Predicting German day-ahead power prices

The topic of this work is related to the European Electricity Market. I have data for power prices, demand, productions, net import, costs, and weather. With this historical data I'm going to predict the day-ahead electricity prices for May and June 2022 of the German Market.

Methodology:

The main task is to forecast the day-ahead power prices. Here's how I'm going to do so:

1) Data preprocessing: clean the data to remove potential outliers and keep only relevant data.
2) Build models: I will build different models and perform feature selection to predict future prices and assess their performance to see which model is the best. The models I will test are Linear Regression, Ridge Regression, Random Forest, ARIMA, LSTM and XGBoost.
3) Predictions: I will predict May and June power prices using the best model.

I. Data:

Before doing any calculation, I had a look at the data to see if the data was clean. Wind, Clean Dark Spread and Clean Spark Spread had some missing values so I forward filled the dataset, i.e the values with missing values were replaced by the value of the previous non missing value. As I had only one price per day for the Spread while the others dataset had hourly data, I assumed that the price of the Spread was the same for each hour of the day to make things easier.

Then I take care of the potential outliers. I had a look at the distribution of the different features and saw that most of the features do not have a normal distribution so I can not use the z-score or standard deviation technique to detect outliers. I use the Interquartile Range method.

To do so, I define a custom range that accommodates all data points that lie anywhere between 0.5 and 99.5 percentile of the dataset. Rather than just deleting these outliers, I prefer to set the value to NaN and then fill the value by interpolation rather than forward fill to have a more realistic and smoother value.

Now that my data is cleaned, I can build models to predict power prices.

Note that my global data set consists of hourly day-ahead prices for German markets, hourly actual energy productions for different areas, actual hourly consumption, net physical electricity flows from neighbor countries to Germany (import-export), Clean spark spread and clean dark spread, average country temperature and wind speed from **2019-01-01 to 2022-04-30.**

II. Building Models:

a) *Linear Regression*

Linear regression is a fundamental statistical method used to model and analyze the relationship between a dependent variable and one or more independent. The basic idea is to find the best-fitting straight line through the data points that can be used to predict the value of the dependent variable based on the values of the independent variables. It can be described by the following equation:

$$y = b_0 + b_1.x_1 + b_2.x_2 + \cdots + b_n.x_n$$

Where y: dependent variable in my case it is day-ahead power prices.

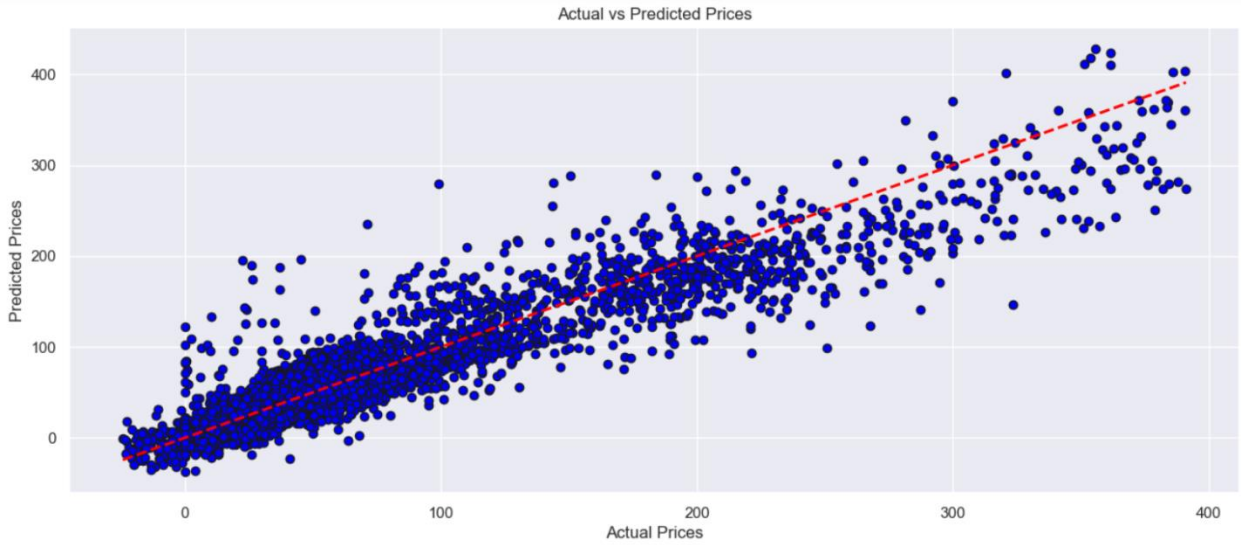
$$x_1, x_2, x_n \; are \; the \; independent \; variables \; in \; my \; case \; it's \; production, consumption, \ldots$$

$$b_1, b_{2,} b_n \; are \; the \; coefficients \; of \; the \; independent \; variables \; that \; I \; need \; to \; find \; and \; b_0 \; the \; intercept$$

