Machine Learning Engineer Nanodegree

Capstone Project

Gurnek Sandhu
September 25th, 2020

I. Definition

Project Overview

The purpose of this capstone project was to predict the future climate of the city of Vancouver based on a dataset retrieved from Kaggle which gave us Canadian weather data from the past eighty years.

The dataset has a very detailed set of data which I used to train a DeepAR model and predict the temperature in Vancouver, from 2010-2020

Dataset download: Canadian Climate Data

Problem Statement

In this capstone project, we used deep learning in the form of the DeepAR network from AWS to predict the temperature of Vancouver from 2010-2020 based on the data from 1940-2009

Outline of steps:

#1) The full dataset which was downloaded from Kaggle contains data for all the different major weather centers in Canada from the 1940s and onward. I only wish to find the temperature for Vancouver, so I will have to get rid of all the other columns but LOCAL_DATE and MEAN_TEMPERATURE_VANCOUVER.

Dataset download: Canadian Climate Data

#2) Next I will preprocess the data. This means removing all the NAN data points in the dataset and replacing it with the mean of the column to get the most accurate result.

#3) I then am going to do the moving average part of this project, which is predicting the 2010-2020 temperature. I am then going to record the mse of the test set predictions and start with the machine learning.

#4) I trained the sagemaker DeepAR model based on past data (1940-2009) and estimated the temperature for the test set.

Metrics

The model was evaluated on the mean squared error (dividing the sum of squares of the residual error by the degrees of freedom). The smaller the mean squared error, the more accurate our model.

I chose the Mean squared error over the mean absolute error as the mean absolute error fails to punish large errors in predictions.

II. Analysis

Data Exploration

Dataset download: [Canadian Climate Data]

This is a very large dataset with over 27 columns and 29222 rows. We, however, only used 2 columns and 29222 rows.

This is a small sample of how the dataset looks like.

	LOCAL_DATE	MEAN_TEMPERATURE_CALGARY	TOTAL_PRECIPITATION_CALGARY	MEAN_TEMPERATURE_EDMONTON	TOTAL_PRECIPITATION_EDMONTON
0	01-Jan-1940 00:00:00	-11.4	0.5	2.311584	1.246239
1	02-Jan-1940 00:00:00	-12.0	0.5	2.311584	1.246239
2	03-Jan-1940 00:00:00	-12.0	1.0	2.311584	1.246239
3	04-Jan-1940 00:00:00	-11.4	0.8	2.311584	1.246239
4	05-Jan-1940 00:00:00	-13.1	0.5	2.311584	1.246239
5	06-Jan-1940 00:00:00	-9.8	0.0	2.311584	1.246239
6	07-Jan-1940 00:00:00	-11.4	0.0	2.311584	1.246239
7	08-Jan-1940 00:00:00	-11.1	1.0	2.311584	1.246239
8	09-Jan-1940 00:00:00	-15.6	0.0	2.311584	1.246239
9	10-Jan-1940 00:00:00	-10.6	1.3	2.311584	1.246239

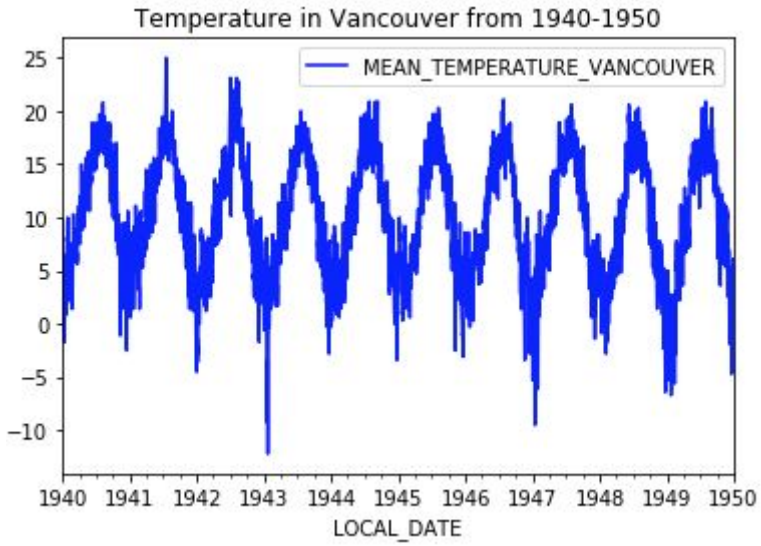
We can see that there is a LOCAL_DATE column that gives us the date, month and year of when each data point was recorded. We also see that there is MEAN_TEMPERATURE data and TOTAL_PRECIPITATION data for 2 out of 13 major Canadian weather stations in the dataset. As this is a small example, we only get samples for 2 weather stations, however, there is data for 13 weather stations. They are listed below.

