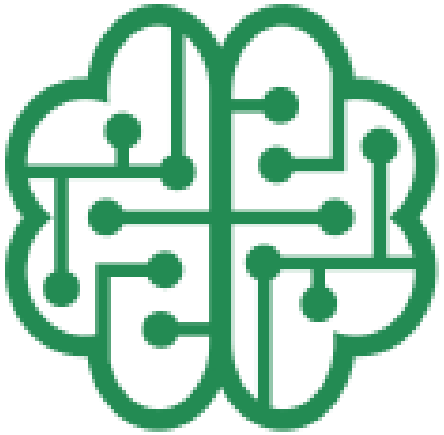


Machine Learning Graz

Electricity load forecasting for N. Macedonia

Final project for the course **“Data Science at home”** organized by ML Graz

Prepared by: Petranka Boncheva

Data sources

- ENTSO-E (European Network of Transmission System Operators for Electricity) platform for electricity data
- Wunderground.com (no downloadable data available, web scraping script)

Objectives

- Exploring data
- Cleaning and merging data
- Build a model on electricity load forecasting based on historical values and temperature data

Cleaning and visualisation: Electricity load

3 data sets cleaned and combined together:

- ! wget https://eepublicdownloads.blob.core.windows.net/public-cdn-container/clean-documents/Publications/Statistics/MHLV_data-2015-2017.xlsx
  df=pd.read_excel("MHLV_data-2015-2017.xlsx")

- df_MK_2018=pd.read_csv("Total Load - Day Ahead _ Actual_201801010000-201901010000.csv")
- df_MK_2019=pd.read_csv("Total Load - Day Ahead _ Actual_201901010000-202001010000.csv")

	MeasureItem	DateUTC	DateShort	TimeFrom	TimeTo	CountryCode	Cov_ratio	Value	Value_ScaleTo100
0	Monthly Hourly Load Values	2014-12-31 23:00:00	2014-12-31	23:00:00	00:00:00	DE	98	46419.79	47367.132653
1	Monthly Hourly Load Values	2015-01-01 00:00:00	2015-01-01	00:00:00	01:00:00	DE	98	44898.30	45814.591837
2	Monthly Hourly Load Values	2015-01-01 01:00:00	2015-01-01	01:00:00	02:00:00	DE	98	43305.31	44189.091837
3	Monthly Hourly Load Values	2015-01-01 02:00:00	2015-01-01	02:00:00	03:00:00	DE	98	41918.17	42773.642857
4	Monthly Hourly Load Values	2015-01-01 03:00:00	2015-01-01	03:00:00	04:00:00	DE	98	41330.17	42173.642857

df_MK=df[df.Country Code == "MK"]

	LoadMW	dayweek	month	year	day	hour	weekend
8789	1187.0	3	12	2015	31	23	False
8823	1142.0	4	1	2016	1	0	False
8858	1059.0	4	1	2016	1	1	False
8893	976.0	4	1	2016	1	2	False
8928	929.0	4	1	2016	1	3	False

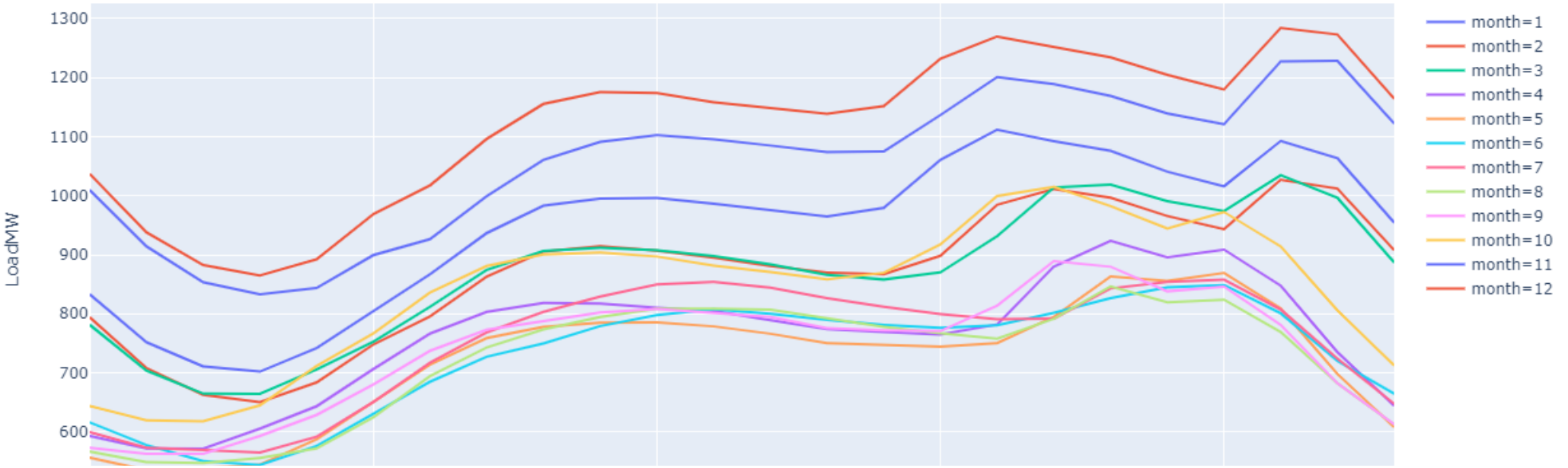
Cleaned data with added columns:

