# Machine Learning Engineer Nanodegree

## Capstone Project - DEEPAR TIMESERIES SNOW DEPTH PREDICTION

Ian Denness
August 14th, 2021

# I. Definition

https://github.com/boffyd/UdacityML-Capstone

## Project Overview

Predicting the level of snow within a ski resort during winter would allow pundits to monitor the season based on current levels to better plan their trip. Currently there are only short-term snow fall predictions ~7 days put out by the weather bureau.

As such, its proposed for this assignment is to explore natural snow depths within the Australian Alpine region. Ultimately, the intent would be to provide an interface that that will use the current seasons results to predict the natural snowfall levels.

The dataset utilised is sourced from data.gov.au, Victorian Alpine Resorts - Daily Snow Depth Records Falls Creek

*https://arcc.vic.gov.au/wp-content/uploads/2021/06/Data-2020-Daily-Snow-Depth-Records_Falls-Creek.csv*

## Problem Statement

Can Time Series modelling be used to predict a ski resort natural snow depth. This would take historical information (historical data) and try and predict the future output based on current season information. This would rapidly update as the season unfolded. The advantage of hindcasting to compare against previous years is that snow seasons year on year can be radically different.

Its proposed to use AWS SAGEMAKER DEEPAR for the process. Its set up for learning from many different time series, in this case different seasons. DEEPAR is noted to be good at learning seasonal dependencies, so its proposed that its quite well suited for predicting the dataset. Its noted that this is a univariate process, and at the moment the prediction will only be based on historical information.

The process will consist of the following process.

1. *Download the dataset*
2. *Check data for consistency and missing values, treat accordingly*
3. *Determine the seasonal range of the dataset*
4. *Determine baseline model = Average Daily Values across all years*
5. *Create Test and Train Datasets*
6. *Convert datasets to Json Format as required by DEEPAR*
7. *Upload to AWS S3*
8. *Train the DEEPAR Estimator*
9. *Test the Estimator and compare against baseline model. Is the modelling a better predictor than just using data set averages?*
10. *Deploy the model.*

## Metrics

The model will use Root Mean Squared Error to compare predicted outputs from DEEPAR against the baseline model. This metric compares the error of the predicted values against actual values. It is also a defining metric for the algorithm which uses this metric to tune and select the appropriate model comparing the test and train split.

# II. Analysis

## Data Exploration

The data downloaded and presented only includes only the selected snow season ranges, which makes sense, essentially you would except to have no snow fall outside of this window (particularly in Australia).

The dataset has one variable which is snow depth, and a corresponding date (daily).

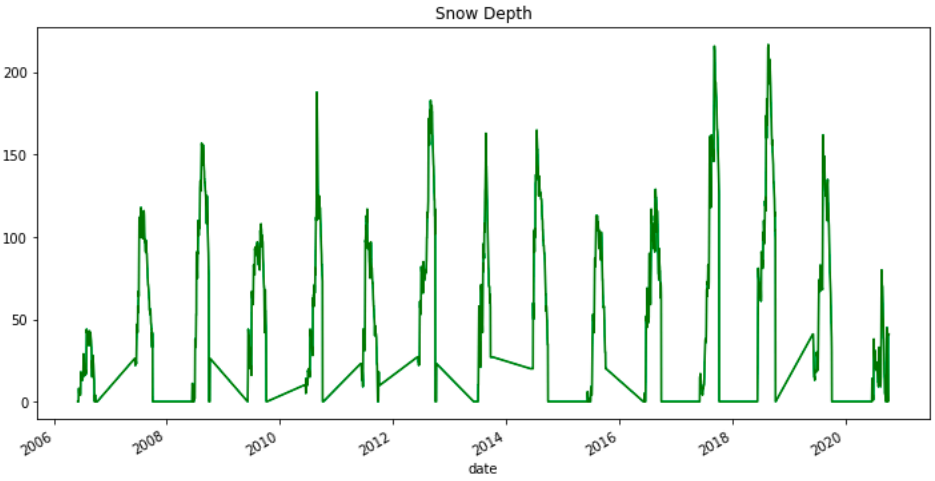
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **date** | 2020-06-05 00:00:00 | 2020-06-06 00:00:00 | 2020-06-07 00:00:00 | 2020-06-08 00:00:00 | 2020-06-09 00:00:00 | 2020-06-10 00:00:00 | 2020-06-11 00:00:00 | 2020-06-12 00:00:00 |
| **snow_depth** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

