Building Digital Twins

Mr Ashlin Darius Govindasamy
University of South Africa
Department of Mathematics and Computer Science

August 3, 2022

Abstract

A digital twin is a virtual representation of a physical object that is designed to exactly reflect it. The object under investigation — say, a wind turbine — is equipped with a variety of sensors that monitor various aspects of its operation. These sensors collect data on the energy production, temperature, weather conditions, and other characteristics of the physical object's performance. This information is subsequently sent to a processing machine, where it is applied to a digital copy. Once the virtual model has been given this information, it may be used to run simulations, investigate performance concerns, and suggest improvements, all with the purpose of gaining important insights that can later be applied to the original physical device. In this paper methods of machine learning will be discussed, statistics, forecasting, mathematical equations, real-world examples, and methodology for building digital twins.

Keywords: machine-learning; digital twins; forecasting; statistics; equations; dashboard

Contents

Chapter 1

Introduction

A digital twin is a representation of a physical thing, process, or service in digital form. A digital twin is a computer representation of a real thing, such as a jet engine or wind farm, or even larger objects, such as skyscrapers or entire towns. The digital twin technology can be used to reproduce processes in order to collect data and anticipate how they will perform, in addition to real assets. A digital twin is a computer program that creates simulations based on real-world data to forecast how a product or process will perform. To improve the output, these applications can use the internet of things (Industry 4.0), artificial intelligence, and software analytics. These virtual models have become a standard in modern engineering to drive innovation and enhance performance, thanks to advancements in machine learning and elements such as big data. In brief, having one can help improve strategic technological trends, prevent costly breakdowns in physical objects, and test processes and services utilizing superior analytical, monitoring, and predictive capabilities. The development of a mathematical model that simulates the original begins with professionals in applied mathematics or data science examining the physics and operational data of a physical object or system. The creators of digital twins make certain that the virtual computer model may get feedback from sensors that collect data from the real-world version. This allows the digital version to mimic and emulate what is happening with the original version in real-time, allowing for the collection of data on performance and potential issues. A digital twin can be as complicated or as basic as you need it to be, with the quantity of data used deciding how closely the model mimics the actual version in the real world. The twin can be used in conjunction with a prototype to provide feedback on the product as it is

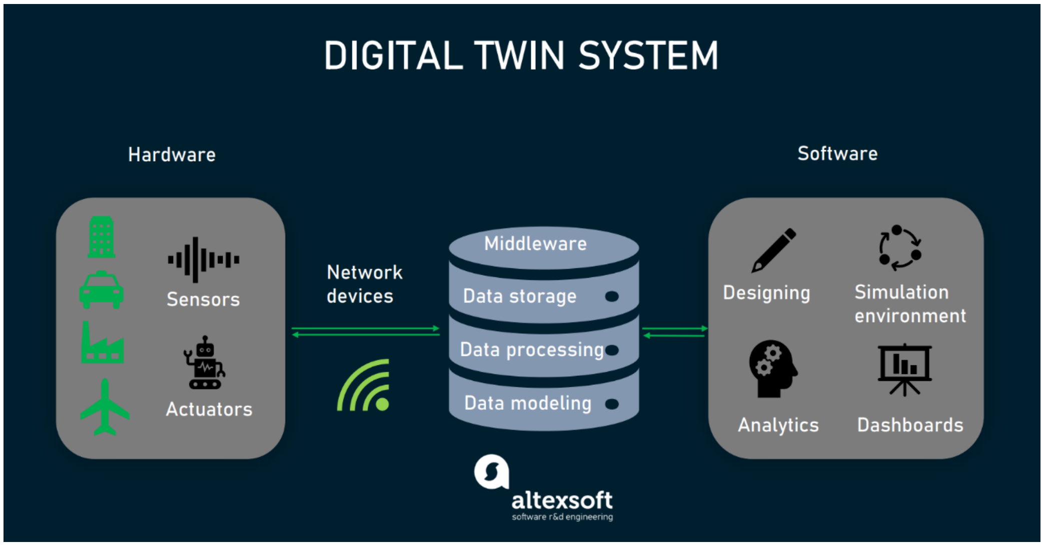


Figure 1.1: Components of a Digital Twin.

being created, or it can be used as a prototype in and of itself to simulate what might happen when the real version is built.

1.0.1 Background Knowlege of Digital Twins

You must first comprehend what you are attempting to construct when creating a Digital Twin. When creating a digital twin of a water plant or a power plant, for example, you'll need to understand how the facility works in real life. To run analytics on that plant, you'll need to gather information about it. Companies will make it difficult for you to obtain data because exposing the info to the public could be dangerous. Let's say you obtained the information by obtaining permission or collecting it yourself or with the help of coworkers. You must be aware of the variables, qualities, and influences on plant changes. You must ask yourself this question. What we want to foretell or what we need to forecast is the question you must now ask yourself. We can start building our digital twin after we know what our goals are and have a good understanding of our data. I've included some questions below that may assist you in asking the right questions as you begin to develop your digital twin.

- Do we understand our dataset?

- Is our data useful to make predictions?

- Does our dataset have outliers? Yes, then the software should flag/alert outliers or remove them. When running predictions?

- What behavior are we trying to predict or forecast?

- Are there any relationships in our dataset's parameters/variables?

- What machine learning methods are we going to use to run analytics/predictions?

1.0.2 Aim of this study and motivation

If you look at organizations that offer or construct Digital Twin software, you'll see that they charge tens of thousands of dollars to build these solutions. A corporation approached me and told me about the plant's digital twin problem statement. I promised them I could build it for hundreds of dollars in a few weeks. There aren't a lot of resources available on digital twins. I'm filling in the blanks with my experience and knowledge from a solid background in Data Science, Mathematics, and Computer Science. This paper is intended to inspire you to make your businesses smarter and more inventive. It is the fourth Industrial Revolution, and it is time for a shift. The goal of this research is to teach you how to create digital twins. The process for creating digital twins and the knowledge I gained on my own while creating a digital twin of a plant with no prior understanding of digital twins but a background in statistics, data science, and machine learning.

1.0.3 Prerequisites

- You will need to have a good understanding of a good statistical/data science language example Python or R

- Understanding of statistics and mathematics (if creating custom machine learning models or neural networks) for optimizing machine learning models

- Experience in Web Frameworks (used for creating Web Dashboards), Machine Learning Modules (required to forecast and predict data), and a graphing library (MATLAB, Seaborn, etc)

Chapter 2

Methodology

2.1 Data Dashboards in Data Science

A data dashboard is a tool that businesses use to track, analyse, and display data, usually to gain a better understanding of the overall health of the organization, a department, or even a specific process. Dashboards connect a variety of metrics, data sources, APIs, and services behind the scenes, assisting businesses in extracting relevant information from those sources and displaying it in user-friendly ways. Data dashboards, similar to a car's dashboard, organize and display important information at a glance to help you understand your company's most valuable data and find answers to critical questions. By linking dashboards to specific metrics or key performance indicators (KPIs), you gain valuable business intelligence as well as the ability to delve deeply into specific pieces of information to continuously monitor success. Dashboards, like those in cars, show how far you are along your journey and how long it may take to get to your destination. A dashboard's ability to provide up-to-date information and context to help inform business decisions and empower employees is one of its most powerful features. A dashboard, for example, could be used by an IT team to help detect signs of a security breach. Alternatively, a company could incorporate the dashboard into an app or mobile device for first-line workers in the field, ensuring that they have access to the data they require at all times. As for the digital twin case in this paper, our paper will be used visualize data, forecast data for Nth number of years, make predictions on a given date, find out the impacts of parameters in your dataset and find out how to optimize the production