Linear Regression relies on 4 assumptions:

- The relationship between the dependent and independent variables is linear.
- Observations are independent of each other.
- The residuals (errors) have constant variance at every level of x.
- The residuals of the model are normally distributed.

To do so I prepare my different x_i and my dependent variable y and split the dataset into 80% training to obtain the coefficients and 20% validation to predict the prices with the coefficients I found and compare this to the actual true prices. So I train Linear Regression on my training data and evaluate it on my validation data. Here are the results I obtain:



I obtain a **coefficient of determination of 0.8588, an MSE of 621 and a MAE of 16.83**. By looking at the above plot we notice that there is some dispersion between the actual and predicted prices. Indeed, the red dashed line represents a perfect prediction line where actual and predicted values would be identical. We can see that points do not cluster closely around the diagonal red dashed line they are scattered, indicating that the model's predictions are not that close to the true values.

The coefficient of determination indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, so mine with 0.8588 indicates a good fit.

However, MAE is the average of the absolute differences between the predicted values and the actual values. It gives an idea of how much the predictions differ from the actual values on average while MSE is the average of the squared differences between the predicted values and the actual values. Squaring the differences emphasizes larger errors more than smaller ones. In my case, I have a MSE of 621 and MAE of 16.83 which are quite high values suggesting that Linear regression performed poorly even though I have a high coefficient of determination.

To see if my model is correct, I need to test if residuals follow a normal distribution because it is one of the main assumptions of Linear Regression.

If the residuals do not follow a normal distribution, it may indicate that the model does not adequately capture the variability of the data or that there are nonlinear or non-Gaussian structures in the data. In such cases, the model's predictions may not be reliable outside the range of the observed data. In the context of linear regression, the model coefficients are interpreted as the average effects of each independent variable on the dependent variable, under the assumption that the residuals follow a normal distribution. If this assumption is violated, the interpretation of the coefficients may be incorrect.

To test the normality of the residuals I do a Shapiro test where the null Hypothesis is that the residuals follow a normal distribution, and the alternative hypothesis is that residuals do not follow a normal distribution. I get a Shapiro Test Statistic of 0.908 so we reject the null hypothesis so residuals do not follow a normal distribution and this assumption is violated so it may explain why Linear Regression did not perform really well.

To improve things, I will add a ridge penalty and run a Ridge Regression.

b) Ridge Regression

In a standard Linear Regression model, the aim is to minimize the residual sum of squares (RSS) defined as:

$$RSS = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \; where \; y_i \; are \; the \; actual \; values \; and \; \hat{y}_i \; the \; predicted \; values$$

Ridge regression modifies the linear regression cost function by adding a penalty (regularization term) proportional to the square of the magnitude of the coefficients. The objective function to minimize in ridge regression is:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \; where \; \lambda \text{ is the penalty term and } \beta_j \; are \; the \; coefficients \; of \; the \; model$$

The regularization term $\lambda \sum_{j=1}^{p} \beta_j^2$ is added to the RSS to penalize large coefficients, effectively shrinking them towards zero but not setting them exactly to zero.

By adding a penalty to the coefficients, Ridge Regression reduces the variance associated with them, which helps in dealing with multicollinearity. The regularization term helps in preventing overfitting, thus improving the model's generalization to new, unseen data.

To choose the appropriate value for λ I explored a range of λ values and used cross-validation to find the λ that minimizes the prediction error on my training set. Then with the validation set, I predict future values using the estimated parameters. Here are the results:

Same as for the Linear Regression, we notice that there is some dispersion between the actual and predicted prices. To evaluate properly the Ridge Regression model, I look at the coefficient of determination, MSE and MAE. I obtain a **coefficient of determination R^2 of 0.8588, MSE of 621.38 and MAE of 16.83.**

The Ridge Regression did not improve the performance of the model at all. Next, I will perform feature selection using Random Forest and then run a Regression with the selected features.