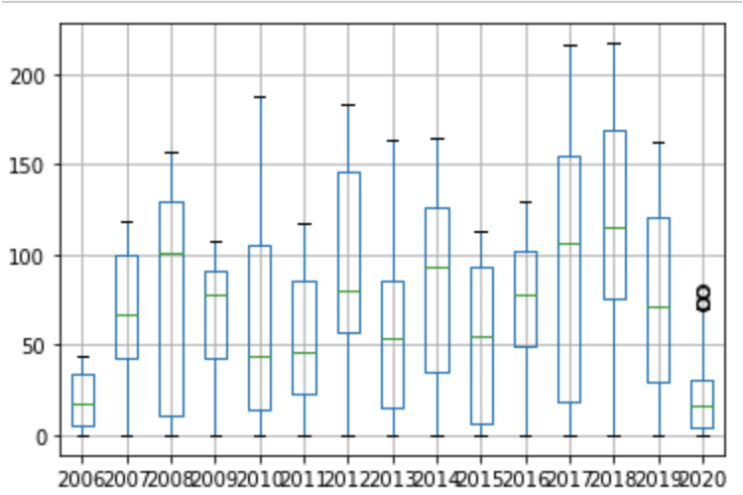
There were missing numbers identified and instead of using mean or median as a replacement the imputation method selected was ffill (or forward fill) which uses the last observed non-null value, in combination with bfill (back fill) which propogates the first observed data point backward. This makes sense that if there was no snow one day, that its more likely that there is no snow the next day and not 67mm (the average across the entire dataset).

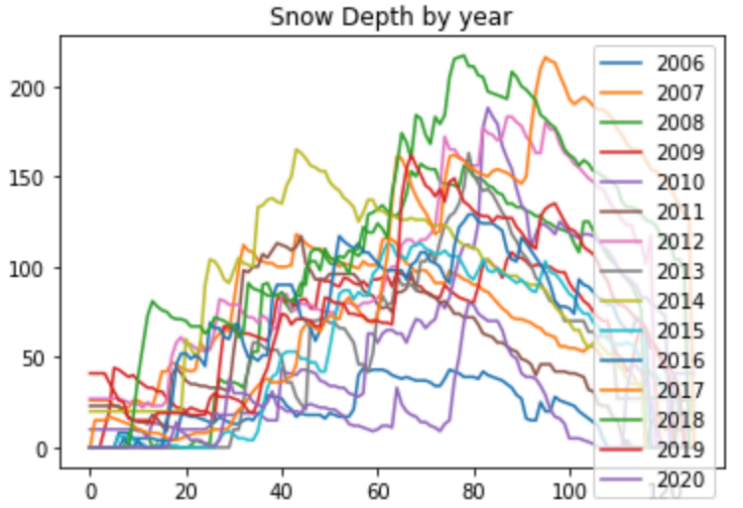
The statistical spread of the data is referenced below.

```
count    1905.000000
mean       67.465092
std        51.858718
min         0.000000
25%        20.000000
50%        66.000000
75%       103.000000
max       217.000000
Name: snow_depth, dtype: float64
```

## Exploratory Visualization




Missing values where treated as discussed, and there were limited numbers of outliers as shown by the above box plot. These were left untreated and treated during modelling as they were only present in one year (known to have issues with COVID). 2020 was exluded from the training set and 2019 kept as a validation set.

## Algorithms and Techniques

The timecasting forecast method for the project was DEEPAR which is a Recurrent Neural Network (RNN). This is a supervised learning algorithm for univariate (one dimensional data). It is particularly well suited to seasonal or repetitive time series e.g. weather or sales forecasts of core products.