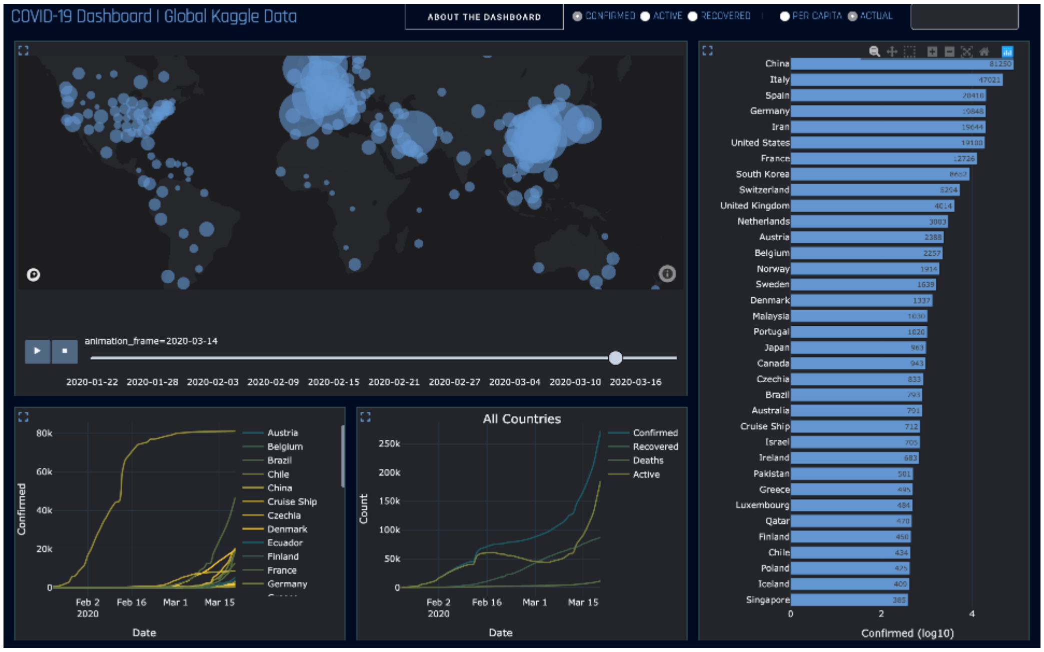


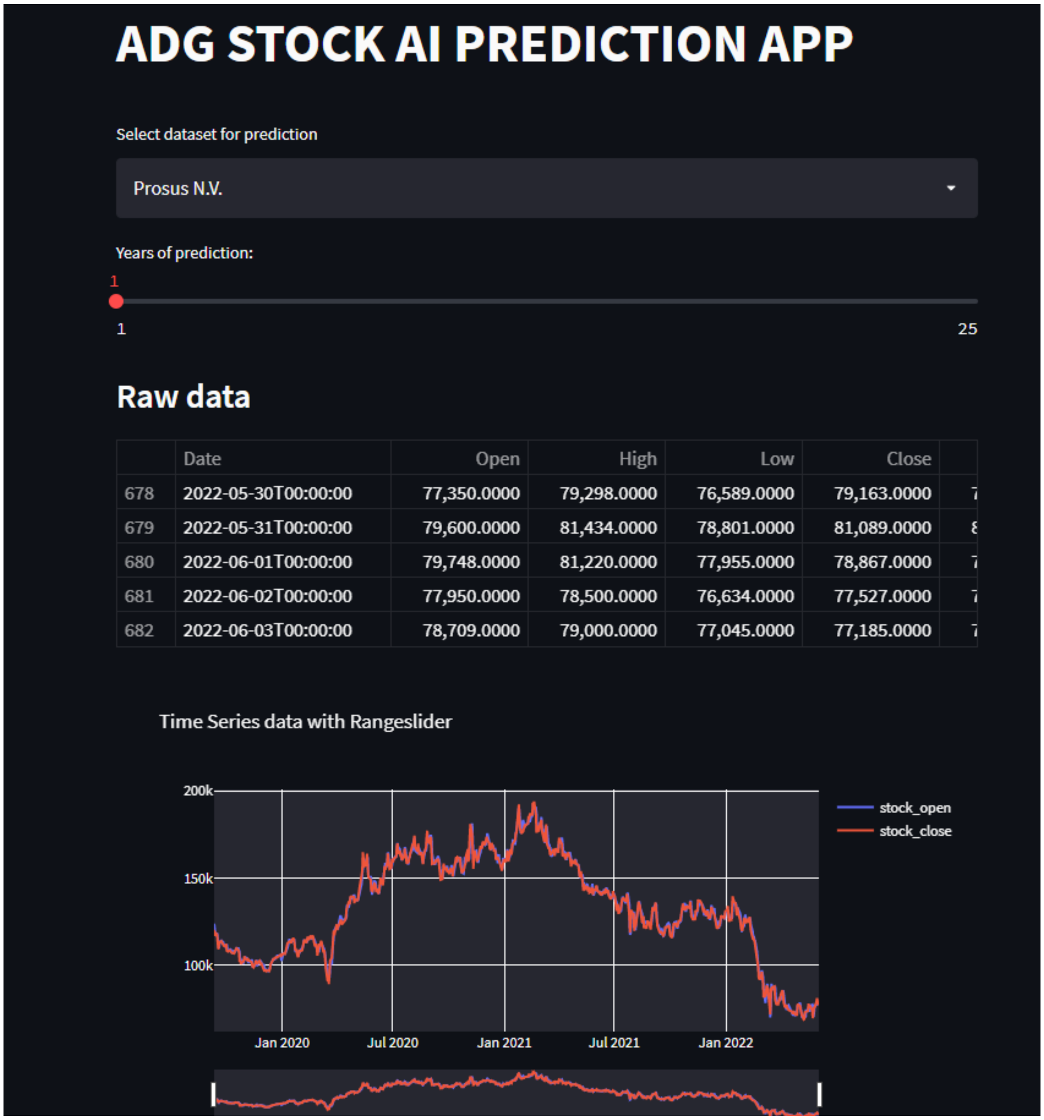
Figure 2.1: Example of a Covid-19 Dashboard just visualizing data with filters.

line or produce the best outcome for your plant/company. Our parameters (dataset headers example .csv) can be described as a set of

$$\{p_n | n = p_1, p_2, p_2, \ldots\}$$

where p_n is a parameter.

Let's say for example in a water plant our p_1 can be DateAndTime , p_2 = PropertyMeasurement , p_3 = SomeChemicalCompound … p_n = our dataset parameters for a given index We could sketch graphs on this data and predict values based on parameter behaviors. We can play around, modify some parameters and see what effect that makes on our plant. Our AI can also optimize the plant for the best-fine-tuned results for our plant. It can generate insights for us and tell us how we can improve the quality of our plant or object. Below are examples of the dashboard and the functionality it presents to us.



ADG STOCK AI PREDICTION APP

Select dataset for prediction

Prosus N.V.

Years of prediction:

1

1 25

Raw data

	Date	Open	High	Low	Close	
678	2022-05-30T00:00:00	77,350.0000	79,298.0000	76,589.0000	79,163.0000	7
679	2022-05-31T00:00:00	79,600.0000	81,434.0000	78,801.0000	81,089.0000	8
680	2022-06-01T00:00:00	79,748.0000	81,220.0000	77,955.0000	78,867.0000	7
681	2022-06-02T00:00:00	77,950.0000	78,500.0000	76,634.0000	77,527.0000	7
682	2022-06-03T00:00:00	78,709.0000	79,000.0000	77,045.0000	77,185.0000	7

Time Series data with Rangeslider

Figure 2.2: Example of my application dashboard displaying JSE stocks for a given date interval and forecasting stocks for a given number of years https://stonks.adgstudios.co.za

2.2 Machine Learning and Forecasting

2.2.1 Introduction to Machine Learning and Forecasting for building digital twins

Machine learning and forecasting are important tools in the digital twin toolbox. It is the most important component of the digital twin; without these tools and knowledge, the digital twin would not exist. Artificial Intelligence includes the subcategory of Machine Learning. Machine Learning is used to find trends/equations created from Machine Learning models that best describe our data and will be used to make predictions based on input and output data. Let me describe the above in simple mathematical equations.