- Dayweek
- Month
- Year
- Day
- Hour
- Weekend

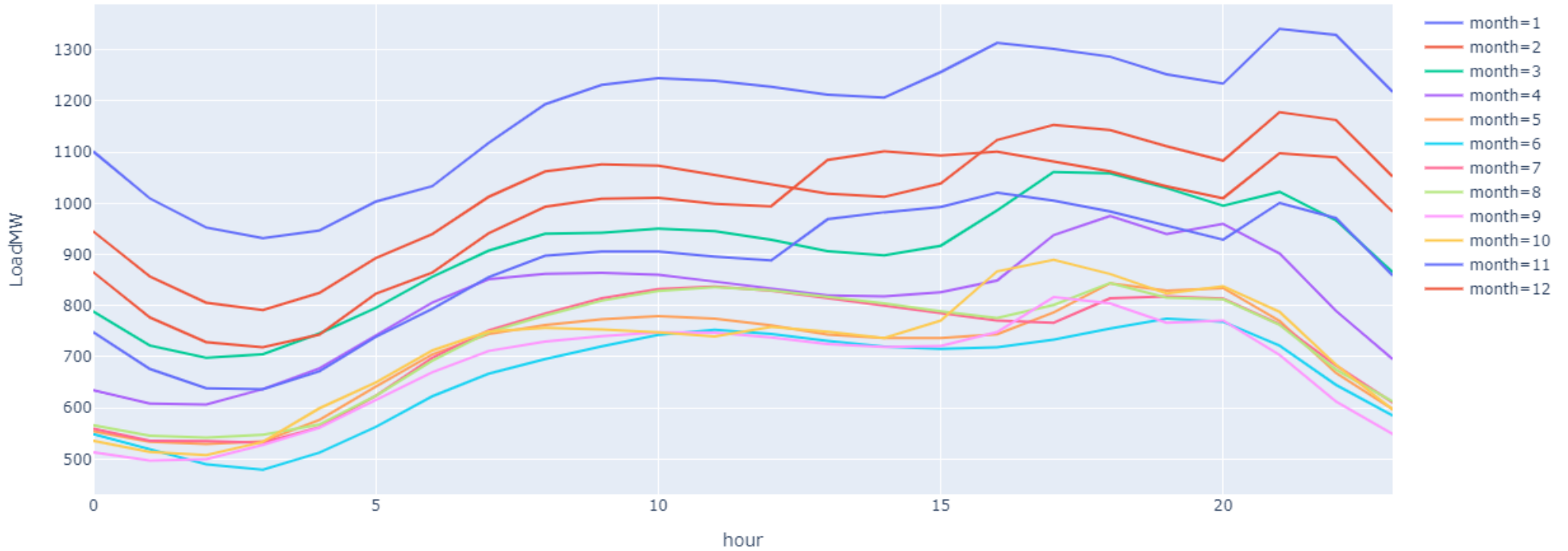
Cleaning some typo errors:

```
df_rows4=df_MK_merged[(df_MK_merged.LoadMW >= 3000) & (df_MK_merged.LoadMW <=9999)].index
df_MK_merged.loc[df_rows4, 'LoadMW']=df_MK_merged.loc[df_rows4, 'LoadMW']/10
df_rows5=df_MK_merged[(df_MK_merged.LoadMW >= 10000) & (df_MK_merged.LoadMW <=99999)].index
df_MK_merged.loc[df_rows5, 'LoadMW']=df_MK_merged.loc[df_rows5, 'LoadMW']/100
df_rows6=df_MK_merged[(df_MK_merged.LoadMW >= 100000) & (df_MK_merged.LoadMW <=999999)].index
df_MK_merged.loc[df_rows6, 'LoadMW']=df_MK_merged.loc[df_rows6, 'LoadMW']/1000
df_MK_merged.head()
```