CALGARY, EDMONTON, HALIFAX, MONCTON, MONTREAL, OTTAWA, QUEBEC, SASKATOON, ST JOHN'S, TORONTO, VANCOUVER, WHITEHORSE, and WINNIPEG

For some of the weather stations, such as Edmonton, Halifax, Montreal, Quebec, St. Johns and Whitehorse, there is no weather or precipitation data until as late as 1959. However, we aren't going to use those columns for our project.

There are many columns in the dataset, giving daily precipitation and temperature data for the 13 major Canadian weather centers. We only use the LOCAL_DATE column and the MEAN_TEMPERATURE_VANCOUVER column for our model as we only want to predict the temperature for the Vancouver area and need the date and time of when each prediction was recorded.

There are not any major outliers, just some cold winters and hot summers, but they are in a seasonal pattern. The data as a whole is very seasonal and follows a pattern that is very clear to the human eye. There is a time plot of the data from 1940-50 below.



There are also some nan values. Those would give problems when I tried to plot my data, so I will them with the mean of my dataset

Algorithms and Techniques

The model I am going to use to predict the output is the DeepAR algorithm, built into amazon sagemaker for use by its users. Here is a summary of what it is in short. The information came from AWS Sagemaker documentation

The neural network I am using is the Amazon DeepAR algorithm

What is the DeepAR algorithm?

The DeepAR forecasting algorithm is a supervised learning algorithm for forecasting a one-dimensional time series using an RNN.

What is the training input?

The training input for the DeepAR algorithm is one or more target time series that the same/similar processes have created. The algorithm trains a model that learns an approximation of this process/processes based on this input dataset and uses it to predict how the target time series evolves. A vector of static (time-independent) categorical features provided by the catfield and a vector of dynamic (time-dependent) time series provided by the dynamic field can be optionally associated with each target time series. By randomly sampling training examples from each target time series in the training dataset, SageMaker trains the DeepAR model. A pair of adjacent background and prediction windows with set predefined lengths are composed of each training example.

Technique:

The information below came from Towards Data Science, 10 machine learning Methods and Analytics Vidhya, ANN vs CNN vs RNN.

As the DeepAR algorithm utilizes an RNN, the machine learning technique I am using is “neural networks and deep learning.” An RNN(recurrent neural network) is one of the many different types of neural networks in deep learning, along with CNN’s (convolutional neural networks), and ANNS (artificial neural networks)

Neural networks and deep learning are important kinds of machine learning techniques. They are very widely used and are heavily successful in areas of vision, such as image classification, text, audio and video.

Benchmark

The benchmark result will be the data from 2010-2020. The models I use will train using the data from 1940-2009 and will test on the 2010-2020 data. I will first start with the moving average to get accuracy. It will be the benchmark for model accuracy.

I will keep trying to be as accurate as I can without overfitting. The final model will have to be as accurate as possible and more accurate than the moving average to be considered production-ready.

III. Methodology

Data Preprocessing

I did all the same steps that I had previously outlined, like removing all the nans from the data and replacing them with the mean, and also getting rid of the columns but the LOCAL_DATE and MEAN_TEMPERATURE_VANCOUVER, however, there were a couple of extra things I had to do as well to feed the data to the model.

One thing I noticed when trying to plot the data for the first time was that the LOCAL_DATE column was not in pandas DateTime format, so the month and year of the datapoint would not show up in the x-axis. This made it hard to recognize when each data point was recorded. I had to manually convert the datatype of the LOCAL_DATE column from string to pandas DateTime format to be able to see the month and year in the x-axis. This made understanding the visualizations much easier.

Another thing I did that I forgot to think I would do was normalize the data. It would help tremendously in making the model more accurate as the distance between the different data points would be greatly reduced, and the shape of the data would still be maintained.

I also had to make train and test series out of my data using a function that I used from the practice notebooks from the course

Before I fed the data to my model, I also had to make a JSON dataset out of the train and test time series and save them to an s3 bucket before I could feed the data to my model to train

Implementation

When I implemented the model, I used the same model as I stated I was going to be using, the DeepAR model from Amazon Sagemaker. As I had used this model many times during the practice notebooks, I thought this project was going to be fairly easy. However, during implementation, there were many problems.

First I tried a very simple model with only 2 layers and 4 cells. But the model was not accurate at all, with it failing to capture the seasonality of the data. I tried to replace the simple, easy to run model with a more complex one which I hoped would capture the trend of the data. However, when I tried to make the model more complex, it was taking too long to respond and give me a prediction. I then tried a batch transform, but that also had its issues.

I then tried a batch transform so it would not have that problem of the model not responding in time. However, when I tried to download the prediction from the model, I face another error. It turned out that I had to use os.path.join to fix the error.