Let's take our old-school way of a simple function/formula to determine some sort of output. Let us say

$$\{p_n | n = p_1, p_2, p_2, \dots\} \tag{2.1}$$

where p_n is our parameters.

General Equations we are taught in life

$$f(p_n) = p_n + rules \tag{2.2}$$

$$f(\{p_n | n = p_1, p_2, p_2, \dots\}) = \{p_n | n = p_1, p_2, p_2, \dots\} + rules \tag{2.3}$$

In Layman's Equation Terms
Input + rules = output

(Input is put through the rules returning some form of output)

In normal programming

$$\{p_n | n = p_1, p_2, p_2, \dots\} + rules = ? \tag{2.4}$$

In Layman's Equation Terms
Input + rules = output

Our question mark represents (output, but depends on input and/or rules)

In Machine Learning

Let us define some machine learning equation let's say if we are using the KNN algorithm (more about that later in this paper)

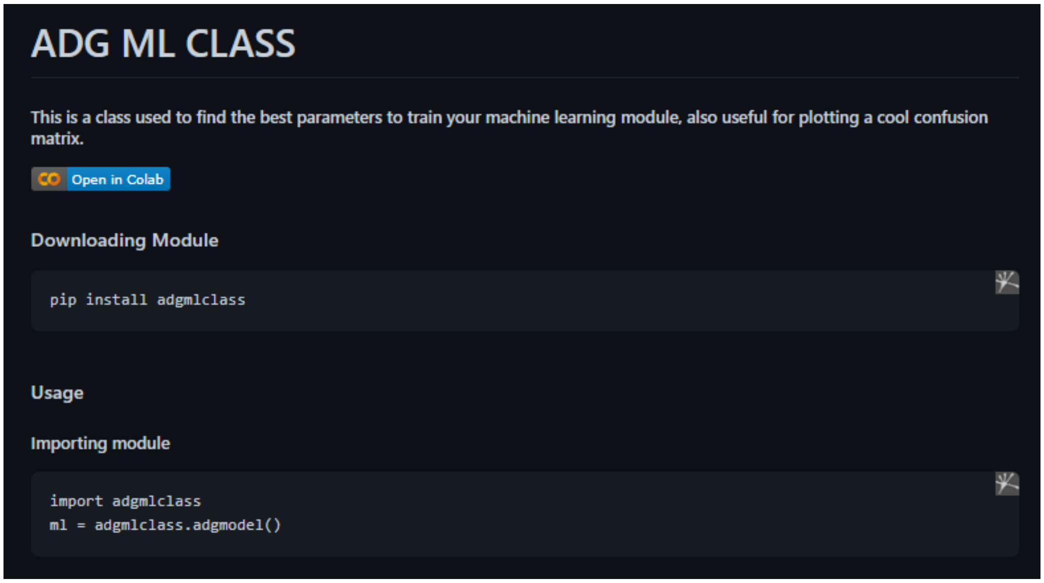
$$f(p_n) = f_{KNN}(p_n) \tag{2.5}$$

$$f(\{p_n|n = p_1, p_2, p_2, \dots\}) = f_{KNN}(\{p_n|n = p_1, p_2, p_2, \dots\}) \tag{2.6}$$

In Layman's Equation Terms
Input + ? = output

In this case the ? means the rules (program makes rules based on input and expected output)

I have built a Python Module to find the best parameters to trains your machine learning module. It is called adgmlclass which is on PyPi and code is open source on my GitHub. You could also set the rules to a generic machine learning algorithm, modify the algorithm with fine-tuned Machine Learning parameters.



ADG ML CLASS

This is a class used to find the best parameters to train your machine learning module, also useful for plotting a cool confusion matrix.

Downloading Module

```
pip install adgmlclass
```

Usage

Importing module

```
import adgmlclass
ml = adgmlclass.adgmodel()
```

Figure 2.3: ADGMLCLASS : https://github.com/adgsenpai/adgmlclass

2.2.2 What is Machine Learning?

Machine learning, also known as automatic learning, is a branch of science and a subcategory of artificial intelligence. It entails allowing algorithms to find "patterns" in data sets, i.e., recurring patterns. This information can take the form of numbers, words, images, statistics, and so on. Machine Learning can use anything that can be stored digitally as data. Algorithms learn and improve their performance in performing a specific task by detecting patterns in this data.

2.2.3 Which Mathematical Concepts Are Used in Machine Learning and Data Science?

Statistics, Linear Algebra, Probability, and Calculus are the four key concepts that drive machine learning. While statistical concepts are at the heart of all models, calculus aids in the learning and optimization of those models.

2.2.4 Importance of Machine Learning

Machine learning is important because it allows businesses to see trends/predictions in customer behavior and business operational patterns while also assisting in the development of new products. Machine learning is at the heart of many of today's most successful businesses, including Facebook, Google, and Uber. In this paper, the importance of Machine Learning is that we are using it to predict and forecast data based on our input data which is the parameters to determine an output value on an output parameter.

2.2.5 Machine Learning Types, Algorithms, and Usage

2.2.5.1 Types of Machine Learning:

Supervised Learning
Supervised learning is a type of machine learning in which machine learning from known datasets (set of training examples), and then predict the output. A supervised learning agent needs to find out the function that matches a given sample set. Supervised learning further can be classified into two categories of algorithms:

1. Classifications

2. Regression

Unsupervised Learning

Unsupervised learning is associated with learning without supervision or training. In unsupervised learning, the algorithms are trained with data which is neither labeled nor classified. In unsupervised learning, the agent needs to learn from patterns without corresponding output values. Unsupervised learning can be classified into two categories of algorithms:

1. Clustering
2. Association

Reinforcement Learning

Reinforcement learning is a type of learning in which an AI agent is trained by giving some commands, and on each action, an agent gets a reward as feedback. Using these feedbacks, agent improves its performance. Reward feedback can be positive or negative which means on each good action, agent receives a positive reward while for wrong action, it gets a negative reward. Reinforcement learning is of two types:

1. Positive Reinforcement learning
2. Negative Reinforcement learning

2.2.5.2 List of Common Machine Learning Algorithms

- **Linear Regression – (will explain the algorithm in paper)**
- Logistic Regression
- Decision Tree
- SVM
- Naive Bayes
- **KNN – (will explain the algorithm in paper)**
- K-Means
- Random Forest

- Dimensionality Reduction Algorithms
- Gradient Boosting algorithms
- Autoencoder
- Convolutional Neural Network
- Recurrent Neural Network
- Support Vector Machine
- Deep Learning

Deep learning is another complicated topic that I won't go into much detail about in this paper; instead, I'll give you some general information about it. Deep learning is a subset of a larger family of machine learning techniques based on representation learning and artificial neural networks. There are three types of learning: supervised, semi-supervised, and unsupervised. For image classification, object detection, image restoration, and image segmentation, deep learning has delivered superhuman accuracy—even handwritten digits can be recognized. Deep learning, which makes use of massive neural networks, is teaching machines to automate the tasks that human visual systems perform. Deep learning is a machine learning and artificial intelligence (AI) technique that mimics how humans acquire knowledge.