AWS literature <https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html> defines the training process as:

*The training input for the DeepAR algorithm is one or, preferably, more target time series that have been generated by the same process or similar processes. Based on this input dataset, the algorithm trains a model that learns an approximation of this process/processes and uses it to predict how the target time series evolves.*

DEEPAR Key Hyperparameters are:

| | |
|---|---|
| **prediction_length** | The number of time-steps that the model is trained to predict, also called the forecast horizon. |
| **context_length** | The number of time-points that the model gets to see before making the prediction. |
| **epochs** | The maximum number of passes over the training data. Typical values range from 10 to 1000. |
| **time_freq** | The granularity of the time series in the dataset.<br><br>M: monthly, W: weekly, D: daily, H: hourly, min: every minute |
| **learning_rate** | The learning rate used in training. Typical values range from 1e-4 to 1e-1.<br><br>Default value: 1e-3 |
| **mini_batch_size** | The size of mini-batches used during training. Typical values range from 32 to 512.<br><br>Default value: 128 |
| **num_cells** | The number of cells to use in each hidden layer of the RNN. Typical values range from 30 to 100.<br><br>Default value: 40 |

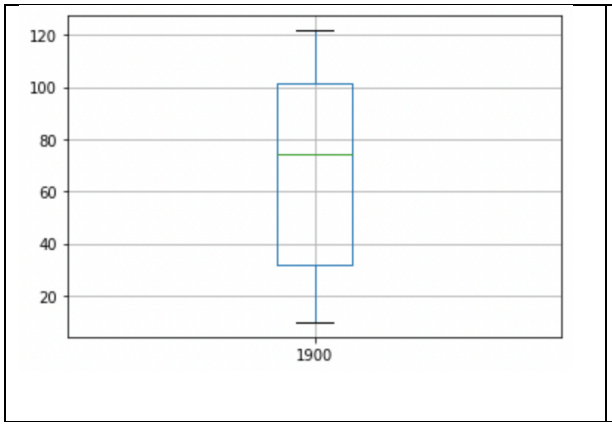
| | |
|---|---|
| **num_layers** | The number of hidden layers in the RNN. Typical values range from 1 to 4.<br><br>Default value: 2 |
| **test_quantiles** | Quantiles for which to calculate quantile loss on the test channel.<br><br>Default value: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] |

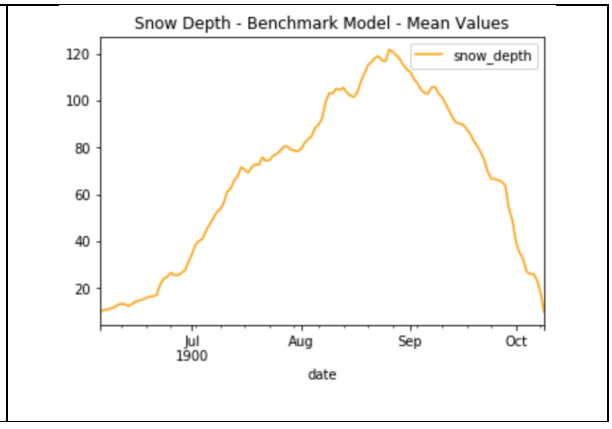
The data is converted to a json format and fit to the modelling estimator.

## Benchmark

Its proposed for the purposes of the exercise to create a benchmark dataset based on the median results across all years and compare this to the model. They hypothesis is that for highly seasonal data such as alpine snow fall an advanced algorithm such as Recurrent Neural Networks will be better than just providing the mean.

Its noted that the proposed algorithm utilises SAGEMAKERS DEEPAR which is a univariate machine learning timeseries technique. If the median results are more accurate, this would indicate that either the hyperparameters need more optimisation, there aren't enough datapoints in the dataset or alternatively the variation associated with the output has more independent variables for consideration. As such this may lead to explore alternate methods to explore.



# III. Methodology

## Data Preprocessing

The data required for the proposed algorithm is univariate time series and as such consideration of feature engineering has not been considered, nor has including more independent variables for process. The pre-processing steps involved downloading the data, tidying the data and treatment of missing variables as detailed below.