c) *Random Forest*

Random Forest is an ensemble learning method primarily used for regression tasks. It operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees. It's also useful for feature selection. Feature Selection involves identifying the most

important features in the dataset that contribute the most to the prediction. I chose to use random forest for the following reasons:

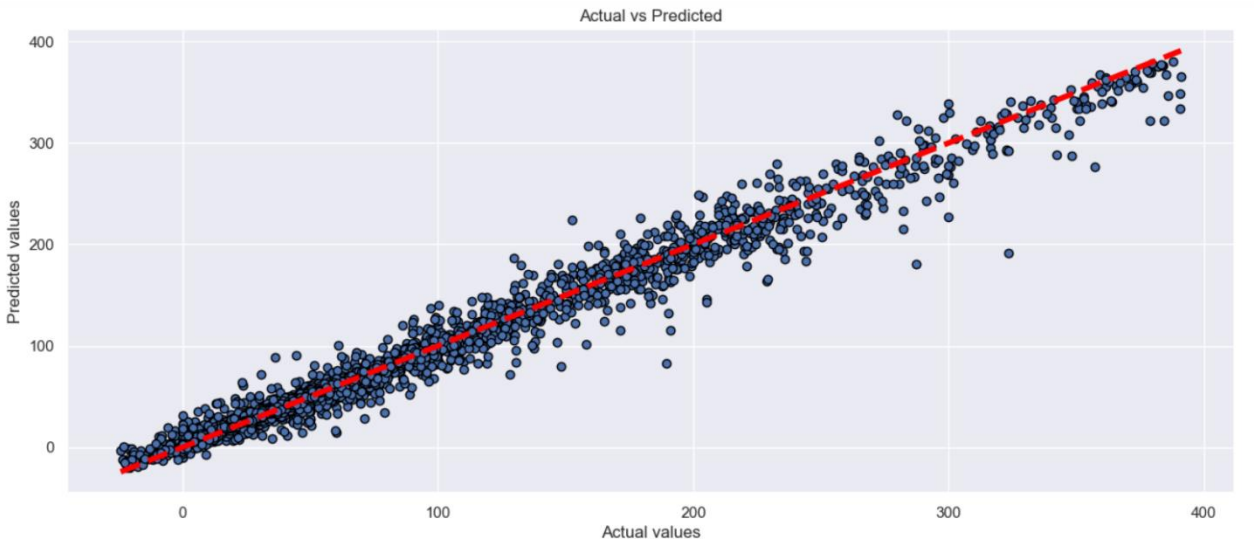
- **Feature Importance Measurement**: Random forest evaluates feature importance by measuring how much each feature decreases the variance in a tree. Features that are frequently used in the decision trees, especially at higher levels (nearer to the root), are typically more important.
- **Handling Multicollinearity**: By considering random subsets of features for each tree, random forests manage multicollinearity better than some other feature selection methods.
- **Reduced Overfitting**: Random forests are less prone to overfitting compared to individual decision trees, making their feature importance scores more reliable.
- **No Independence Assumption**: Unlike linear models that assume feature independence when estimating coefficients, random forests do not require this assumption, providing an advantage in feature selection.

I split my dataset into 80% training set and 20% validation set. Then, I train my Random Forest model on my training set. I compute features importance and rank features based on their importance score. This is the result:

	feature	importance
11	Clean Dark Spread	0.689238
10	Clean Spark Spread	0.087098
4	HardCoal	0.072142
2	NaturalGas	0.037159
5	Nuclear	0.024048
3	Ror	0.019140
9	NetImport	0.017899
13	WIND	0.014262
8	consumption	0.013638
12	TEMPERATURE	0.006947
0	Dam	0.005398
6	WindOnshore	0.004945
7	WindOffshore	0.004084
1	Solar	0.004002

We can see that Clean Dark Spread explains nearly 70% of the variance of the power prices which is huge. To select important features, I define a threshold of 0.01. That is, I will keep the features that I have an importance score higher than 0.01. I decided to choose 0.01 as a threshold, when looking at the table it does not make sense to choose a higher threshold as it will only keep Clean Dark Spread as an important feature and a lower one will keep all the features.

The selected features are: Clean Dark Spread, Clean Spark Spread, HardCoal, NaturalGas, Nuclear, Ror, NetImport, WIND and consumption. Then, I run a Random Forest Regression on my training set comprising only the selected features and I predict the future power prices using the validation set. Here are the results I obtain:



I notice that there is less dispersion between the actual and predicted prices. The predicted prices are much closer to the actual values.

