1.DATA UNDERSTANDING

The data was obtained from the Kaggle website. We used three datasets from the five that were made available:

- sales_train_validation.csv - containing the data about sales of individual items in selected shops of the Walmart supermarket chain
- sell_prices.csv - with data about the sales prices of individual goods
- calendar.csv - with the dates on which the products are sold

Sales data

The original data frame *sales_train_validation* comprises 30490 rows and 1919 columns.
The first group of columns contains data about the item id, product category id, id of the department which it belongs to, and also the store and state in which the item was bought. One item is offered in multiple stores, therefore we can see that there is always one id for one particular item in one particular store. Hence it makes sense that there are 3049 unique items and 10 unique stores, which after multiplying goes up to 30490 rows.

```
Number of
 unique states: 3
 unique stores: 10
 unique departments: 7
 unique item categories: 3
 unique items: 3049
```

We also see that the items belong to 3 possible categories ('Hobbies', 'Household' or 'Foods') and come from 7 various departments, which are derived from the categories (see code results below). We can also see that the data comes from 3 states - California, Texas and Wisconsin, whereby there are 3 stores in each Texas and Wisconsin and 4 stores in California.

```
['HOBBIES' 'HOUSEHOLD' 'FOODS']
['HOBBIES_1' 'HOBBIES_2' 'HOUSEHOLD_1' 'HOUSEHOLD_2' 'FOODS_1' 'FOODS_2'
 'FOODS_3']
['CA_1' 'CA_2' 'CA_3' 'CA_4' 'TX_1' 'TX_2' 'TX_3' 'WI_1' 'WI_2' 'WI_3']
['CA' 'TX' 'WI']
```

The other group of columns, which is much more abundant, contains day numbers from *d_1* to *d_1913*. These columns help us understand how many units of the given product were sold on a particular day.
After inspecting, we find out that all 1913 'day columns' contain integer values, whereas the first descriptive columns mentioned above are all object values.

We also checked for the existence of null values in this dataframe, but we did not find any column- or row-wise. The data seems to be complete from this point of view. Just to make sure, we also checked for duplicated rows. Again, no such rows were found.

Calendar

The original data frame *calendar* comprises 1969 rows and 14 columns.

The columns contain data about:

- Date (*date*),
- ID of the week that day belongs to (*wm_yr_wk*),
- Name and order of weekdays (*weekday, wday*),
- Month and year of observation (*month,year*),
- Ordinal number of days (*d*),
- Name and type of event, occurring that day - two times for two different events happening at the same time (*event_name_, event_type_1, event_name_2, event_type_2*),
- Variables indicating, if at that particular day SNAP purchases were allowed in Walmarts in the state of California, Texas or Wisconsin respectively(*snap_CA, snap_TX, snap_WI).*

```
Number of
 unique years: 6
 unique events: 36
 unique types of events: 5
```

As we can see, those observations have been taken during six unique years (2011-2016) with 35 unique events, classified into 4 groups. Variable *event_name_2* does not contain any unique variables, those are not present in *event_name_1.*

```
[2011 2012 2013 2014 2015 2016]
[nan 'SuperBowl' 'ValentinesDay' 'PresidentsDay' 'LentStart' 'LentWeek2'
 'StPatricksDay' 'Purim End' 'OrthodoxEaster' 'Pesach End' 'Cinco De Mayo'
 "Mother's day" 'MemorialDay' 'NBAFinalsStart' 'NBAFinalsEnd'
 "Father's day" 'IndependenceDay' 'Ramadan starts' 'Eid al-Fitr'
 'LaborDay' 'ColumbusDay' 'Halloween' 'EidAlAdha' 'VeteransDay'
 'Thanksgiving' 'Christmas' 'Chanukah End' 'NewYear' 'OrthodoxChristmas'
 'MartinLutherKingDay' 'Easter']
[nan 'Sporting' 'Cultural' 'National' 'Religious']
[nan 'Easter' 'Cinco De Mayo' 'OrthodoxEaster' "Father's day"]
```

```
 object    7
int64     7
dtype: int64
```

Types of columns are divided into two groups - object and integer. Both groups contain 7 variables each.

```
date              object
wm_yr_wk           int64
weekday           object
wday               int64
month              int64
year               int64
d                 object
event_name_1      object
event_type_1      object
event_name_2      object
event_type_2      object
snap_CA            int64
snap_TX            int64
snap_WI            int64
```

Null values are present in the table due to the absence of specified events (via variable *event_name_1/2* and *event_type_1/2*). As we can see, *event_name/type_2* is present only 5 times, 1964 of rows missing any value. For *event_name/type_1* - 162 rows with values and 1807 without. Column-wise only those 4 variables are missing values in some of the rows.

```
0       4
1281    4
1293    4
1292    4
1291    4
       ..
85      0
1233    0
827     0
1177    0
1968    0
Length: 1969, dtype: int64
```

```
event_name_2    1964
event_type_2    1964
event_name_1    1807
event_type_1    1807
date               0
wm_yr_wk           0
weekday            0
wday               0
month              0
year               0
d                  0
snap_CA            0
snap_TX            0
snap_WI            0
```

No duplicate rows were found, we can say that the dataset is complete with understanding of null values within event-related variables.