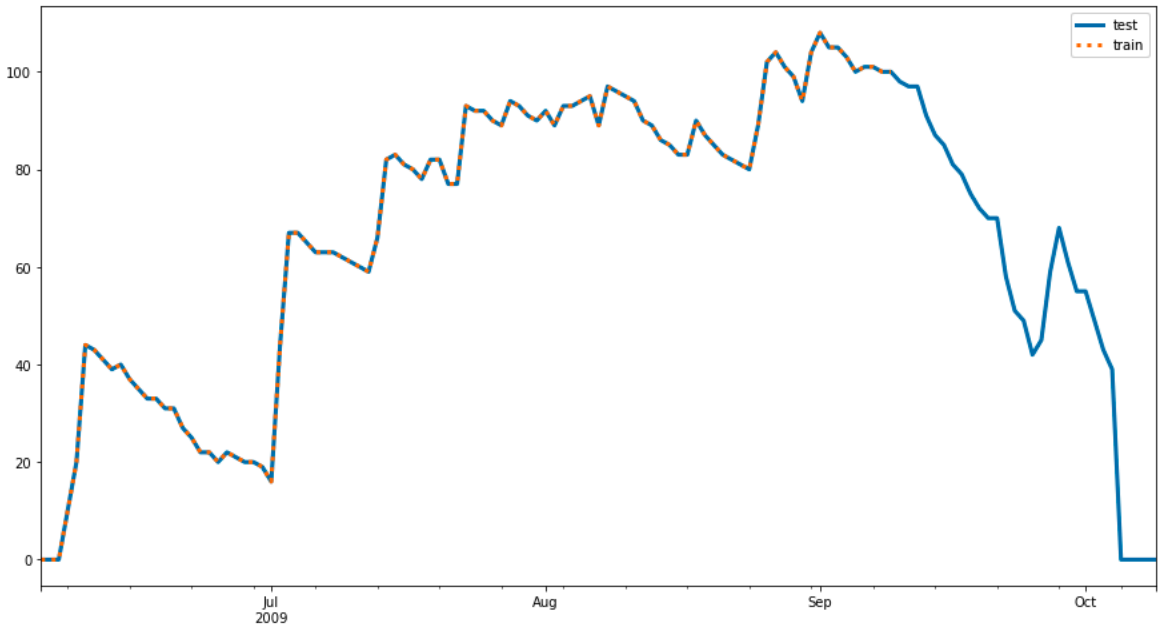
- *The information provided from the data source was imported and treated using R <Capstone Project - DEEPAR TIMESERIES DATA GRAB.Rmd> as it was found that the base importer for R was more robust for treating the data than the default pandas importation process,*
- *Importation and exploration of the dataset shows that the data is spanning approximately 90 columns of data and included variables such as multiple date fields, collated averages and values of natural and man made snow, given we are trying to predict natural snow fall man made snow data was dropped, as well as averages and repeated date fields.*
- *The data remaining was then gathered to provide a date field and a variable field (snow depth).*
- *Investigation into missing numbers has identified several missing numbers. A simple method would be to use the mean value for imputation however this was deemed not suitable for the purposes of the exercise and hot deck imputation was explored and ffill (or forward fill) was used which uses the last observed non-null value, conversely bfill propogates the first observed data point backward.*
- *The dataset was reviewed for outliers.*
- *We have a complete time series set ready for modelling, which is converted into JSON training and test sets of time series and uploaded to S3.*
- *Instantiating and training a DeepAR estimator*
- *Deploying a model and creating a predictor*
- *Evaluating the predictor, refine as necessary.*

## Implementation

As mentioned previously a key advantage of DEEPAR is that it can detect seasonality amongst the data set. The training input required for the algorithm is time series across multiple seasons. It learns this process/seasonality to attempt to predict how the time series evolves season by season.

The key hyper parameters are the prediction length and context length that define how far forward to try and predict and how far into the past the network can see. Typically, these are set to be the same.

In training an amount of datapoints (as shown below) is kept for evaluating the output of the model (model metrics) of which predictions can be compared against. Typically, the prediction length is kept as the training set for each time series (year, season).




The baseline model was compared as per the defined metrics against the validation set and subsequent passes of the DEEPAR algorithm where compared against this for results. The primary metric was deemed suitable throughout the implementation window.

## Refinement

| Model | Benchmark | Pass 0<br><br>All data | 2019 passed as a validation set<br><br>Pass 1 Epoch = 50 | Pass 2 Epoch = 100 |
|---|---|---|---|---|
| DEEPAR Test Root Mean Squared Error | | 70.6 | 50.4 | 38.1 |
| Mean Absolute Error | 15.6 | Not measured | 19.3 | 19.2 |
| Root Mean Squared Error | 19.9 | Not Measured | 21.1 | 21.0 |

The refinement process throughout was to modify the hyperparameters through the model, which is a typical process for model refinement. Due to potential cost implications and the risk or consuming too many resources the tuning was quite limited. What it did observe though was an improvement in the test results by no marked improvement in the model performing on unseen data.