To evaluate properly the Random Forest model, I look at the coefficient of determination, MSE and MAE. I obtain a **coefficient of determination R^2 of 0.978, MSE of 94.66 and MAE of 5.456.**

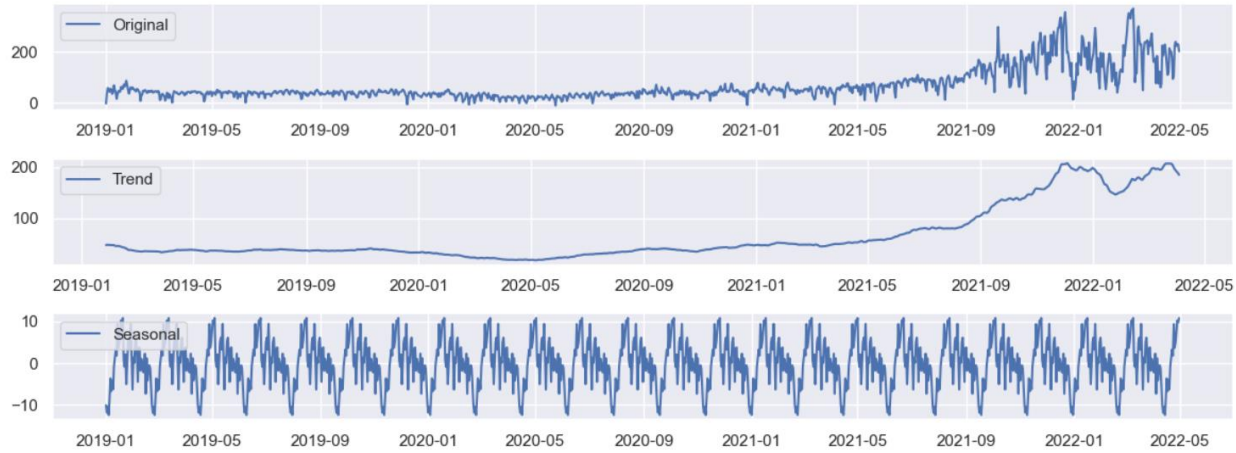
I see a dramatic improvement in the coefficient of determination, mean squared error and mean absolute error compared to linear regression and ridge regression. For now, Random Forest is the best model.

d) ARIMA

The ARIMA model is a popular statistical method used for time series forecasting. It has an AutoRegressive part, an Integrated part and a Moving Average part so it allows to handle a wide variety of time series data, including data with trends, seasonal patterns, and cyclic patterns, to model complex patterns in the data and it provides accurate forecasts.

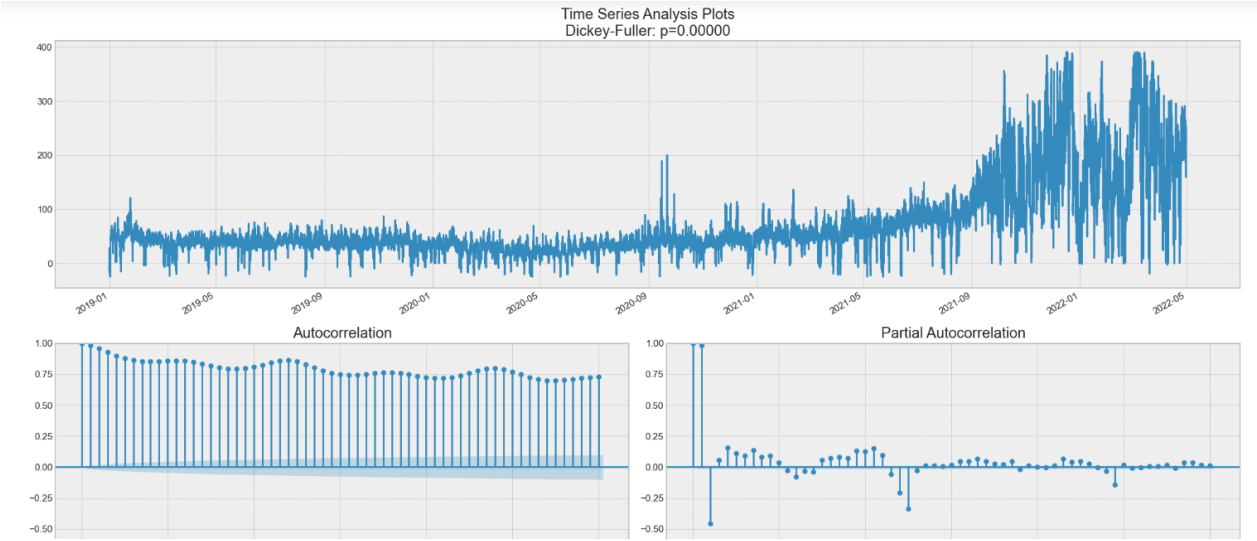
An ARIMA model is generally denoted as ARIMA(p,d,q) where p is the number of lagged values in the autoregression part, d is the number of differencing steps to make the series stationary and q is the number of lagged forecast errors in the moving average part. I need to estimate the parameters p, d and q.

