APMC/Mandi Machine Learning Challenge

Interpreting the Data & Visualizations

“Stick to the basics”

Monthly_data_cmo.csv

Data given was cleaned to a maximum context, only thing was to create a date time variable with a datetime object.

Redundant column state_name was removed as it has only one unique value.

Since it was a monthly data, it is necessary to look at the count of commodities sold for each month.

For each district, date, Commodity, APMC, and month we calculated mean and median of modal_price, min_price, max_price, arrivals in qtl subsequently.

CMO_MSP_Mandi.csv

Redundant column msp_filter was removed as it has only 1 unique value

Combining both the dataframes

First of all, we did a inner join between the two dataframes, of course which has no NAs.

Dataframe size is of 4120 data points.

Similarly here for each year, and Type we calculated mean and median of modal_price, min_price, max_price, arrivals in qtl subsequently.

Understanding the correlation between msprice and modal price, msprice and min price, msprice and max price, and ms price and arrivals in qtl.

Pearson correlation factor between ms price and modal price, min price, max price is positively correlated, is around 0.9.

Pearson correlation factor between ms price and arrivals in qtl is very negatively correlated, is around -0.004

Understanding variable Month with Type variable, Crops were only found in November month where no. of Kharif crops is 3280, Rabi crops is 835 and other crops is 5.

We also did a left join between the two dataframes, to have a look of missing values.**[not needed]**

Since it is a time series forecasting problem, and we are not assuming it as a a supervised learning problem, I know we can. But not.

So all the variables will not contribute anything for forecasting modal price. Only we need date time and modal price.

Here we noticed that each combination of APMC and commodity, we have to separately design a forecasting model. Since we have a lot of combinations of APMC and commodity.

And of course one tuned forecasting model for one data/problem may not be used for other data/problem.

So here we cannot just put one forecasting model for all the different problems.

So we filter out variable APMC as ‘Ahmednagar’ and Commodity as ‘Bajri’ and lets design best tuned forecasting models on this.

We saw the correlation factor between modal price and min price, max price and arrivals in qtl. What we got is modal price is linearly dependent with min price and max price with correlation factor more than 0.9 and modal price is negatively correlated and not linearly dependent where correlation factor is -0.24.

In the filtered out data, where we are going to design forecasting models, it has 22 data points and we have arranged it into a form of date, we will take first 15 data points as training data and remaining 7 data points as testing data for all the forecasting models so that we can compare all of them on common front.

Time Series Forecasting methods

“Be fundamental”

I will try to explore different ways of doing time series forecasting here from naive approaches to deep learning models

Naive approach

Here, we take the last month value and estimate the same value for the next month.
We got root mean squared error as 186.65

Simple average

Here we take the average of all previously observed points to forecast for the next month
We got root mean squared error as 259.289

Moving average

Here we forecast the next value based on the average of a fixed finite number ‘p’ of the previous values.
We got root mean squared error as 197.86

Simple Exponential Smoothing

Here we calculate forecasts based on weighted averages where the weights decrease exponentially as observations come from further in the past, the smallest weights are associated with the oldest observations.
Used a common approach of removing trend with differencing and then applied Simple Exponential Smoothing
Key point is that Simple Exponential smoothing is suitable for data with no trend and seasonality.

While in coding part the focus was on tuning the smoothing parameter α to determine the weight of each observation.
There are some basic intuitions behind the value of α:
When α is closer to 0 we consider this *slow learning* because the algorithm gives historical data more weight.
When α is closer to 1 we consider this *fast learning* because the algorithm gives more weight to the most recent observation

Here for our data we got α as 0.003

We got root mean squared error as 161.60655

Simple Exponential proves to be better than above forecasting techniques.

Holt’s Linear Trend Method

We observed that above all methods, they aren’t taking account the trend. They are basically generalizing based on assumptions.

Now, we applied a method that can map the trend accurately without any assumptions which is Holt’s linear trend method.

