**Basic Forecasting Techniques**

**An overview of some very simple forecasting models**



Photo by [Mark König](https://unsplash.com/@markkoenig?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

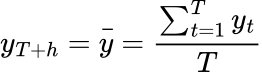
**Introduction**

Forecasting is a wide domain with numerous applications in almost every industry. Due to this, the range of forecasting models is also very large with each model having its own pros and cons.

In this article, I want to go over some basic and simple forecasting models. Despite their simplicity, these models can offer good results in practice and provide a good basis to iterate from.

**Average Forecast**

The first model we will consider is the *average forecast.* This model simply assumes that all future values are equal to the mean of all the previous observations:



Equation generated by author in LaTeX

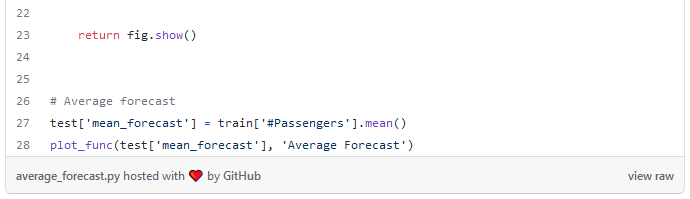
Where ***h*** is the future time-step we are forecasting for, ***T*** is the length of the time series, ***y\_t*** is an observed value at time ***t*** and ***y\_bar*** is the mean of the observed values. For this model we must have some past data available to compute the forecast.

We can implement this in Python using the US air passenger dataset:

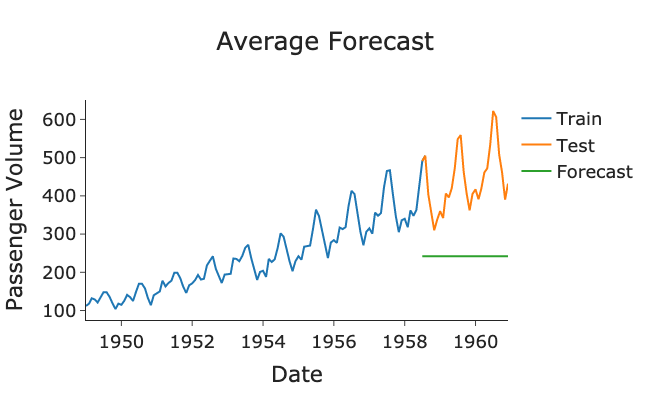
Data [sourced from Kaggle](https://www.kaggle.com/datasets/ashfakyeafi/air-passenger-data-for-time-series-analysis) with a CC0 licence.

<https://gist.github.com/egorhowell/3e04f84ad40e0a5c5ce000dc804a2e47#file-average_forecast-py>





GitHub Gist by author.



Plot generated by author in Python.

From the above plot, the forecast is clearly not very good as it hasn’t captured the trend or seasonality in the data and is clearly significantly under-forecasting.

**Naive Forecasting**

The second model, *naive forecasting,* is setting the future forecast equal to the latest observed value:

https://miro.medium.com/max/130/1*hdUzr4rNtWQ7NNOG4gBgeg.png

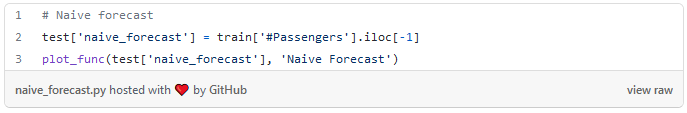
Equation generated by author in LaTeX

This model is considered the benchmark for any forecast and is often used to model stock market and financial data due to its erratic nature. The naive model can also be called *random-walk-without-drift model.*

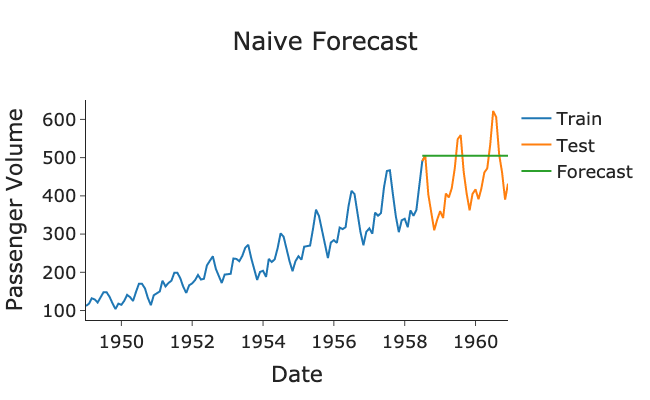
It is also the basis behind the error metric [**mean absolute scaled error (MASE)**](https://en.wikipedia.org/wiki/Mean_absolute_scaled_error)**.** This is the accuracy metric of choice by many professional forecasters as it is: scale independent, interpretable and symmetrical. You can read more MASE in my previous post here:

An example of plotting and carrying out the naive forecast method is show below in Python:

<https://gist.github.com/egorhowell/85a3a160638a3adc5a36ad18d6711cf0#file-naive_forecast-py>



GitHub Gist by author.



Plot generated by author in Python.

This is better than the average forecast as it has somewhat captured the trend and is not under-forecasting as much. However, it is still missing the yearly seasonality variability.

**Seasonal Naive Forecasting**

The third method is an extension of the naive method, but this time the forecast is equal to the most recent observed value in the same season. Hence, it is know as the *seasonal naive* model. For example, the forecast for the next quarter one is equal to the previous years quarter one value. This model is useful when we have a clear and large seasonal variation in our time series.