Firstly, I look at the seasonality of my data. I resampled the data to daily data because before doing that I couldn't conclude anything regarding the seasonality as there were too many data points.



By looking at the above graphs, we notice that there is a monthly seasonality in the German Power prices.

Now I'm going to check if the power prices are stationary. I use the Augmented Dickey Fuller test. This tests to see if there is a unit root. The null hypothesis states that there is a unit root.

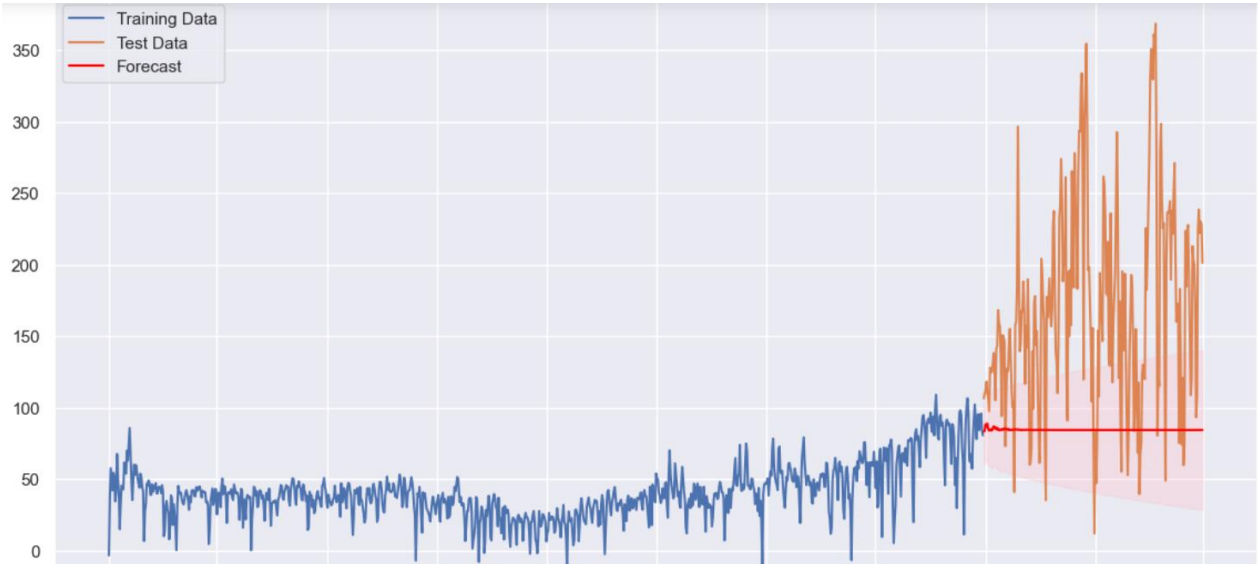


By looking at the graphs, I notice that the time series is stationary (as the p-value is lower than 5% so we reject the null hypothesis which states that there is a unit root) but also on the Autocorrelation graph the blue-marked zone denotes the 95% significance level. It is common for power prices to have highly significant autocorrelation on previous lags. This indicates that all previous lags considered by the acf should be included in the model. On the partial autocorrelation, the blue-marked area denotes the 95% significance level. It becomes clear that significant pacf lags are a bit trickier to spot than significant acf lags.

Now that I know my data is stationary, I need to find the parameters that best fit the ARIMA model. I split my data into 80% training set and 20% validation set. I fit an ARIMA model to my training set to find the parameters p,d and q that best fit my training data. On my first attempt, it made my computer crash

because the dataset is huge so I had to resample my hourly data into daily data to have fewer data points so that my computer could handle it.