Improvements with this technique would be to include more data, across more years and really open up the hyperparameter tuning. What was noted however that when pulling out the 2019 data for validation purposes there was a significant step change in the model results. It's unclear as to why this is the case, however its hypothesised that because of the shorter nature of the seasons and the fact that the end of one year is the start of the next year, the time series range plays a larger influence in the outcome than the hyperparameters and tuning of the model.

# IV. Results

## Model Evaluation and Validation

## Benchmark Evaluation

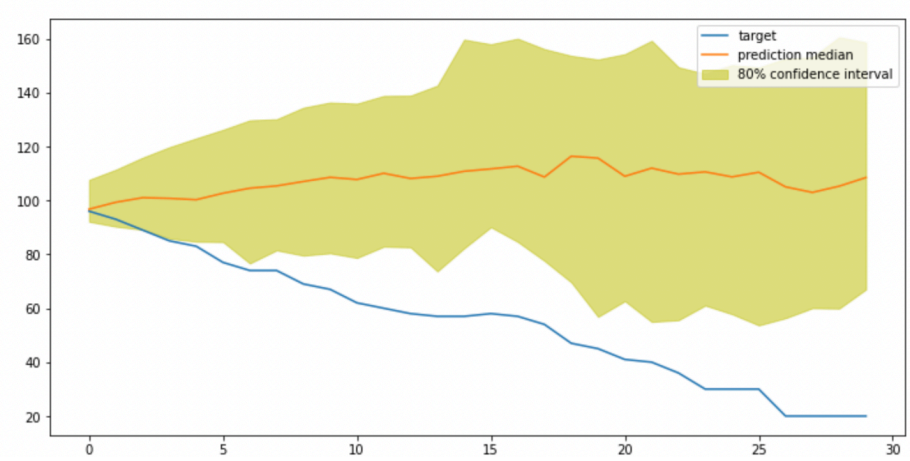
The first step was to compare the baseline model against actual values.  The key metric defined in the proposal was RMSE.

| Model | Benchmark | Pass 0<br><br>All data | 2019 passed as a validation set<br><br>Pass 1 Epoch = 50 | Pass 2 Epoch = 100 |
|---|---|---|---|---|
| DEEPAR Test Root Mean Squared Error | | 70.6 | 50.4 | 38.1 |
| Mean Absolute Error | 15.6 | Not measured | 19.3 | 19.2 |
| Root Mean Squared Error | 19.9 | Not Measured | 21.1 | 21.0 |

## Model Evaluation first pass

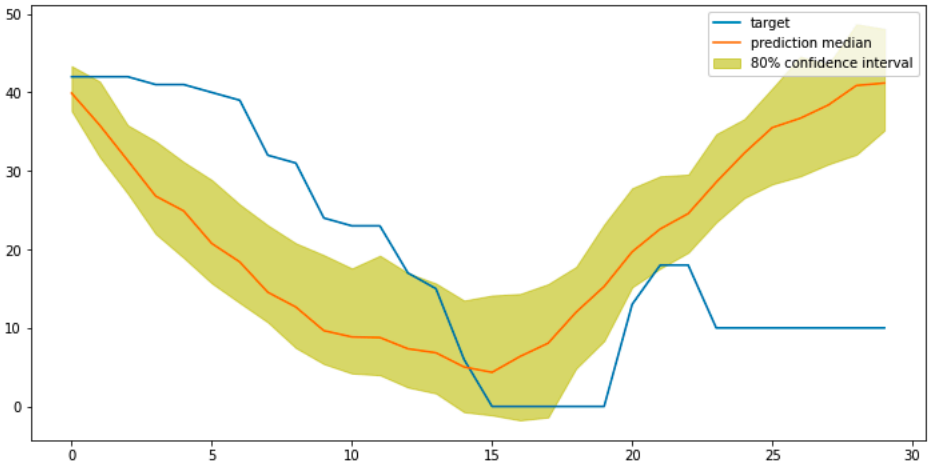
The first part of the process was to consider how well the model performed against the test data.

```
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, RMSE): 70.58822459914727
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, mean_absolute_QuantileLoss): 15701.872324356767
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, mean_wQuantileLoss): 0.6435193575556053
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.1]): 0.76497158996199
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.2]): 0.8614479390754075
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.3]): 0.859004481526672
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.4]): 0.8129119442603627
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.5]): 0.7330360291067695
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.6]): 0.6349572213575488
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.7]): 0.5193940889991697
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.8]): 0.38375143365000103
[08/15/2021 09:37:07 INFO 140525115258240] #test_score (algo-1, wQuantileLoss[0.9]): 0.22219949006252598
[08/15/2021 09:37:07 INFO 140525115258240] #quality_metric: host=algo-1, test RMSE <loss>=70.58822459914727
[08/15/2021 09:37:07 INFO 140525115258240] #quality_metric: host=algo-1, test mean_wQuantileLoss <loss>=0.6435193575556053
#metrics {"StartTime": 1629020227.060253, "EndTime": 1629020227.080374, "Dimensions": {"Algorithm": "AWS/DeepAR", "Host": "alg
o-1", "Operation": "training"}, "Metrics": {"setuptime": {"sum": 7.594823837280273, "count": 1, "min": 7.594823837280273, "ma
x": 7.594823837280273}, "totaltime": {"sum": 120805.499792099, "count": 1, "min": 120805.499792099, "max": 120805.499792099}}}

Training seconds: 201
Billable seconds: 201
CPU times: user 842 ms, sys: 43.3 ms, total: 886 ms
Wall time: 5min 47s
```