Mathematically the model is written as:

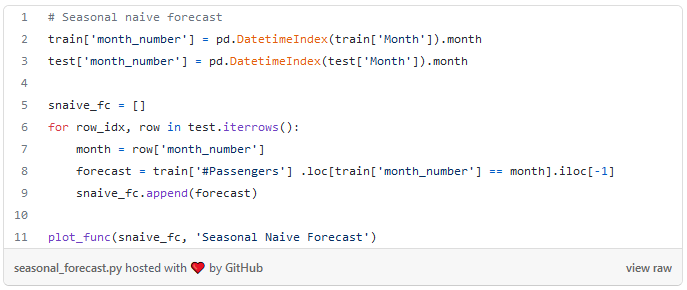
https://miro.medium.com/max/193/1*hQvCUw8F3NM7_688ZXRTNg.png

Equation generated by author in LaTeX

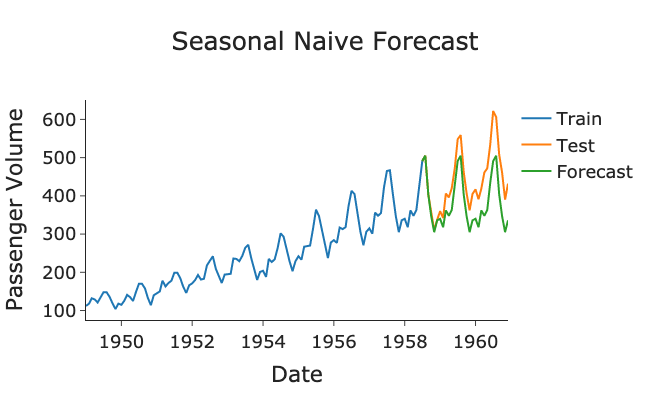
Where ***m*** is the seasonality of the data. So, for monthly data with a yearly seasonality ***m=12,*** quarterly data would have ***m=4*** and weekly data would have ***m=52***.

Below is an implementation of the *seasonal naive* model in Python for our US airline passenger dataset:

<https://gist.github.com/egorhowell/c99d58ac959394bc26cb2e04dcc26cbf#file-seasonal_forecast-py>



GitHub Gist by author.



Plot generated by author in Python.

As our model has quite an obvious and large seasonality component, the seasonal naive model is performing pretty well. However, it hasn’t full captured the trend of data as we expect the passenger volumes to increase overtime.

If you want to learn more about seasonality and trend, checkout my previous posts on them:

**Drift Model**

The final model we will consider is the *drift model.* This is also an extension of the *naive forecast* where we let the prediction either linearly increase or decrease through time as a function of time step, ***h,*** scaled by the average historical trend:

https://miro.medium.com/max/327/1*NfRAQU9DoK7Dby4ipedzxw.png

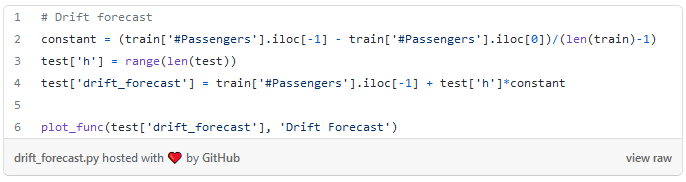
Equation generated by author in LaTeX.

This is basically just drawing a straight line from the first to last point and extending it forwards through time. However, this is where the issue lies as the model will always either increase or decease through time which is often not the case in real life scenarios.

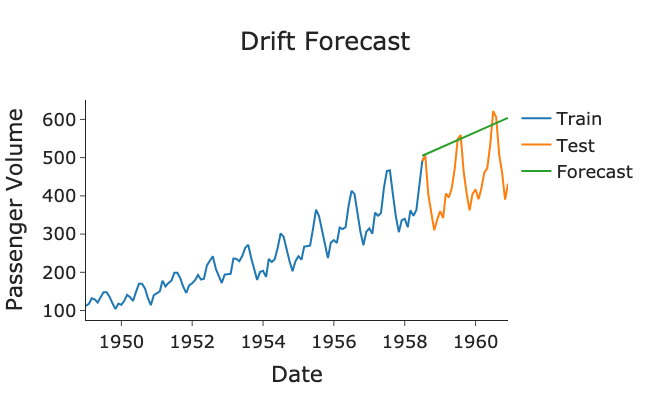
This model is example of [**stochastic drift**](https://en.wikipedia.org/wiki/Stochastic_drift)**.** Other models such as [**Geometric** **Brownian Motion**](https://en.wikipedia.org/wiki/Geometric_Brownian_motion) and [**Random Walk**](https://en.wikipedia.org/wiki/Random_walk) also use stochastic drift.

Below is an implementation of the *drift* model in Python:

<https://gist.github.com/egorhowell/1bf9d523becd8b3b38ae248b7169500a#file-drift_forecast-py>



GitHub Gist by author.



Plot generated by author in Python.

The model captures the general trend quite well due to the passenger volumes mostly increase linearly through time. However, it fails to capture any seasonality in the forecast which is observed in the data.

**Summary and Further Thoughts**

In this post we have gone over four basic forecasting methods. If your model is highly seasonal, the naive seasonal forecast is probably the best choice if it is not, the drift model is the next best option.

In general, these basic models will often not give the optimal results but are good starting points from which to build your more complex models, such as [**ARIMA**](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average) and [**Exponential Smoothing**](https://en.wikipedia.org/wiki/Exponential_smoothing)**.** These basic methods are also good comparison models to measure your more sophisticated model’s performance against.