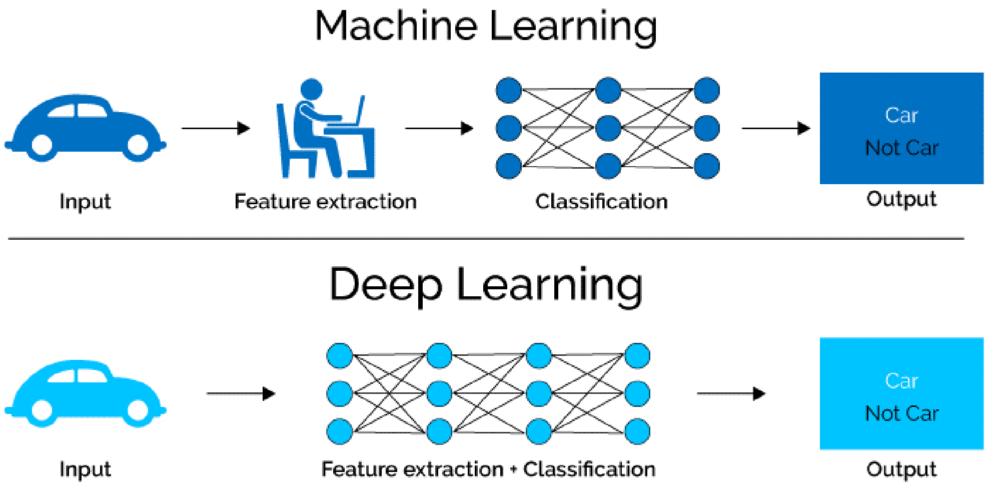


Figure 2.4: Deep Learning General Structure Diagram

2.2.6 Linear Regression

2.2.6.1 Introduction to Linear Regression

A linear approach to modeling the relationship between a scalar response and one or more explanatory variables is known as linear regression in statistics. Simple linear regression is used when there is only one explanatory variable; multiple linear regression is used when there is more than one. Linear regression is a supervised machine-learning regression algorithm.
A typical linear regression equation that you learned in high school is in the form of

$$y = b_0 + b_1 x \tag{2.7}$$

where y is the response variable, b_0 is the intercept, and b_1 is the coefficient.

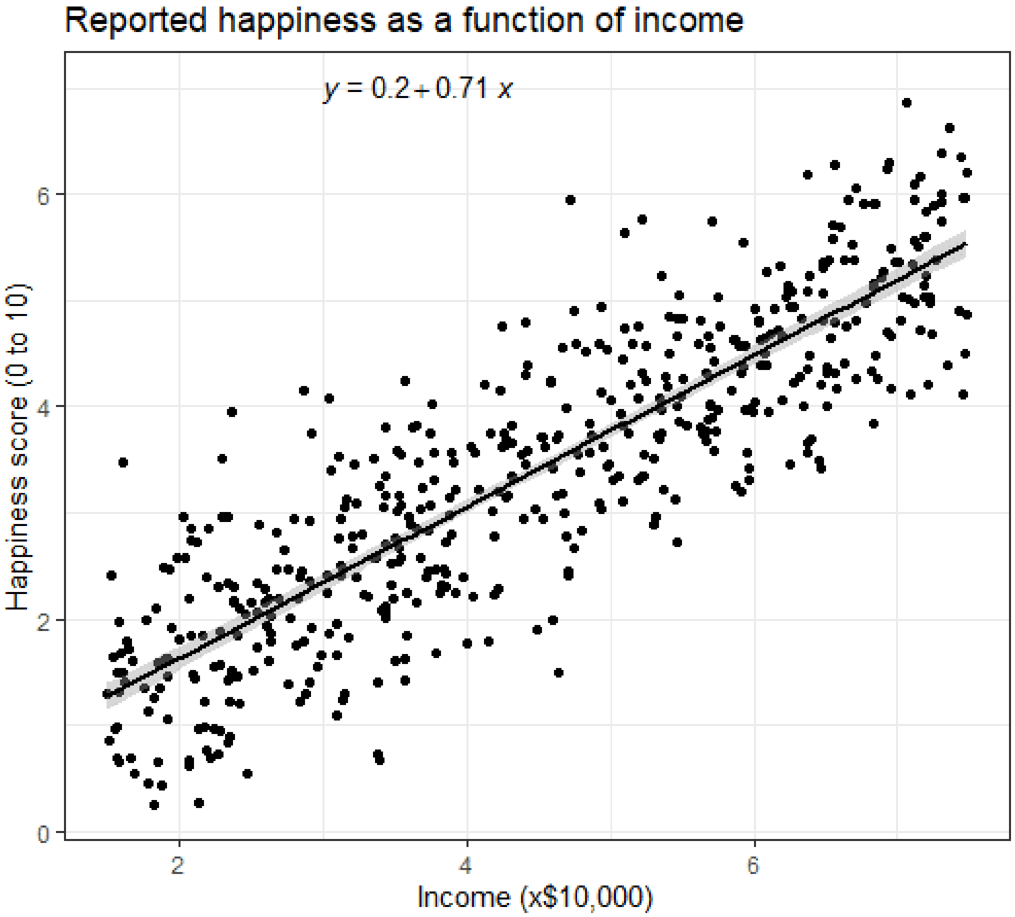


Figure 2.5: Linear Regression function of Reported Happiness by income values

In this example, employees are interviewed and are asked to fill in the form below. Rate your happiness in life from

$$y \in \{0..10\} \tag{2.8}$$

We capture our data and represent it in a digital format or record it on a page. The data scientist visualizes the data in a scatter graph. To make predictions on this data he calculates the regression equation which is

$$y = 0.2 + 0.71x \tag{2.9}$$

and plots the function on the scatter graph. If you look at the diagram above that is exactly what is being done to predict data.

Simple Prediction and Flaw with Linear Regression
Let us predict my happiness score based on this model.

where x is the income in 10k
and y is the happiness score.

$$y = 0.2 + 0.71(100) \tag{2.10}$$

Our predicted score is **71.2**. That is not possible as the scale can only be $y \in \{0..10\}$ You can see that the model above is invalid and inaccurate. The training size is too small. We could get a better model if we use a larger population. The point I'm trying to make with Linear Regression is that you will get false alerts and reading based on the input data. I would also say emotions with people are not accounted for when working with Machine Learning models. For example, some people who are multi-millionaires are depressed with life. Some people with low income are happy in life they appreciate the simple things in life. Machine Learning does not account for emotion there is another aspect in AI that can help predict emotions and that is called Natural Language Processing. It works with text and speech. Deep learning powers that technology.

Other types of Linear Regression
What I showed you in the introduction was a **simple linear regression** equation which is

$$y = b_0 + b_1x \tag{2.11}$$

where y is the response variable, b_0 is the intercept, and b_1 is the coefficient.

A simple graph of that is

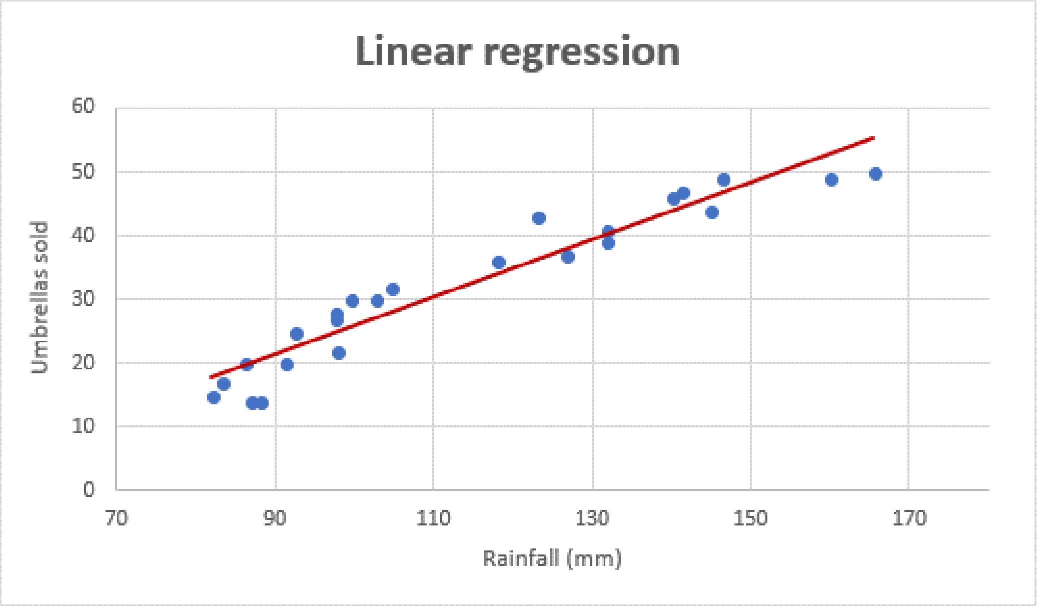


Figure 2.6: Simple Linear Regression

The second type of linear regression is called **Multiple Linear Regression**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2 + \cdots + \beta_n x^n \qquad (2.12)$$

where y is the response variable, β_0 is the intercept, and β_1 is the coefficient for x_1, β_2 is the coefficient for x_2^2, and so on.