Once, I found the best parameters I used my validation test to predict power prices using the ARIMA model. Here are the results:



I got an MSE **of 12449 and MAE of 90**. In red it's the predicted prices while in orange is the true prices. By looking at the plot we can see that there is a huge error between the forecast and true values. This is confirmed by the MSE and MAE scores which are much higher than the previous models. In fact, the ARIMA model is for now the worst model. This can be explained by the fact that I had to resample my data into daily data points, so the model had fewer points to train.

e) Neural Networks

Lastly, I decided to implement Neural Networks models because they can model complex and non-linear relationships between inputs and outputs. This allows them to capture intricate patterns in data that may not be easily captured by traditional linear models. Moreover, Neural networks can scale to handle large datasets and complex problems.

I need to prepare my data to pass into the different Neural Networks I will use.

I split my dataset into 80% training, 10% validation and 10% test. My dependent variable will be hourly power prices and my independent variables will be the rest of the features. I normalize the data set to help accelerate convergence, have better performance and to have a uniform scale because each feature does not have the same scale.

Then I perform PCA to reduce the dimensionality of my training data set while preserving important information. This is how PCA works: PCA computes the covariance matrix of the standardized data, then performs eigenvalue decomposition on the covariance matrix. This process yields eigenvectors and

eigenvalues. Eigenvectors represent the directions (or principal components) of maximum variance in the data, while eigenvalues indicate the magnitude of variance along each eigenvector.

It sorts the eigenvectors based on their corresponding eigenvalues in descending order. The eigenvector with the highest eigenvalue becomes the first principal component, which captures the most variance in the data. The principal components obtained from PCA can be interpreted as new features that are linear combinations of the original features. These components are orthogonal (uncorrelated) to each other, making them suitable for reducing multicollinearity in the data. I've decided to only keep the Principal components that explain 80% of the total variance when I sum their weights.

Now that I have selected the Principal Components I can design my Neural Networks.

1) LSTM

I decided to implement a LSTM NN model. Indeed, LSTMs are designed to capture long-range dependencies within sequences. In my context, this means they can potentially learn and remember patterns, trends, and relationships over extended time intervals, making them effective for forecasting power prices. However, they are not known for their speed.

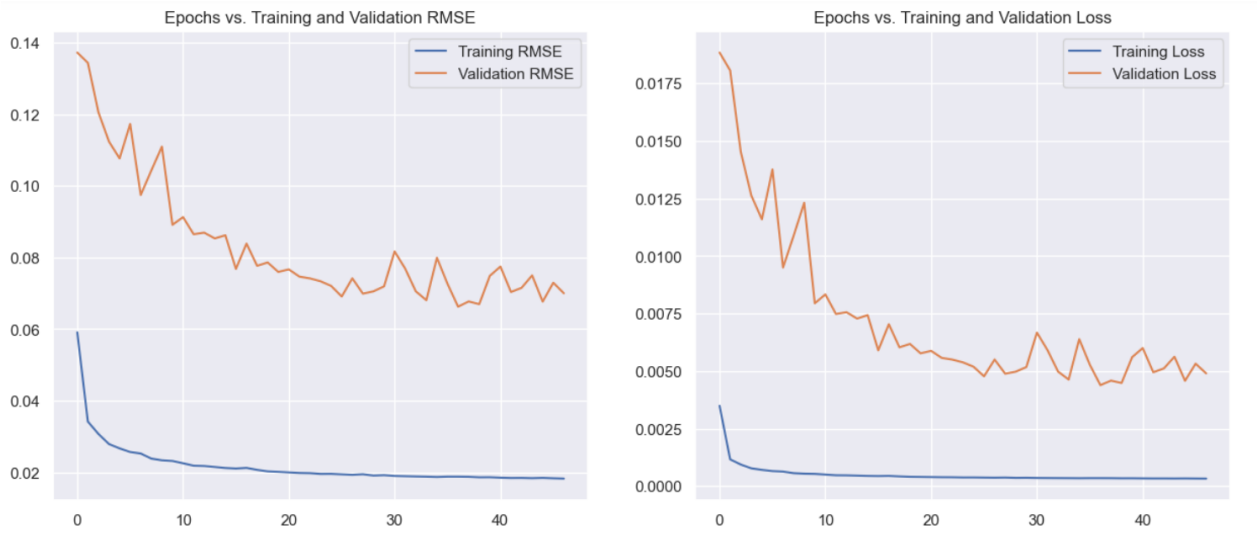
My LSTM model is going to take the last 24 data points for each feature to predict the next one. It makes sense as I have hourly data to take 24 data points to have a full day of data. I decided to choose 100 neurons to have a better performing model.