As the model took a JSON file as input and also outputted a JSON file, I had to convert the output JSON file to a CSV using the pandas.read_json() function. However, when I tried to use the function to convert the file, I got an error saying the function failed to read a particular character to CSV. I just ended up using read_csv instead

When I used pd.read_csv(), it sort of worked, but there were still characters that the JSON had added, such as square brackets, parenthesis, and more. I ended up using the .replace() to just get rid of those characters from each data point in the dataset. That was the solution that worked for me and allowed me to get an MSE score.

Speaking of MSE, since the time series was 10 years long, split into individual time series of 1 year each, I had to get the mean mse of the 10 individual years of the time series. I added the mse of each year together and divided it by the number of years(10), and I got a mean mse of 0.19, which was not great, however, it was the first problem I solved on my own, so that’s a win in my books. I’ll get better over time.

Refinement

I had to make some refinements to the hyperparameters of the model when the first very simple model I tried, was not able to capture the trend of the data. I tried many different combinations of hyperparameters to get the model as accurate as possible. I went through many different models changing the # of epochs in a range from 20-400, cells from 1-400, layers from 1-10, and prediction length, from 166 and context length from 0-166, and found my current model I then found the model that I have currently, the one trained with 263 epochs, 4 context_length, 166 prediction_length, 2 layers, and 177 cells. This was the model that worked the best for me and that gave me the test accuracy best output. I got rid of the other models, as their output took up lots of space and it took a long time to scroll down and made it hard to keep track of different things in the notebook.

IV. Results

Model Evaluation and Validation

The DeepAR model I used had been trained with 263 epochs of the training data, the context_length was set to 4, the prediction length was at 166, and it had 177 cells.

The DeepAR I used, with its different hyperparameters, beat the moving average benchmark.

I was planning to go even further and create a custom model myself using CNN layers instead of RNNs like DeepAR uses. That is why I couldn't get more accurate. However, I was limited by the amount of time given for this project. I thought I would be able to make one, however, there were many complications with the implementation of my DeepAR model, and the deadline was fast approaching.

I also wanted to try and test the model using different data to test the robustness, however, I ran out of time and the deadline was fast approaching. So I could not do the sensitivity analysis I wanted to do.

Justification

The DeepAR model with all of its optimal hyperparameters which were tuned for accuracy with the dataset it was trained with, got an MSE of 0.0195, which was marginally better than the 0.028 achieved by the moving average. Therefore, the results I have achieved have beaten the benchmark model and the problem has been solved.

However, I would have liked to try and create my own model, but the deadline was fast approaching and I ran out of time.

V. Conclusion

Reflection

In this project, We used neural networks and did deep learning in the form of the DeepAR model to try and accurately predict temperatures in Vancouver.

I had to preprocess the data for use by the model by getting rid of nans, converting the LOCAL_DATE column to pandas DateTime, making train and test time series out of the data and converting the 2-time series to JSON so the data could be used by the model for training and testing.

I went into this project thinking it was going to be very easy and I was going to get it done a week or so before the deadline, however, I was very wrong, as I had many problems when implementing the model.

When I implemented the model, I used the same model as I said I was going to be using, the DeepAR model from Amazon Sagemaker. As I had used this model many times during the practice notebooks, I thought this project was going to be fairly easy. However, during implementation, there were many problems, here are the 2 biggest problems I faced during implementation written below.

First I tried a very simple version of the DeepAR model with only 2 layers and 4 cells. But the model was not accurate at all, with it failing to capture the seasonality of the data. I tried to replace the simple, easy to run DeepAR model with a more complex DeepAR model with more layers and cells which I hoped would capture the trend of the data. However, when I tried to make the model more complex, it was taking too long to respond and give me a prediction. I then had to use a batch transform to make the model work.

As the model took a JSON file as input and also outputted a JSON file, I had to convert the output JSON file to a CSV using the pandas.read_json() function. However, when I tried to use the function to convert the file, I got an error saying the function failed to read a particular character to CSV. I just ended up using read_csv instead and removed the extra characters added by the conversion to JSON, that weren't removed by the pd.read_csv() function.

Though I think the DeepAR model did pretty good, I think to solve these kinds of problems, one should use a model with CNN layers instead of the RNN DeepAR uses. This would improve the accuracy of the model for large datasets. I would have done this if it wasn't for the deadline of the project.

Improvement

I would have made some improvements to this project that would have greatly improved if I had more time. The deadline was the main bottleneck for me.

For example, I would have tried to make my own model that used CNN layers instead of the RNNs used by the DeepAR model. That would have allowed me to make a much more accurate model that would be better suited for a production environment.

I definitely think that a way better solution to the DeepAR model exists, and I would have tried to look for it if I had time. And if I wasn't bound by the deadline of this project, I am definitely sure that my accuracy would have been much higher, perhaps in the 0.009 range, much better than the 0.019 I achieved with the DeepAR model.