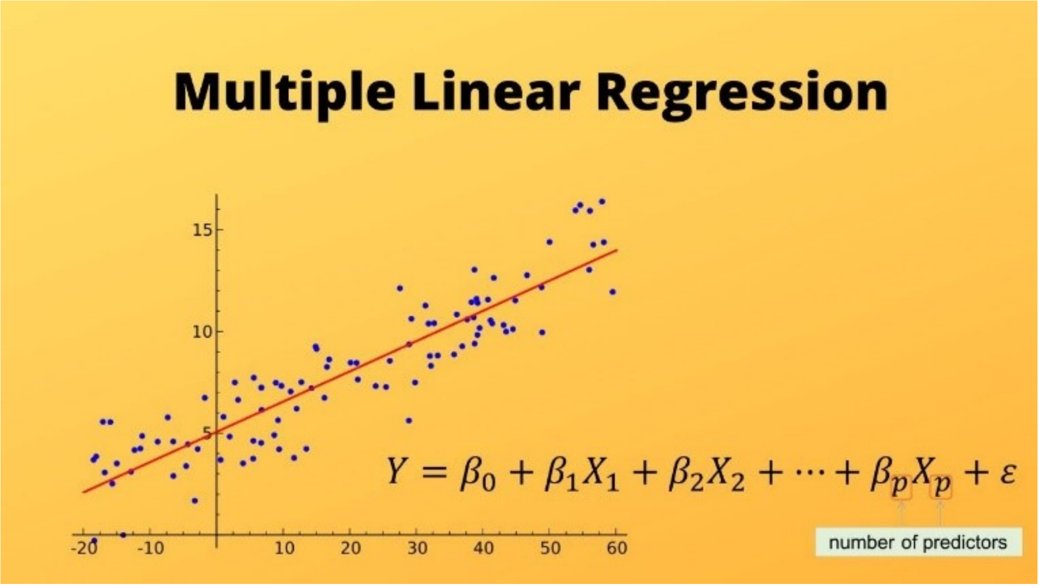


Figure 2.7: Multiple Linear Regression

A graph for multiple linear regression looks similar to the simple linear regression above.

The last type of linear regression is called **Polynomial Linear Regression**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots + \beta_n x_1^n \tag{2.13}$$

where y is the response variable, β_0 is the intercept, and β_1 is the coefficient for x_1, β_2 is the coefficient for x_1^2, and so on.

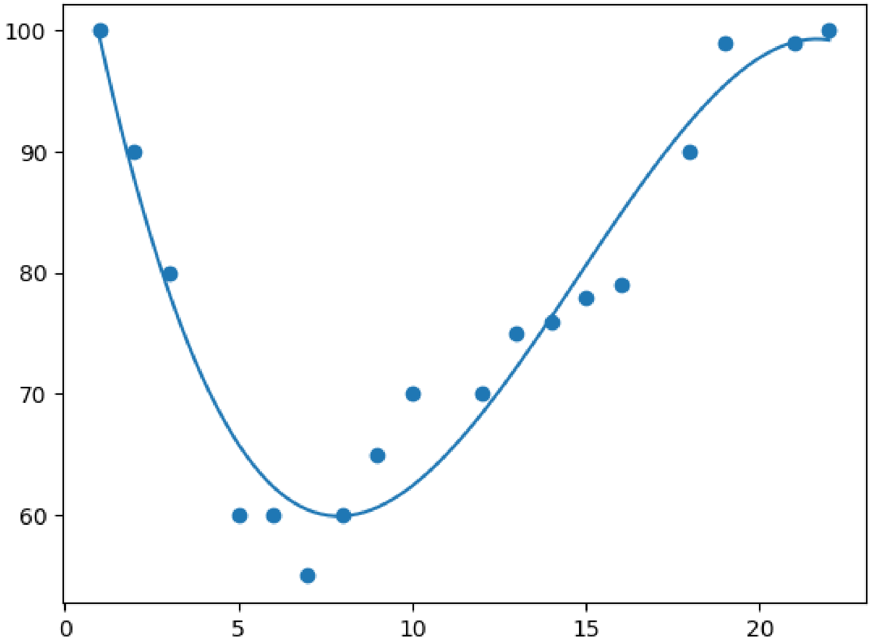


Figure 2.8: Multiple Linear Regression

2.2.7 K-Nearest Neighbors

2.2.7.1 Introduction to K-Nearest Neighbors

K-nearest neighbors (k-NN) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks. Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.
Consider the below diagram:

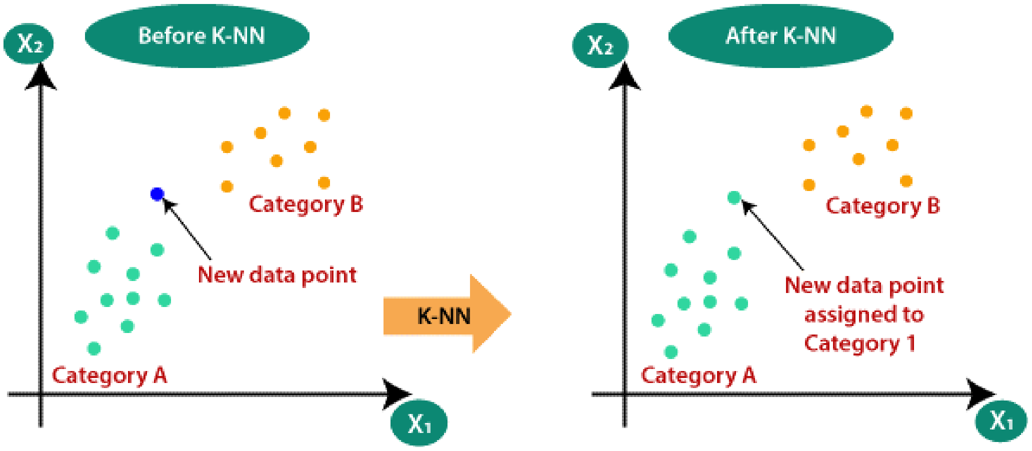


Figure 2.9: K-Nearest Neighbors

The K-NN algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression tasks.

How does the k-nearest neighbors algorithm work?

1. Select the number K of the Neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbour is maximum.

Suppose we have a new data point and we need to put it in the required category.