Sell prices

The original data frame *sell_prices* comprises 6841121 rows and 4 columns.

Columns contain data about:

- Store_id (*store_id*)
- Item_id (*item_id*)
- ID of the week (*wm_yr_wk*)
- Sell price (*sell_price*)

```
Number of
 unique stores: 10
 unique item ids: 3049
 unique id_weeks: 282
 unique sell_prices 1048
```

We can see that the number of unique stores and item ids is identical to dataset *sales_train_validation,* also we have 282 unique weeks and 1048 unique prices - some of the items are sold for the same price.

```
['CA_1' 'CA_2' 'CA_3' 'CA_4' 'TX_1' 'TX_2' 'TX_3' 'WI_1' 'WI_2' 'WI_3']
['HOBBIES_1_001' 'HOBBIES_1_002' 'HOBBIES_1_003' ... 'FOODS_3_825'
 'FOODS_3_826' 'FOODS_3_827']
[11325 11326 11327 11328 11329 11330 11331 11332 11333 11334 11335 11336
 11337 11338 11339 11340 11341 11342 11343 11344 11345 11346 11347 11348
 11349 11350 11351 11352 11353 11401 11402 11403 11404 11405 11406 11407
 11408 11409 11410 11411 11412 11413 11414 11415 11416 11417 11418 11419
 11420 11421 11422 11423 11424 11425 11426 11427 11428 11429 11430 11431
 11432 11433 11434 11435 11436 11437 11438 11439 11440 11441 11442 11443
 11444 11445 11446 11447 11448 11449 11450 11451 11452 11501 11502 11503
 11504 11505 11506 11507 11508 11509 11510 11511 11512 11513 11514 11515
 11516 11517 11518 11519 11520 11521 11522 11523 11524 11525 11526 11527
 11528 11529 11530 11531 11532 11533 11534 11535 11536 11537 11538 11539
 11540 11541 11542 11543 11544 11545 11546 11547 11548 11549 11550 11551
 11552 11601 11602 11603 11604 11605 11606 11607 11608 11609 11610 11611
 11612 11613 11614 11615 11616 11617 11618 11619 11620 11621 11121 11122
 11123 11124 11125 11126 11127 11128 11129 11130 11131 11132 11133 11134
 11135 11136 11137 11138 11139 11140 11141 11142 11143 11144 11145 11146
 11147 11148 11149 11150 11151 11152 11201 11202 11203 11204 11205 11206
 11207 11208 11209 11210 11211 11212 11213 11214 11215 11216 11217 11218
 11219 11220 11221 11222 11223 11224 11225 11226 11227 11228 11229 11230
 11231 11232 11233 11234 11235 11236 11237 11238 11239 11240 11241 11242
 11243 11244 11245 11246 11247 11248 11249 11250 11251 11252 11301 11302
 11303 11304 11305 11306 11307 11308 11309 11310 11311 11312 11313 11314
 11315 11316 11317 11318 11319 11320 11321 11322 11323 11324 11106 11107
 11108 11109 11110 11111 11112 11113 11114 11115 11116 11117 11118 11119
 11120 11101 11102 11103 11104 11105]
[  9.58   8.26   8.38 ... 107.32   8.07  18.47]
```

Data types of variables are object for *store_id* and *item_id*, *wm_yr_wk* is integer and *sell_price* is float.

```
store_id        object
item_id         object
wm_yr_wk         int64
sell_price     float64
dtype: object
```

Null values are not present in data frame column-wise and row-wise. Duplicate rows were not found either. Data set seems complete.

2. DATA PREPROCESSING

Following the first part of data understanding we proceeded to data preprocessing. We have uploaded the item_id_sample dataset and printed what's inside. This dataset was basically a list of items which we should be considering in the whole rest of the process. This allowed us to get rid of a decent portion of the initial sales_validation data by removing the undesired items.

Firstly, we checked whether there are some id's in the sales_validation which are not in the items dataset and found out there are a lot of them. So with the merge function we **filtered out** all of the rows with ids which are not in the items dataset.