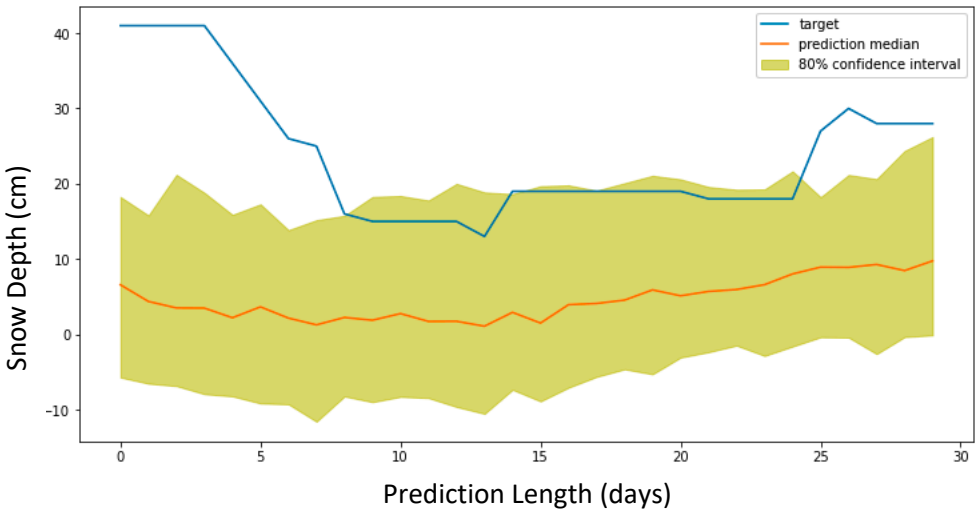

The second modification was to remove the 2019 dataset from the training data, interestingly the baseline model performance improved

```
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, RMSE): 50.400665052759145
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, mean_absolute_QuantileLoss): 9637.451766204833
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, mean_wQuantileLoss): 0.43588655658999703
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.1]): 0.5990551815934443
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.2]): 0.6116000446373928
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.3]): 0.5839306817945448
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.4]): 0.5362768453712756
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.5]): 0.4769037174671713
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.6]): 0.40716458607453226
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.7]): 0.32912264243772493
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.8]): 0.240222034704518
[08/16/2021 08:37:20 INFO 139925975954816] #test_score (algo-1, wQuantileLoss[0.9]): 0.13870327522936904
[08/16/2021 08:37:20 INFO 139925975954816] #quality_metric: host=algo-1, test RMSE <loss>=50.400665052759145
[08/16/2021 08:37:20 INFO 139925975954816] #quality_metric: host=algo-1, test mean_wQuantileLoss <loss>=0.43588655658999703
#metrics {"StartTime": 1629103040.2573452, "EndTime": 1629103040.276615, "Dimensions": {"Algorithm": "AWS/DeepAR", "Host": "al
go-1", "Operation": "training"}, "Metrics": {"setuptime": {"sum": 8.323907852172852, "count": 1, "min": 8.323907852172852, "ma
x": 8.323907852172852}, "totaltime": {"sum": 122385.22171974182, "count": 1, "min": 122385.22171974182, "max": 122385.22171974
182}}}


2021-08-16 08:37:30 Uploading - Uploading generated training model
2021-08-16 08:37:30 Completed - Training job completed
Training seconds: 193
Billable seconds: 193
CPU times: user 944 ms, sys: 39.3 ms, total: 984 ms
Wall time: 6min 14s
```