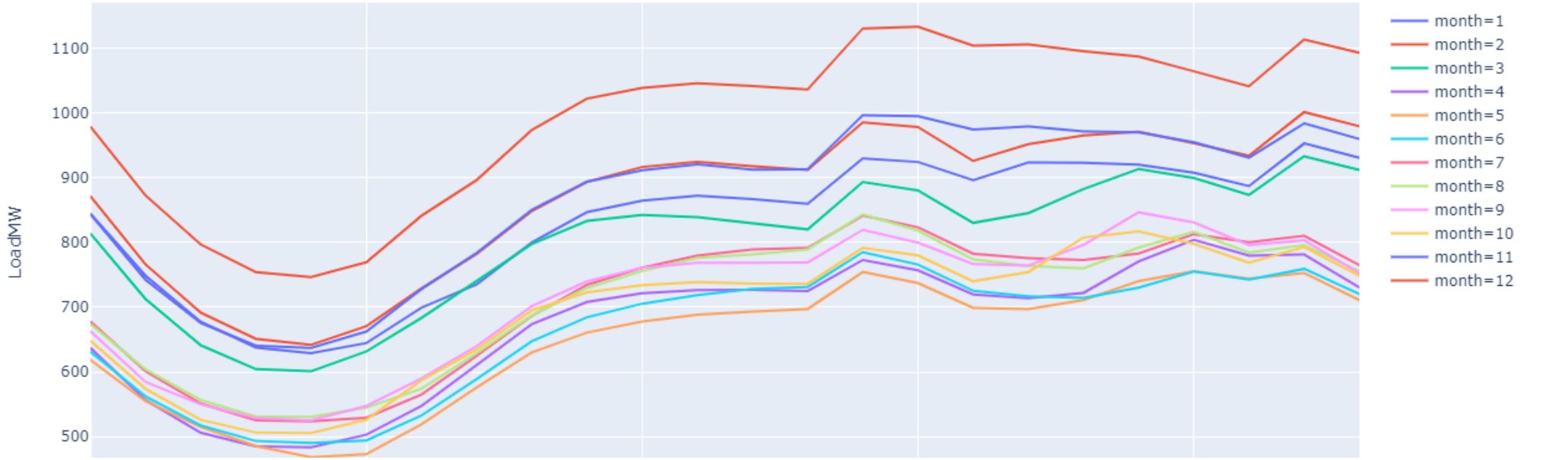
Average hourly consumption of the total consumption for 2016



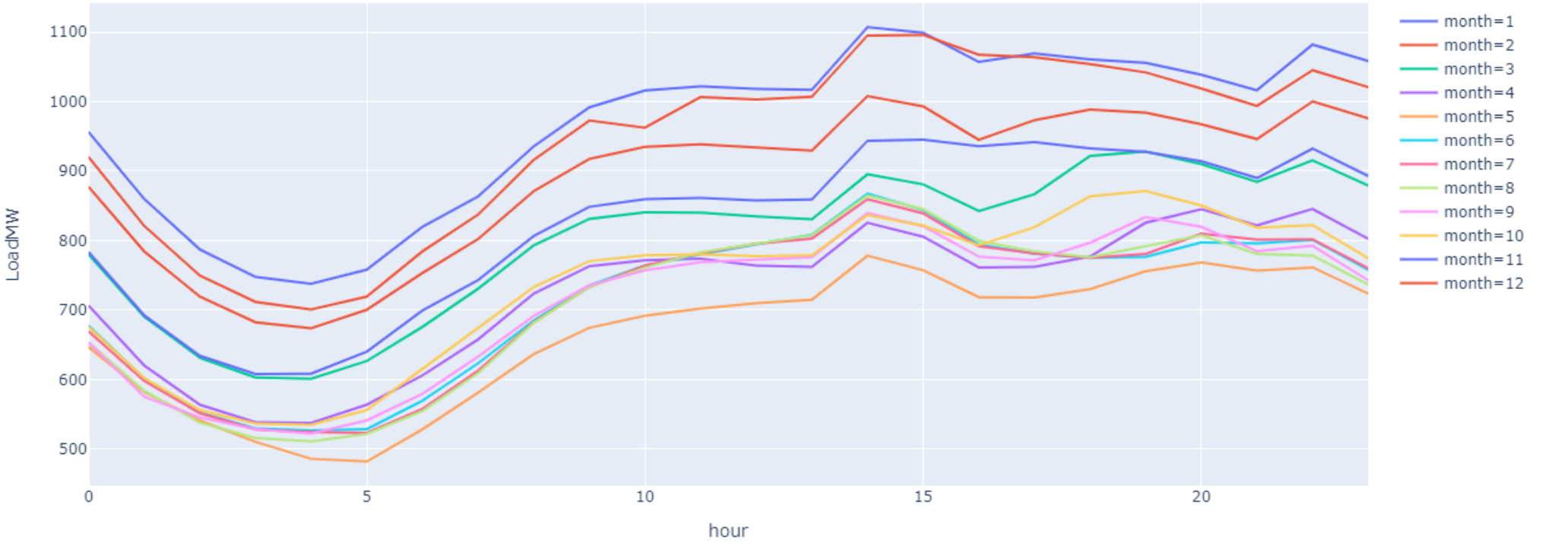
Average hourly consumption of the total consumption for 2017