I decided to choose:

- Adam as the optimizer because it is helpful in training deep neural networks like mine and preventing the learning rate from being too high or too low, which might cause slow convergence or divergence.
- MSE as the loss function because it is suitable for regression problems.

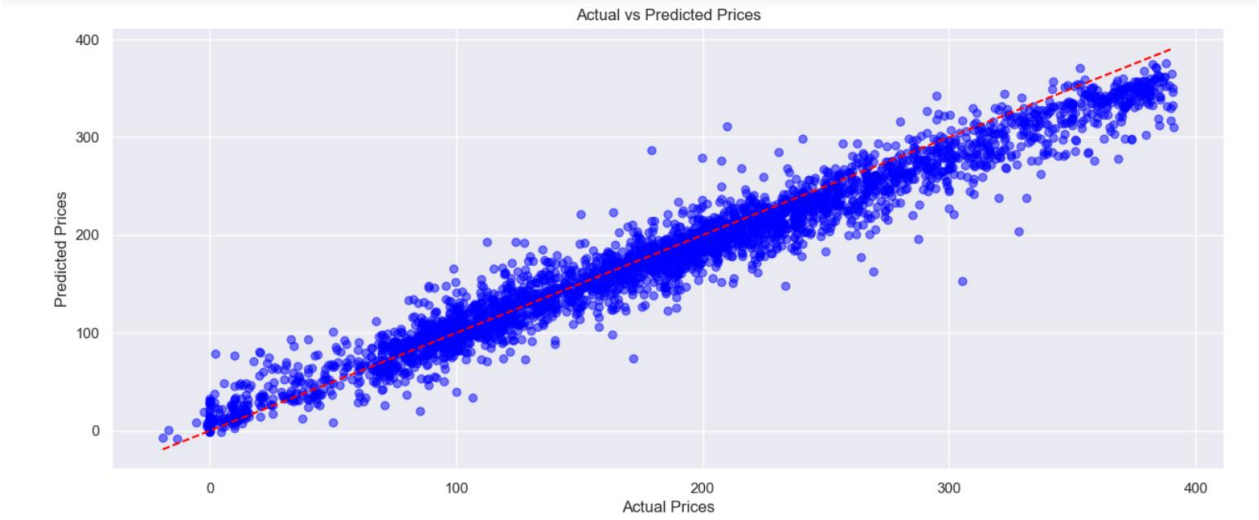
Finally, once the Neural Network has been trained, I use the test set to make predictions using the trained LSTM model and I inverse transform the predicted values to their original scale to make their interpretation easier.
The loss value is decreasing with the number of epochs, slowly converging to 0. The loss value is really low which means that the predictions made by the model align more closely with the actual target values.

However, it's also important for the model to generalize well on new, unseen data. Monitoring both training and validation/test performance is crucial. That's why, I'm going to look at the loss value on validation data.
To do so, I simply plot training and validation loss values and RMSE throughout the epochs to see if they correspond or not. Indeed, overfitting can be indicated by a significantly lower loss on the training set compared to the validation set.



By looking at the above graph, we can see that the 2 curves do not coincide so there is overfitting and we can say that our model does not generalize well and that the model is not really learning the underlying patterns present in the data without specifically memorizing the training set's details.
We now have to test our neural network's pricing capabilities against test set.

To do so, we are going to calculate some performance metrics to assess how well the model performs on the test set and we are going to visualize the predicted values against the actual values in the test set to observe how well the model's predictions align with the true values.



I have a **MSE of 489 and MAE of 16**.

The red dashed line represents a perfect prediction line where actual and predicted values would be identical. By looking at the Actual vs Predicted Prices plot, we can see that points do not really cluster closely around the diagonal red dashed line, indicating that the model's predictions aren't that close to the true values. Moreover, we have relatively high values for both MSE and MAE. A high MSE indicates bad agreement between predictions and actual values while a high MAE also signifies worse model performance. Both MSE and MAE are measures of prediction accuracy. As we have high values for both metrics, we can suggest that our model does not perform very well.