The key performance criteria though, was how well did it perform against the original data. It needs to be noted that because of the 30 day prediction length, the comparison would only be for thirty days.

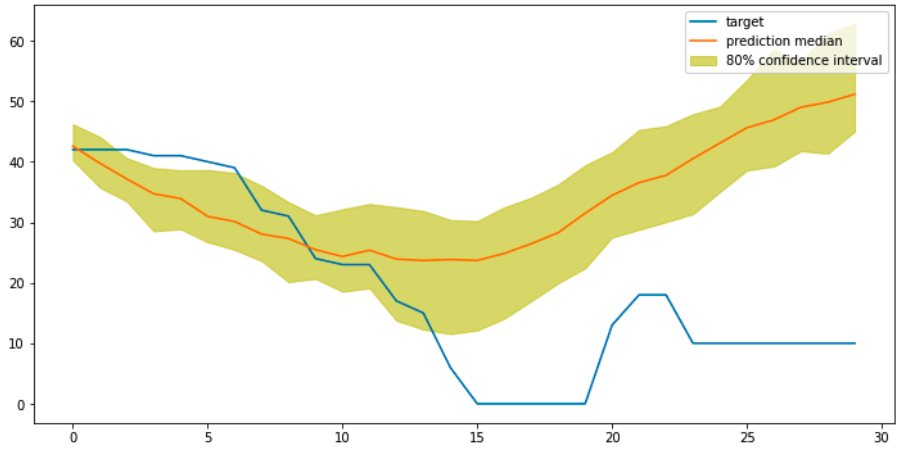
The below image shows 2019 actual vs the model prediction.




```
Mean Absolute Error (MAE): 19.290462199999997
Root Mean Squared Error (RMSE): 21.067965108643744
```

Hyperparameter tuning.  The epochs were changed from 50 to 100

```
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, RMSE): 38.084316322333834
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, mean_absolute_QuantileLoss): 6769.213308758206
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, mean_wQuantileLoss): 0.3061607104820536
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.1]): 0.3993406475727938
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.2]): 0.41308215641209794
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.3]): 0.4048847674676488
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.4]): 0.3786240482891054
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.5]): 0.3430389433627428
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.6]): 0.29473093009545004
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.7]): 0.23922627823862866
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.8]): 0.1768231996547469
[08/16/2021 09:39:54 INFO 140719824602496] #test_score (algo-1, wQuantileLoss[0.9]): 0.10569542324526822
[08/16/2021 09:39:54 INFO 140719824602496] #quality_metric: host=algo-1, test RMSE <loss>=38.084316322333834
[08/16/2021 09:39:54 INFO 140719824602496] #quality_metric: host=algo-1, test mean_wQuantileLoss <loss>=0.3061607104820536
#metrics {"StartTime": 1629106794.4908528, "EndTime": 1629106794.5140557, "Dimensions": {"Algorithm": "AWS/DeepAR", "Host": "a
lgo-1", "Operation": "training"}, "Metrics": {"setuptime": {"sum": 7.854700088500977, "count": 1, "min": 7.854700088500977, "m
ax": 7.854700088500977}, "totaltime": {"sum": 111374.49717521667, "count": 1, "min": 111374.49717521667, "max": 111374.4971752
1667}}}


2021-08-16 09:40:06 Uploading - Uploading generated training model
2021-08-16 09:40:06 Completed - Training job completed
Training seconds: 195
Billable seconds: 195
CPU times: user 822 ms, sys: 72.8 ms, total: 895 ms
Wall time: 5min 45s
```




```
Mean Absolute Error (MAE): 19.2262768
Root Mean Squared Error (RMSE): 21.0212674469745
```

