an estimate of feature importance, which can help you identify which variables are most important in predicting the target variable.

4. Regularization: XGBoost includes regularization techniques to prevent overfitting, which can improve the model's generalization performance.

5. Speed: XGBoost is designed to be computationally efficient, and can train models on large datasets in a relatively short amount of time.

Overall, XGBoost is a versatile algorithm that can handle a wide range of problems and is known for its high accuracy and speed. If you are working with a regression problem and want to try a powerful machine learning algorithm, then XGBoost could be a good choice for your dataset.

XGboost

Uses Decision Tree Ensembles. The tree ensemble model consists of a set of classification and regression trees (CART)