Here we have two smoothing parameters α and β, which correspond to the level and trend components (Note all above techniques correspond to just level components).

While in coding part we tuned both the smoothing parameter α and β with 0.99, 0.0621 respectively

But we got a root mean squared error as 187.52 which is worse than simple exponential smoothing.

This suggests that we have very less data points so holt is not capable to make use of detecting trend mechanism.

Less data points - less trend to find.

Holt Winters Method

Here we consider a new term which is seasonality

Here we have a facility of adding multiplicative or additive forecast to the holt model.
When the trend increases or decreases linearly, additive equation is used whereas when the trend increases of decreases exponentially,

Since our data shows more of an additive trend. So additive gave a very slightly better result than multiplicative.

Holt winter takes into account of both seasonality and trend.

But we got a root mean squared error as 238.78 which is worse than all the above forecasting models.

Major reason is that there is a very little chance of seasonality ocurring within 15 training data points. So holt winter completely focuses on trend and even seasonality. So it performs worse than holt forecasting models.

ARIMA

Used tsclean() for outlier removal.

Any time series analysis has seasonality, trend and cycle, may be not all.

Deconstructing the series can help us understand its behavior and prepare a foundation for building a forecasting model.

We calculated seasonal component of the data using `stl()`. STL is a flexible function for decomposing and forecasting the series. It calculates the seasonal component of the series using smoothing,

We used seasadj() to decompose the series and removed the seasonality by subtracting the seasonal component from the original series.

We used Autocorrelation functions(ACF) and Partial Autocorrelation Function(PACF) to determine q (moving average) and p (Auto regressive) respectively, i.e. parameters of the arima model.

ACF plots display correlation between a series and its lags. Whereas, PACF display correlation between a variable and its lags that is not explained by previous lags.

We got q=1 and p = 2.

Still we got a root mean squared error as 187.88

Facebook prophet

Prophet was developed on top of holt winter and Arima where it has three main model components: trend, seasonality, and holidays.

Modelling seasonality in prophet is same as Holt winter technique.

Without any parameters, prophet gave root mean squared error as 210.32

i) Saturating Forecast

We have log transform the target variable to eliminate trend and it gave better performance than the normal one.

Prophet has a cool characteristics to understand the non linear trend in the data since our data has non linear characteristics over time

By default, Prophet uses a linear model for its forecast. When forecasting growth, there is usually some maximum achievable point: total market size, total population size, etc. This is called the carrying capacity, and the forecast should saturate at this point.

We got a very little improvement, root mean squared error reduces to 209.75.

ii) Changepoints

Often we see in any time series data, it undergoes frequent changes, like a zig-zag. At such points, the growth rate is allowed to change. We can fix a list of changepoints in prophet.

With no of changepoints, we have changepoint_prior_scale to adjust the trend flexibility and solves the problem of overfitting and underfitting.

We have not got any improvement with this approach and root mean squared dropped down to 211.

iii) Holidays

Prophet has a special functionality to understand shocks in a time series data which happens mostly due to holidays and other reasons.

But without any domain knowledge about the data, I was not able to use this functionality of prophet

Deep Learning and LSTM/RNN

Firstly, we phrased the problem as a regression problem.

LSTM Data Preparation:

- Transform the time series into a supervised learning problem
- Transform the time series data so that it is stationary.
- Transform the observations to have a specific scale.

Transform the time series into a supervised learning problem

The LSTM model in Keras assumes that our data is divided into input (X) and output (y) components.

For a time series problem, we can achieve this by using the observation from the last time step (t-1) as the input and the observation at the current time step (t) as the output.

We achieved this using the shift() function in Pandas that pushed all values in a series down by a specified number places. We required a shift of 1 place, which will become the input variables.

We then concatenated these two series together to create a DataFrame ready for supervised learning. The pushed-down series have a new position at the top with no value. A NaN (not a number) value will be used in this position.

We replaced these NaN values with 0 values, which the LSTM model will have to learn as the start of the series or have no data here.