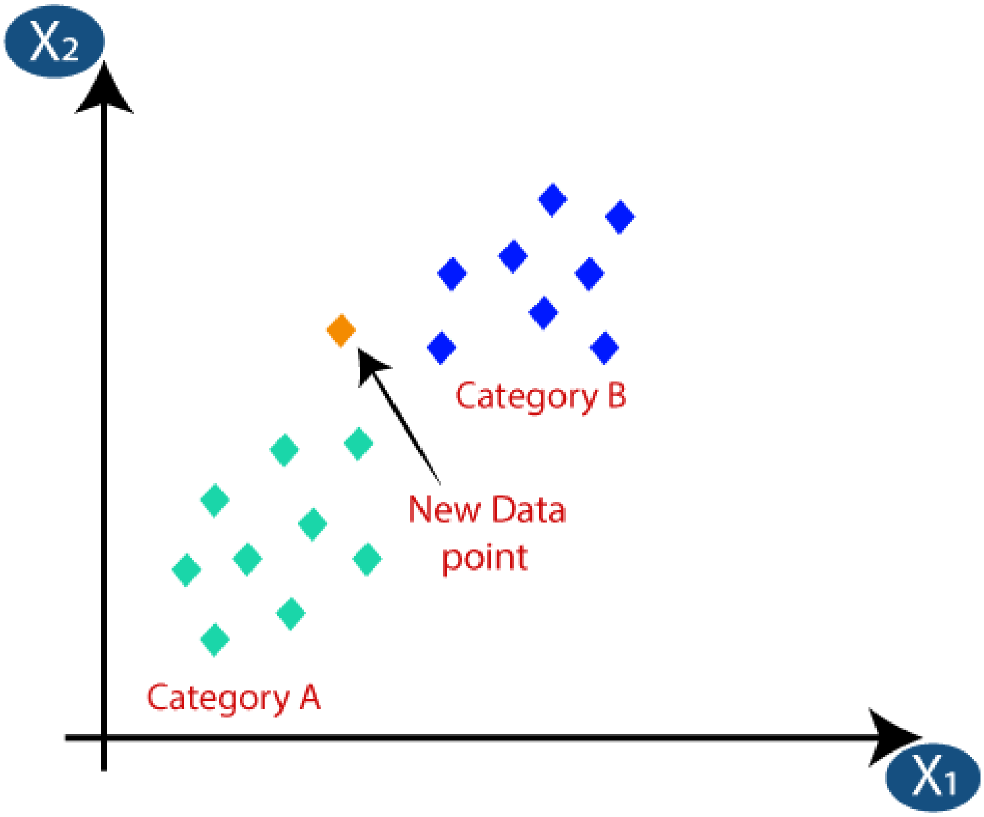


Figure 2.10: K-Nearest Neighbors Datapoint Graph

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the Euclidean distance (which we learnt in high school) between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

$$D_{Euclidean} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.14}$$

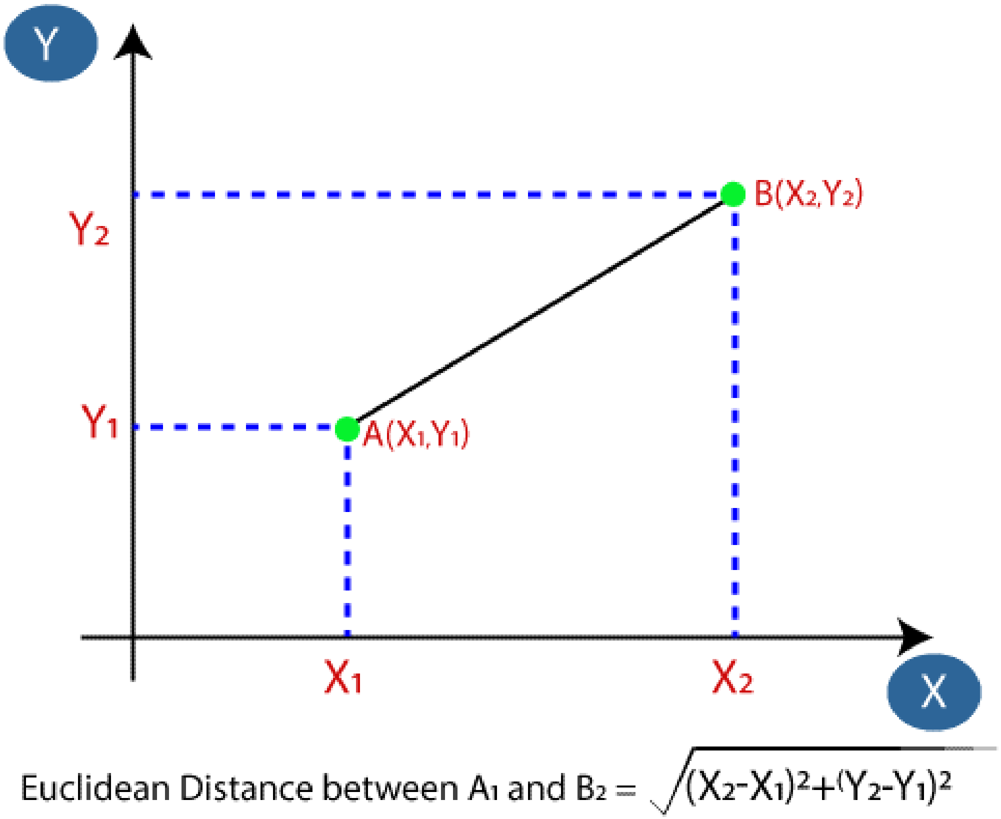

Y
X
Y_2
Y_1
X_1
X_2
$B(X_2, Y_2)$
$A(X_1, Y_1)$
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$

Figure 2.11: Calculating Euclidean distance between two data points

By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

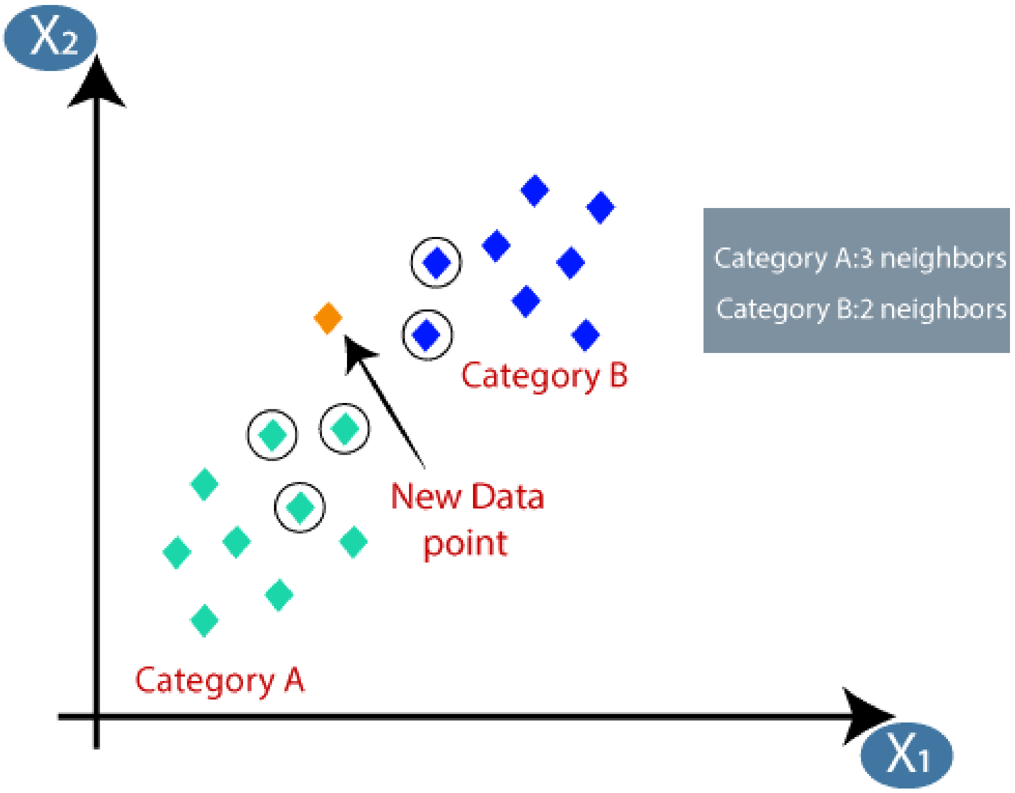


Figure 2.12: Nearest Neighbors

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Advantages of KNN Algorithm

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Distance functions

For KNN we can use different distance functions to achieve better accuracy or different K values.