2) *XGBoost*

XGBoost stands for "Extreme Gradient Boosting" and is an efficient and powerful implementation of the gradient boosting framework. It is designed for speed and performance, providing robust tools for model tuning and handling large datasets with complex patterns. I chose to use XGBoost for several reasons:

- **High Performance and Efficiency**: XGBoost is known for its superior speed and performance.
- **Regularization Techniques**: XGBoost incorporates L1 and L2 regularization to prevent overfitting and enhance generalization.
- **Captures Complex Relationships**: Based on gradient boosting, XGBoost effectively captures complex relationships between features and the target variable, making it ideal for regression tasks like mine.
- **Robust to Noisy Data and Missing Values**: XGBoost can handle missing values internally and is less prone to overfitting compared to some other algorithms.

Here's a breakdown of how the algorithm works:

- **Initial Prediction**: The process begins with an initial prediction, usually the mean of the target variable for regression tasks or a uniform probability for classification tasks.
- **Calculate Residuals**: The residuals (errors) between the actual values and the predicted values are calculated for each observation.
- **Train New Tree**: A new decision tree (or weak learner) is trained to predict these residuals, aiming to find patterns in the errors that can improve the predictions.
- **Update Predictions**: The predictions of the new tree are added to the previous predictions to form updated predictions. This iterative process continues, with each new tree improving upon the errors of the combined ensemble of previous trees.

I use the training set to train the XGBoost model and use it to forecast prices using the test set. I got an **MSE of 16209 and MAE of 99**. This is by far the worst of the models I've used.

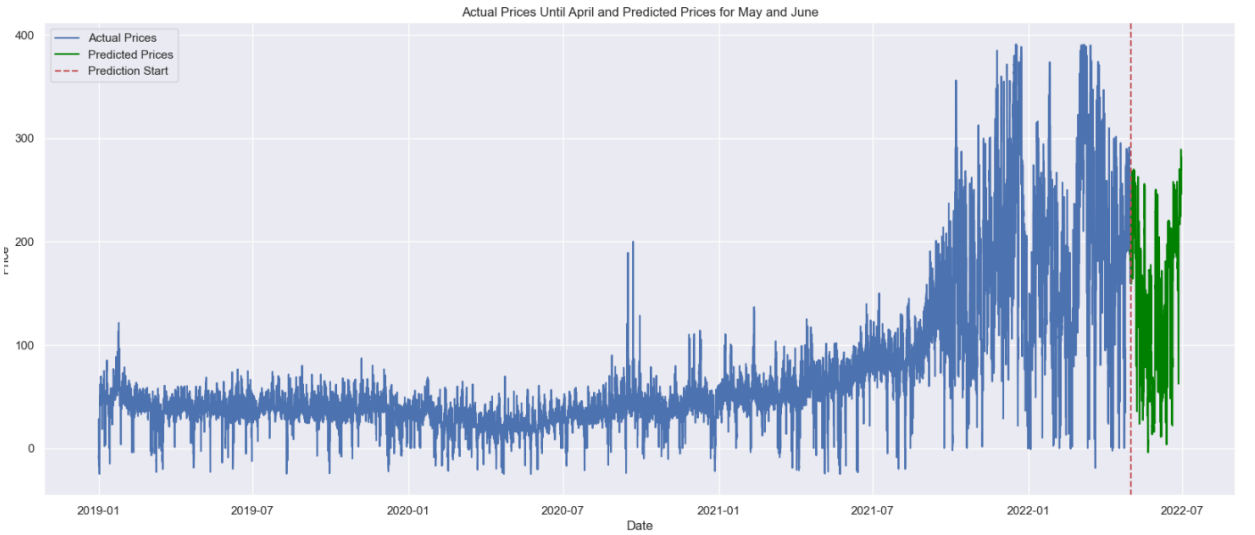
This is the rank of my models based on MSE and MAE:

1) Random Forest
2) LSTM
3) Ridge Regression
4) Linear Regression
5) ARIMA
6) XGBoost

To predict the last May and June 2022 day-ahead electricity prices I am going to use the Random Forest model.

III. Predictions

For this prediction task, I use a new dataset. This time the dataset consists of data from the important features that were selected in the Random Forest feature selection that is Clean Dark Spread, Clean Spark Spread, Hard Coal, Natural Gas, Nuclear, Ror, Net Import, Wind and consumption from 2022-05-01 to 2022-06-29.
As before, I make sure to handle outliers and missing values and I use the Random Forest that I previously compiled to make the predictions for the day-ahead power prices. This is how the result looks:



The predictions seem quite in range with the actual trend. From this prediction, we could implement trades as follows:

- If at time t, our prediction says that at time t+1 the price is going down we take a short position.
- If at time t, our prediction says that at time t+1 the price is going up we take a long position.