Then we checked the calendar dataset and realized there is a column "d" in a **long-form** while our so far-merged data frame contains the same column but in **wide-form**. We've melted columns d_1 to d_1941 to long-form and merged the calendar with our dataframe.

Then we dealt with columns event_name_1, event_name_2, event_type_1, event_type_2 which are, for some reason, splitted into two parts. Because of the data saving we've decided to merge these columns together so instead of 4 columns we have only 2: **event_name** and **event_type**.

Then based on store id, item id and wm_yr_wk variables we've merged into our dataframe last dataset sell prices. The view of the dataset so far is:

```
                          id      item_id  dept_id cat_id store_id state_id  \
0  FOODS_1_011_CA_1_evaluation  FOODS_1_011  FOODS_1  FOODS     CA_1       CA
1  FOODS_1_011_CA_1_evaluation  FOODS_1_011  FOODS_1  FOODS     CA_1       CA
2  FOODS_1_011_CA_1_evaluation  FOODS_1_011  FOODS_1  FOODS     CA_1       CA
3  FOODS_1_011_CA_1_evaluation  FOODS_1_011  FOODS_1  FOODS     CA_1       CA
4  FOODS_1_011_CA_1_evaluation  FOODS_1_011  FOODS_1  FOODS     CA_1       CA

        d  sales        date wm_yr_wk  ... month    year  snap_CA  snap_TX  \
0     d_1      2  2011-01-29    11101  ...   1.0  2011.0      0.0      0.0
1    d_10      1  2011-02-07    11102  ...   2.0  2011.0      1.0      1.0
2   d_100      0  2011-05-08    11115  ...   5.0  2011.0      1.0      0.0
3  d_1000      0  2013-10-24    11339  ...  10.0  2013.0      0.0      0.0
4  d_1001      0  2013-10-25    11339  ...  10.0  2013.0      0.0      0.0

   snap_WI    event_name  event_type sell_price
0      0.0           NaN         NaN       2.28
1      0.0           NaN         NaN       2.28
2      1.0  Mother's day    Cultural       2.28
3      0.0           NaN         NaN       2.48
4      0.0           NaN         NaN       2.48
```

After further examination we have decided to create two more columns: **weekday_binary** and **event_binary**. Weekday binary is a binary column derived from column "weekday" where 0 = weekday, 1 = weekend. Event binary is derived from event_name where 0 = no event this day, 1 = some event on this day. We've created this because it allows us to work efficiently with the data during machine learning transformations and also due to parsimony for our analysis is enough to have binary variables marking potentially significant events

(weekend, holiday) than have two variables marking a lot of different day names and events which could confuse the model and thus worsen the results.
Then we've changed the column “d” the datatype from string to integer by removing the “d_” in each row. It allows us to work with the number of days in numeric form more efficiently for the algorithm.
We've also created a column “**turnover**” (sales*sell_price) because it is a basic economic metric which we assume could be significant for our analysis.

3. MODELLING

While preparing data for modeling, we determined that utilizing aggregate data for departments, stores, and dates would streamline our process. Subsequently, we incorporated aggregate variables related to turnover, such as turnover_month_avg, turnover_week_avg, sales_d_1, profit_d_2, and profit_d_3.

In all our modeling endeavors, we implemented a simple train validation split. We partitioned the training + validation data. We did so by taking the first 80% of the rows for training and the remaining 20% for validation. We did so in order to respect the time order of the series.

After the data split, we started off with two baseline models. If these had better RMSE than our trained models, we could consider our trained models obsolete.
The first baseline model was created solely by predicting zero sales (thus also turnover) for every day. The RMSE of turnover predictions for this model was 7.54.
The other baseline model was slightly more sophisticated. We created our predictions by taking the average monthly sales (moving average for the last 30 days before the prediction day) and multiplied the predicted sales by the sell prices for the given day. This method yielded a better RMSE, 5.34.