Transform Time Series to Stationary

Of course our data is not stationary.Stationary data is easier to model and will very likely result in more skillful forecasts.

The trend can be removed from the observations, then added back to forecasts later to return the prediction to the original scale and calculate a comparable error score.

We have standard way to remove a trend is by differencing the data. That is the observation from the previous time step (t-1) is subtracted from the current observation (t). This removes the trend and we are left with a difference series, or the changes to the observations from one time step to the next.

Transform the observations to have a specific scale.

LSTMs are sensitive to the scale of the input data, specifically when the sigmoid (default) or tanh activation functions are used. It can be a good practice to rescale the data to the range of 0-to-1, also called normalizing.

We normalized our dataset using the MinMaxScaler preprocessing class from the scikit-learn library.

Splitting Data into train and test

After we model our data and estimate the skill of our model on the training dataset, we need to get an idea of the skill of the model on new unseen data. For a normal classification or regression problem, we would do this using cross validation.

With time series data, the sequence of values is important. The first 67% rows(15 data points were used as a training dataset and the the remaining 33% rows(7 data points) were used for testing dataset.

Finally coming to the design of different LSTM models used

i) LSTM for Regression Using the Window Method

We phrased the problem so that multiple, recent time steps can be used to make the prediction for the next time step.

This is called a window, and the size of the window is a parameter that can be tuned for each problem.

When phrased as a regression problem, the input variables are t-2, t-1, t and the output variable is t+1.

ii) LSTM for Regression with Time Steps

We have noticed that the data preparation for the LSTM network includes time steps. Some sequence problems have a varied number of time steps in a sample.

Time steps provide another way to phrase our time series problem. Like above in the window example, we can take prior time steps in our time series as inputs to predict the output at the next time step.

Instead of phrasing the past observations as separate input features, we can use them as time steps of the one input feature, which is indeed a more accurate framing of the problem.

We just reshaped the data, i.e., set the columns to be the time step dimensions (timesteps is not 1 anymore) and change the features dimension back to 1.

iii) LSTM with Memory Between Batches

The LSTM network has memory, which is capable of remembering across long sequences.

Normally, the state within the network is reset after each training batch when fitting the model, as well as each call to model.predict() or model.evaluate().

We can gain finer control over when the internal state of the LSTM network is cleared in Keras by making the LSTM layer stateful. This means that it can build state over the entire training sequence and even maintain that state if needed to make predictions.

It requires that the training data not be shuffled when fitting the network. It also requires explicit resetting of the network state after each exposure to the training data (epoch) by calls to model.reset_states(). This means that we must create our own outer loop of epochs and within each epoch call model.fit() and model.reset_states().

We designed a common function where we incorporated window, timesteps and memory methods.

We designed four different types of LSTM models,

i) ***Window style with memory:*** We got a mean squared error as 52.86

ii) ***Window style with no memory:*** We got a mean squared error as 23.79

iii) ***Timestep style with memory:*** We got a mean squared error as 44.9

iv) ***Timestep style with no memory:*** We got a mean squared error as 26.82

Surely, all LSTM models have surpassed all the above forecasting techniques.

Hence, LSTM Window style with no memory is the best forecasting model in our data

Forecasting Models	RMSE
Naive approach	**186.65**
Simple average	**259.289**
Moving average	**197.86**
Simple Exponential Smoothing	**161.6**
Holt's Linear Trend Method	**187.52**
Holt Winters Method	**238.78**
Arima	**187.88**
Facebook prophet	**209.75**
Deep Learning and LSTM/RNN	**23.79**

Conclusion

Well, it was a great journey of about exploring about different forecasting algorithms. Each of them has their own advantages and disadvantages.

In Data Science, every time you work with an algorithm on a dataset, you know a bit more about the algorithm and how to use it more better.

We see a bunch of new algorithms coming up in Data Science/Machine Learning for all sorts of problems in Data Science, it becomes very important to adapt, understand and apply them on your problems.

“Constant improvement”