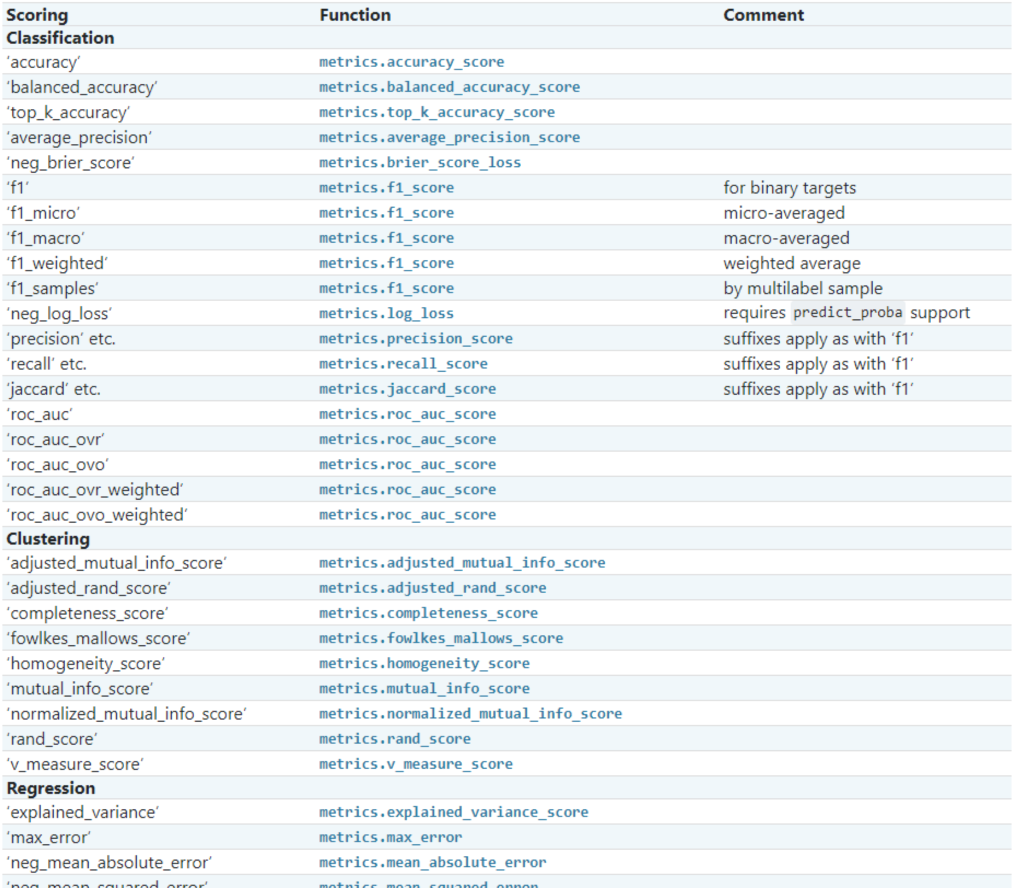
$$D_{Euclidean} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.15}$$

$$D_{Manhattan} = |x_1 - x_2| + |y_1 - y_2| \tag{2.16}$$

$$D_{Minkowski} = \left\{ [((x_1 - x_2)^p)^q]^r \right\}^{1/r} \tag{2.17}$$

2.2.8 Measuring of Performance of Models

We can use functions in our machine learning libraries to measure performance of our model.



Scoring	Function	Comment
Classification		
'accuracy'	metrics.accuracy_score	
'balanced_accuracy'	metrics.balanced_accuracy_score	
'top_k_accuracy'	metrics.top_k_accuracy_score	
'average_precision'	metrics.average_precision_score	
'neg_brier_score'	metrics.brier_score_loss	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'neg_log_loss'	metrics.log_loss	requires `predict_proba` support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'jaccard' etc.	metrics.jaccard_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	
'roc_auc_ovr'	metrics.roc_auc_score	
'roc_auc_ovo'	metrics.roc_auc_score	
'roc_auc_ovr_weighted'	metrics.roc_auc_score	
'roc_auc_ovo_weighted'	metrics.roc_auc_score	
Clustering		
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score	
'adjusted_rand_score'	metrics.adjusted_rand_score	
'completeness_score'	metrics.completeness_score	
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score	
'homogeneity_score'	metrics.homogeneity_score	
'mutual_info_score'	metrics.mutual_info_score	
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score	
'rand_score'	metrics.rand_score	
'v_measure_score'	metrics.v_measure_score	
Regression		
'explained_variance'	metrics.explained_variance_score	
'max_error'	metrics.max_error	
'neg_mean_absolute_error'	metrics.mean_absolute_error	
'neg_mean_squared_error'	metrics.mean_squared_error	

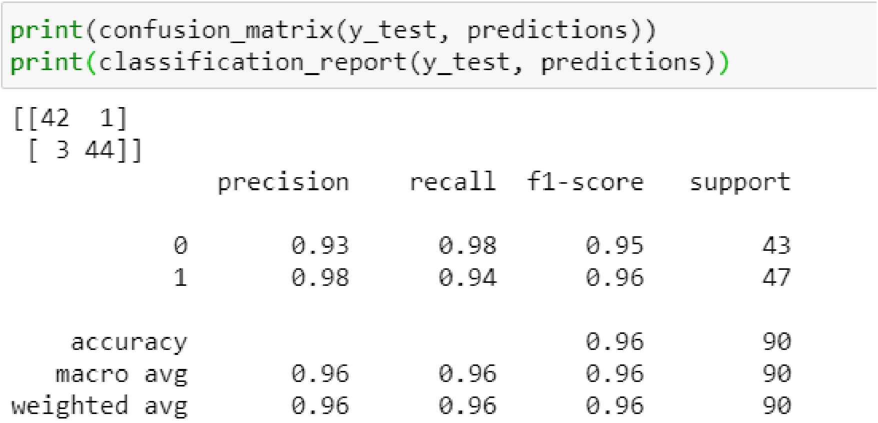
Figure 2.13: Performance measurement functions in Python Scikit-Learn Module

above is a photo of performance metrics functions for the module scikit-learn in Python if you research these function names you can view formulas on these tools.

Check out the following link for more information on performance metrics: `https://scikit-learn.org/stable/modules/model_evaluation.html`

2.2.8.1 Real life example of model evaluation

Below is performance metrics on a model I improved on a project of Galaxy Machine Learning.



```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[42  1]
 [ 3 44]]
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.98      0.94      0.96        47

    accuracy                           0.96        90
   macro avg       0.96      0.96      0.96        90
weighted avg       0.96      0.96      0.96        90
```

Figure 2.14: Performance metrics on a model

For the model above you can see it has extremely good accuracy of around 96% accuracy Anything above 90% I consider good enough there was times I even got 100% accuracy for a model.
Let's check out how our F1-Score is calculated.

$$F1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.18}$$

alternatively, we can use the following formula:

$$F1\text{-}Score = \frac{2TP}{2TP + FP + FN} \tag{2.19}$$

where TP is the number of true positives, FP is the number of false positives, FN is the number of false negatives.

What does TP mean?
That means True Positives in our dataset. Which our correct data in our dataset which tested true in our data.
What does FP mean?
That means False Positives in our dataset. Which our incorrect data in our dataset which tested false in our data.
What does FN mean?

That means False Negatives in our dataset. Which our incorrect data in our dataset which tested true in our data.