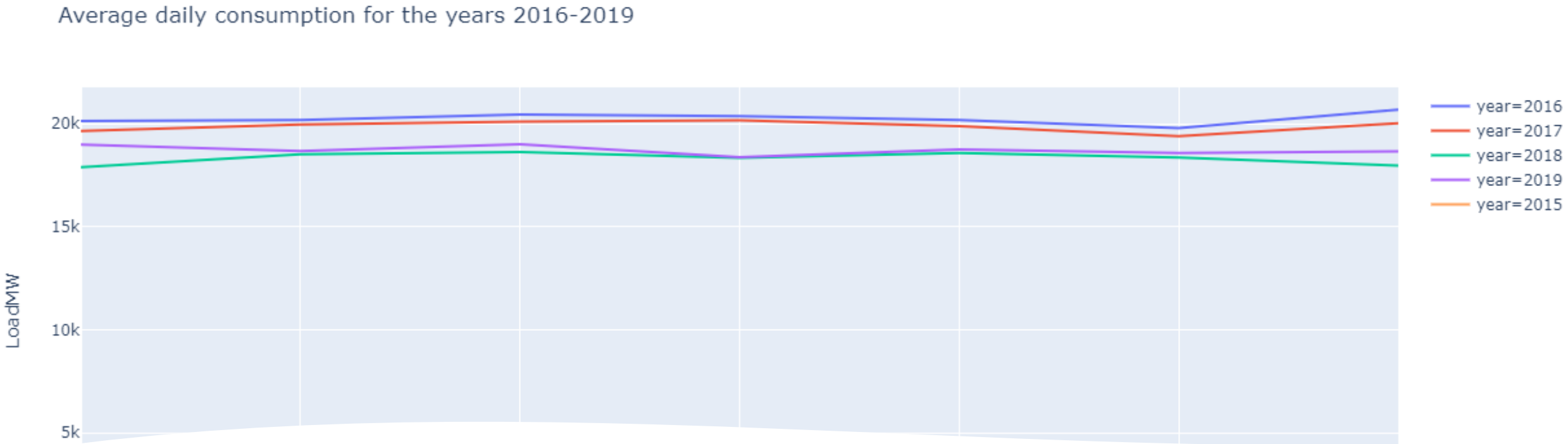


Average hourly consumption of the total consumption for 2018



Average hourly consumption of the total consumption for 2019





- #Average daily consumption of the total consumption for the years 2016,2017,2018,2019
- df_MK_year=df_MK_merged.groupby(["dayweek", "month", "year","hour"]).LoadMW.mean().reset_index()
- df_MK_year.head()
- df_MK_year_grouped=df_MK_year.groupby(["dayweek", "year", "hour"]).LoadMW.mean().reset_index()
- df_MK_year_grouped.head()
- df_MK_year_groupedf=df_MK_year_grouped.groupby(["dayweek", "year"]).LoadMW.sum().reset_index()
- px.line(df_MK_year_groupedf, x="dayweek", y="LoadMW", color="year", title="Average daily consumption for the years 2016-2019")