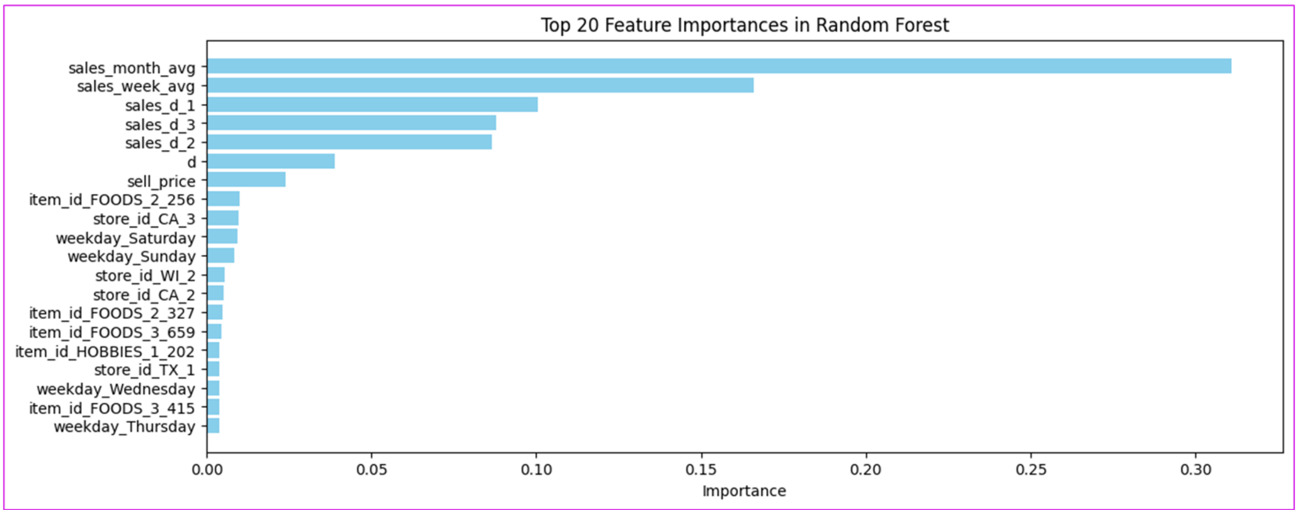
We then delved into other models.
Firstly we experimented with different types of regression models. We began with a simple linear regression model, which had an RMSE of 5.38. Then we decided to execute some regularization of the model because of the high number of features, therefore we used a Ridge regression method. The lowest RMSE reached by this algorithm was attributed to an alpha value of 20. This value expresses the penalty posed on every additional model feature, increasing the value of the loss function of the model.
After trying out the regression methods, we went over to some less explainable algorithms.

The first model we experimented with was the Random Forest. For this one, we created lists of hyperparameter values for the most significant hyperparameters. These were e.g. the number of estimators, maximum depth of the three, minimum number of samples per leaf and others. Using a sequence of for loops we let the computer iterate through all of the hyperparameter combinations and the resulting combination of optimal values (optimal from the selection we provided - almost surely not the overall best) was following:

```
Best parameters for sale prediction:  {'n_estimators': 15, 'max_depth': 10, 'min_samples_split': 8, 'min_samples_leaf': 6, 'max_features': None}
Best parameters for turnover prediction:  {'n_estimators': 15, 'max_depth': 20, 'min_samples_split': 4, 'min_samples_leaf': 2, 'max_features': 'sqrt'}
Best RMSE for sales prediction:  1.0552336607356605
Best RMSE for turnover prediction:  5.322859696665507
```

We can see that the RMSE for the turnover prediction was just slightly better than the one of linear regression models. The question remains, if this is caused by a sub-optimal hyper parameter tuning or just the fact that this algorithm is not performing much better on this type of data.

Using this model, we also inspected the importances of the individual features. As we can see in the bar chart below, the most significant features were the synthetic ones created by us. Namely, the moving monthly and weekly sales average were the most significant ones, accompanied by the sales on previous days d-1, d-2 and d-3.



As a final model, we selected the XGBoost. Again, we tried to tune our hyperparameters by predefining some values and used for loops to iteratively find the best combination. Below, we can see the optimal values of the hyperparameters. We can see that the number of estimators is considerably higher than in the Random forest.

```
Best parameters for sale prediction:  {'objective': 'count:poisson', 'learning_rate': 0.05, 'n_estimators': 400, 'subsample': 0.4, 'max_depth': 5}
Best parameters for turnover prediction:  {'objective': 'count:poisson', 'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 0.8, 'max_depth': 5}
Best RMSE for sales prediction:  1.0477940982840108
Best RMSE for turnover prediction:  5.2132512905735835
```

As one can see from the results, the value of RMSE is the lowest of all models that we tried out, 5.21. We therefore decided to name this model the best fit for this task.

We were also interested in the feature importances in this model. As illustrated in the chart below, the moving monthly sales average turned out to be the most dominant predictor, outweighing all of the other features. Second place, same as in the Random forest model, belonged to the moving weekly sales average. Other than in the Random forest model, however, days of the weekend turned out to have the third and fourth highest importance, accompanied by the month of december. This could most probably be attributed to the fact that people would do a lot of pre-Christmas buys every year. We can see that the sales on the previous days d-1, d-2 and d-3 played less significant roles.