## Justification

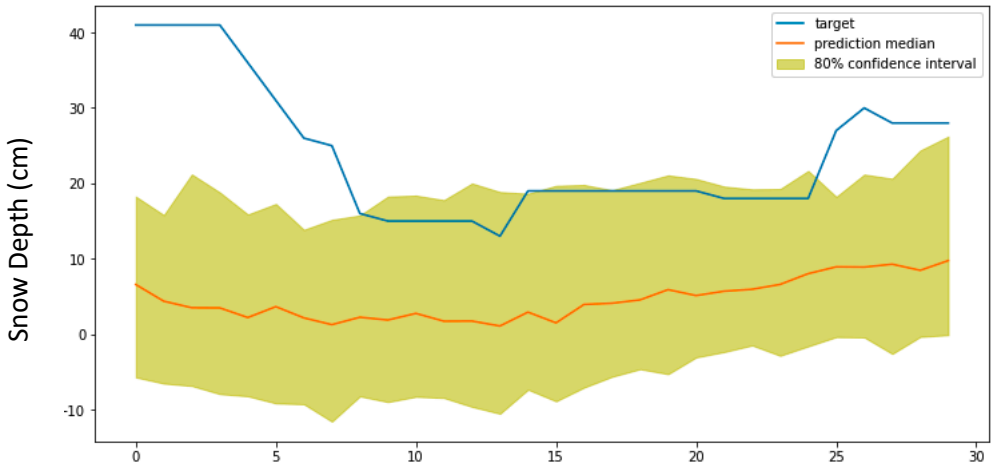
| Model | Benchmark | Pass 0<br><br>All data | 2019 passed as a validation set<br><br>Pass 1 Epoch = 50 | Pass 2 Epoch = 100 |
|---|---|---|---|---|
| DEEPAR Test Root Mean Squared Error | | 70.6 | 50.4 | 38.1 |
| Mean Absolute Error | 15.6 | Not measured | 19.3 | 19.2 |
| Root Mean Squared Error | 19.9 | Not Measured | 21.1 | 21.0 |

As per the results listed above tuning the model saw an improvement in the test result but not a marked improvement in the model performing against unseen data. The benchmark model performed marginally better than the machine learning model and as such we have been unable to solve the problem proposed.

# V. Conclusion

## Free-Form Visualization

As shown in the visualisation the predicted output for the held back data was not all that accurate.




Its proposed that the some of the reasons for this are the short seasonal data provided, i.e. the snow depth only occurs and is only reported in the actual snow season so the ability of the algorithm to look at recent historical (context length) for the start of the season will see it look at data almost 12 months prior. Similarly, the charts indicate its ability to predict the end of the season is quite poor. A method to manage this would be to extend the data at the end of the season which shows as the season draws to a conclusion the snow fall declines. This was purposefully not done, in the hope that the algorithm would be able to recognise this pattern and because the data wasn't present beyond this time frame, i.e. it doesn't exist.

## Reflection

The process can be summed up as per the below:

1. Define a problem within the public realm in this case can we predict snow depth at an alpine snow resort,
2. Research relevant datasets, and download,
3. Pre-process the data
4. Create a benchmark model
5. Convert the data suitable for algorithmic processing
6. Tune the model and compare against the benchmark.

I found moving to and from json with the dataset quite difficult as it wasn't a format that I was used to using. Additionally, some of the learning material was superseded and the changing in syntax is quite difficult to stay on top of. Items like XGBOOST appear to be better understood and easier to research in comparison.

I also found that splitting out the time series challenged both my data processing skills but more importantly I gained a much better understanding of what the algorithm would be looking for, which more typically are complete time series, i.e. year to year.

## Improvement

An improvement would be to provide time series data for independent variables such as temperature and rain fall with the hypothesis that these provide would be good indicators of snow depth. The advantage of this is that these are monitored all year round and DEEPAR would be better suited to predict and have better historical reference for context length.

The outcome would then be two more accurate independent variables that could be then combined to provide input into a simpler regression model such as XGBOOST to predict the snowfall based on these. Using time shift data the variable could include the previous historical data as well, but this feature engineering would have to be treated outside the scope of this project.

Additionally, getting more data would provide a better insight into the seasonality. A more accurate method might be to infer snow depth based on temperature for which there is year round data or to consider multivariate data which is currently outside the scope of DEEPAR.

---