Cleaning: Temperature data (from web scraping)

```
df_t2015=pd.read_csv("2015dec.csv", sep=';')
df_t2016=pd.read_csv("2016n.csv", sep=';')
df_t2017=pd.read_csv("2017n.csv", sep=';')
df_t2018=pd.read_csv("2018n.csv", sep=';')
df_t2019=pd.read_csv("2019n.csv", sep=';')
#drop the last two rows of 2019
#df_t2019 = df_t2019.iloc[:-1]
#only last row of december 2015
df_t2015=df_t2015.iloc[[-1,-2]]
df_total=[df_t2015, df_t2016, df_t2017, df_t2018, df_t2019]



df_temperature=pd.concat(df_total, ignore_index=True, axis=0)
df_temperature.reset_index(drop=True, inplace=True)
df_temperature.head(90000)
```

	Date	Time	Temperature	DewPoint	Humidity	Wind	WindSpeed	WindGust	Pressure	Precip.	Condition
0	2015-12-31	12:30AM	14F	9F	79%	SE	2mph	0mph	29.59in	0.0in	Fair
1	2015-12-31	12:00AM	16F	10F	79%	CALM	0mph	0mph	29.59in	0.0in	Fair
2	2016-1-1	1:00AM	14F	9F	79%	CALM	0mph	0mph	29.59in	0.0in	Fair
3	2016-1-1	1:30AM	14F	7F	73%	CALM	0mph	0mph	29.59in	0.0in	Fair
4	2016-1-1	2:00AM	14F	7F	73%	CALM	0mph	0mph	29.59in	0.0in	Fair

Clean temperature data:

	Temperature	Year	Month	Day	Hour
2	14F	2016	1	1	1
4	14F	2016	1	1	2
6	12F	2016	1	1	3
8	10F	2016	1	1	4
9	9F	2016	1	1	5
...	...	...	...	...	...
68616	32F	2019	12	31	20
68618	34F	2019	12	31	21
68620	32F	2019	12	31	22
68622	32F	2019	12	31	23
68624	32F	2019	12	31	0

```
df_temperaturefinal['Temperature']=df_temperaturefinal.Temperature.str.replace('F', '').astype(float)
```

```
df_temperaturefinal['Temperature']=(df_temperaturefinal['Temperature']-32)*(5/9)
```

```
df_temperaturefinal.to_csv("/content/MK_Temperature2016-2019C.csv", index=False)
```

Model preparation

Using IterativeImputer instead of SimpleImputer to fill the missing values, since it seems wrong to do mean or most frequent values in the case of hourly electricity values.

```
[90] #Dealing with missing values
     df_missing_loc=df_electricity.loc[df_electricity.LoadMW.isna()].index

     from sklearn.experimental import enable_iterative_imputer
     from sklearn.impute import IterativeImputer

     imp = IterativeImputer(max_iter=10, verbose=0)
     imp.fit(df_electricity)
     df_electricity_imp = imp.transform(df_electricity)
     df_electricity_imp = pd.DataFrame(df_electricity_imp, columns=df_electricity.columns)
     df_electricity_imp.LoadMW.isna().sum()
```

Merge the two data frames:

```
dfinal = df_electricity_imp.merge(df_temperature, how='left', left_on=['year
', 'day', 'month', 'hour'], right_on=['Year', 'Day', 'Month', 'Hour'])
```

Final data frame ready for the modelling part:

	LoadMW	dayweek	month	year	day	hour	weekend	Temperature
0	1187.0	3.0	12.0	2015.0	31.0	23.0	0.0	12.777778
1	1142.0	4.0	1.0	2016.0	1.0	0.0	0.0	-8.888889
2	1059.0	4.0	1.0	2016.0	1.0	1.0	0.0	-10.000000
3	976.0	4.0	1.0	2016.0	1.0	2.0	0.0	-10.000000
4	929.0	4.0	1.0	2016.0	1.0	3.0	0.0	-11.111111

The modelling part

- df_electricity_imp = pd.get_dummies(df_electricity_imp, drop_first=True)
- X = df_electricity_imp.drop('LoadMW', axis=1)
- y = df_electricity_imp['LoadMW']

```
y.head()
```

```
0    1187.0
1    1142.0
2    1059.0
3     976.0
4     929.0
Name: LoadMW, dtype: float64
```

```
X.head()
```

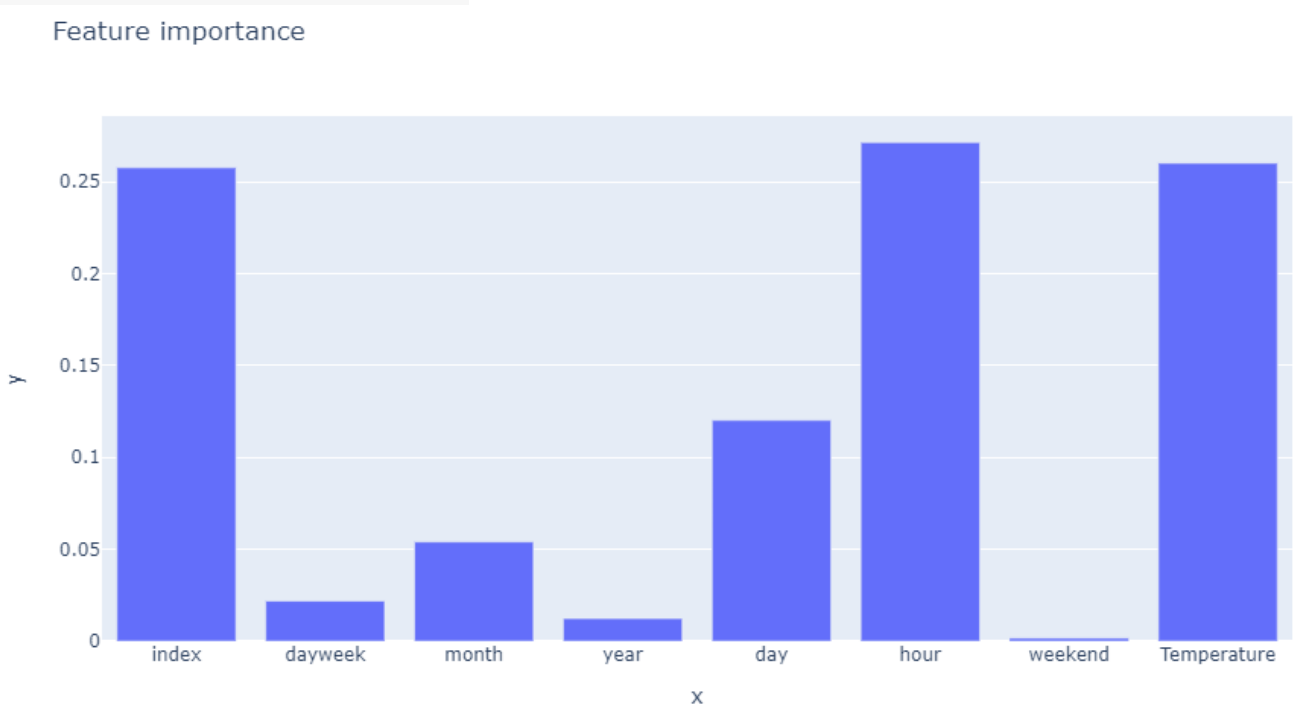
	index	dayweek	month	year	day	hour	weekend	Temperature
0	0	3.0	12.0	2015.0	31.0	23.0	0.0	12.777778
1	1	4.0	1.0	2016.0	1.0	0.0	0.0	-8.888889
2	2	4.0	1.0	2016.0	1.0	1.0	0.0	-10.000000
3	3	4.0	1.0	2016.0	1.0	2.0	0.0	-10.000000
4	4	4.0	1.0	2016.0	1.0	3.0	0.0	-11.111111

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
```

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(verbose=True)
rfr.fit(X_train, y_train)
```

```
y_predrfr = rfr.predict(X_test)
r2_score(y_test, y_predrfr)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBacken
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:
0.9421431009935133
```



R2_score looks really good, BUT…

```python
from sklearn.model_selection import cross_validate
#rfr = RandomForestRegressor(verbose=True)
scores = cross_validate(rfr, X, y.values.ravel(), cv=3,scoring=('r2'),return_train_score=True)
```

```python
scores
```

```
{'fit_time': array([9.10849118, 8.68201542, 8.50724053]),
 'score_time': array([0.11947107, 0.14139843, 0.09344721]),
 'test_score': array([-0.07425893,  0.42342844, -0.03673996]),
 'train_score': array([0.98144646, 0.98471652, 0.99648211])}
```

… something is wrong !

Future work:

- **Target Analysis ???**
- **Seasonality and Trend ???**