Another tool we can use to showcase our TP, FP, FN is called a **confusion matrix**

Mathematically, a confusion matrix looks like this:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{2.20}$$

Graphically could look like this below

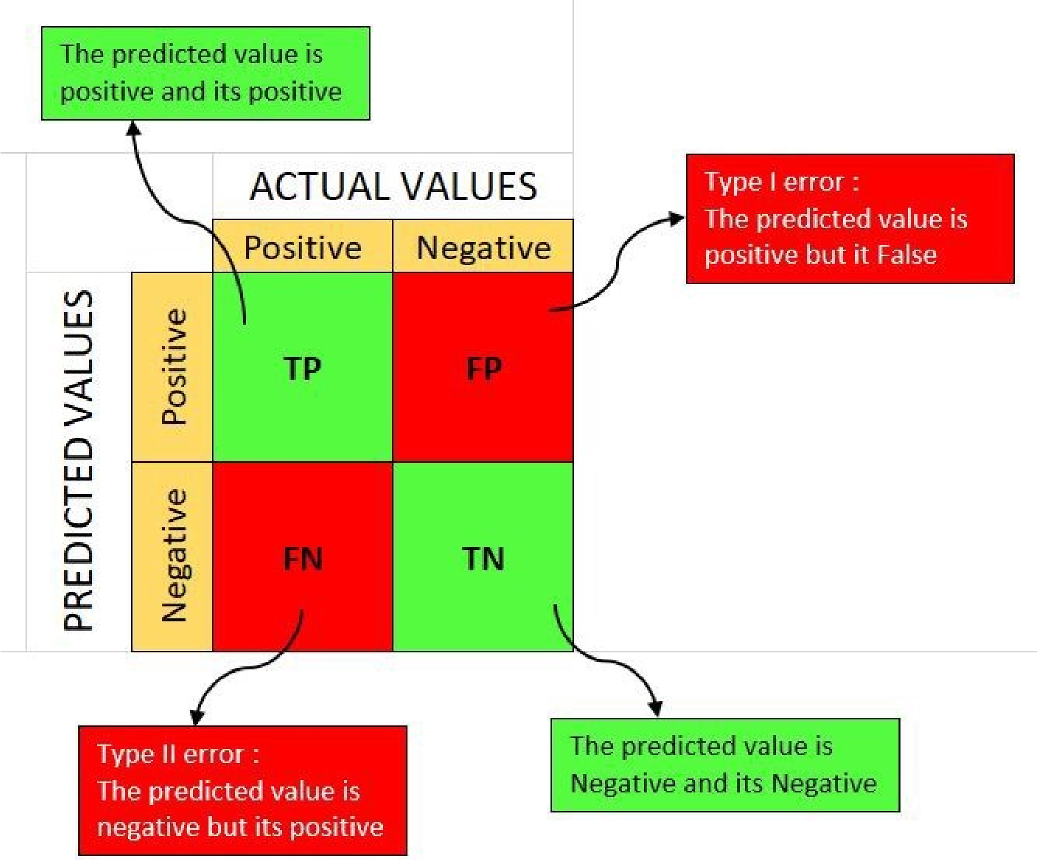
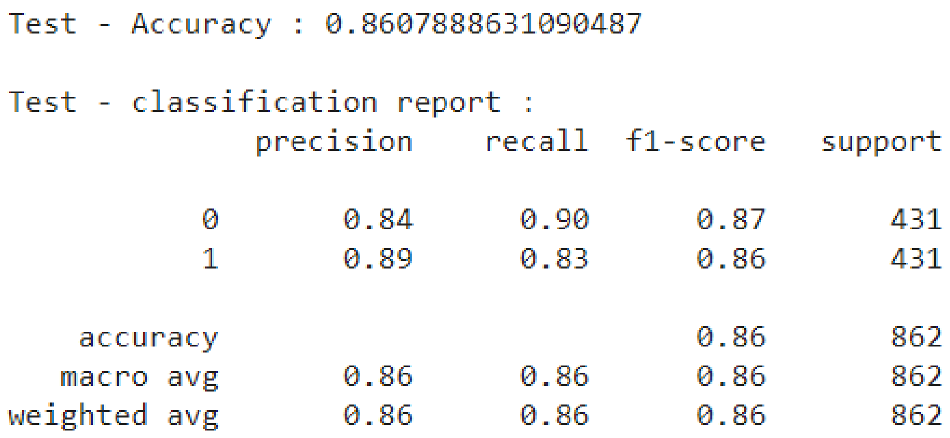

The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
Type I error :
The predicted value is positive but it False
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type II error :
The predicted value is negative but its positive
The predicted value is Negative and its Negative

Figure 2.15: Confusion Matrix

Another graphical representation using the ADGMLCLASS library is shown below:



```
Test - Accuracy : 0.8607888631090487

Test - classification report :
              precision    recall  f1-score   support

           0       0.84      0.90      0.87       431
           1       0.89      0.83      0.86       431

    accuracy                           0.86       862
   macro avg       0.86      0.86      0.86       862
weighted avg       0.86      0.86      0.86       862
```

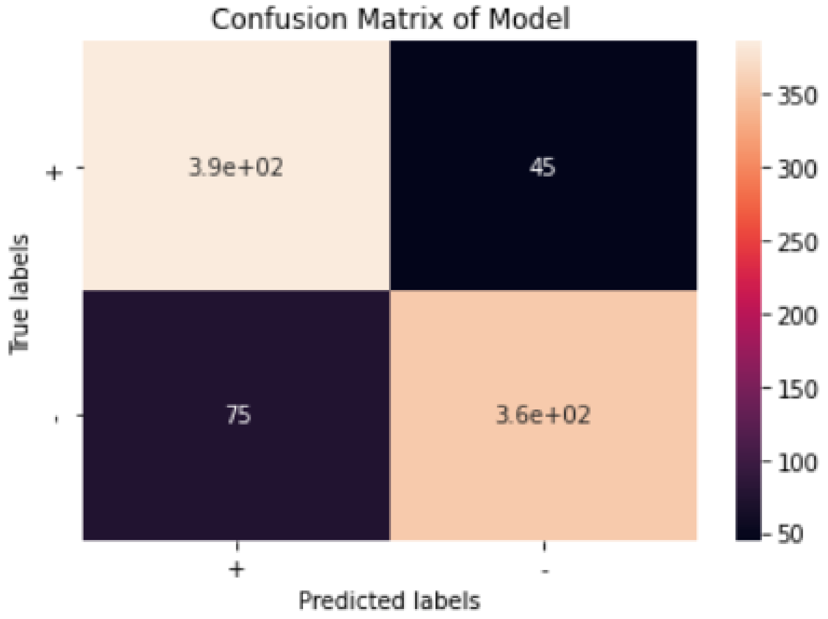
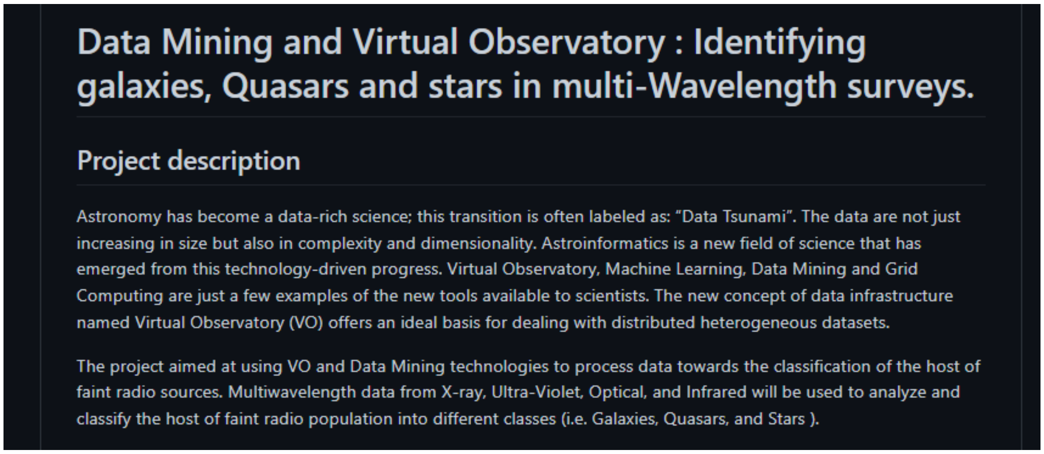


Figure 2.16: Real life Confusion Matrix of a model

2.2.9 Playing with Machine Learning Demos + Real World Examples I did

2.2.9.1 GalaxyML

You can play with GalaxyML. I helped a masters student optimize the ML Solution using adgmlclass and put the repo because this is a real-world example of ML in action.



Data Mining and Virtual Observatory : Identifying galaxies, Quasars and stars in multi-Wavelength surveys.

Project description

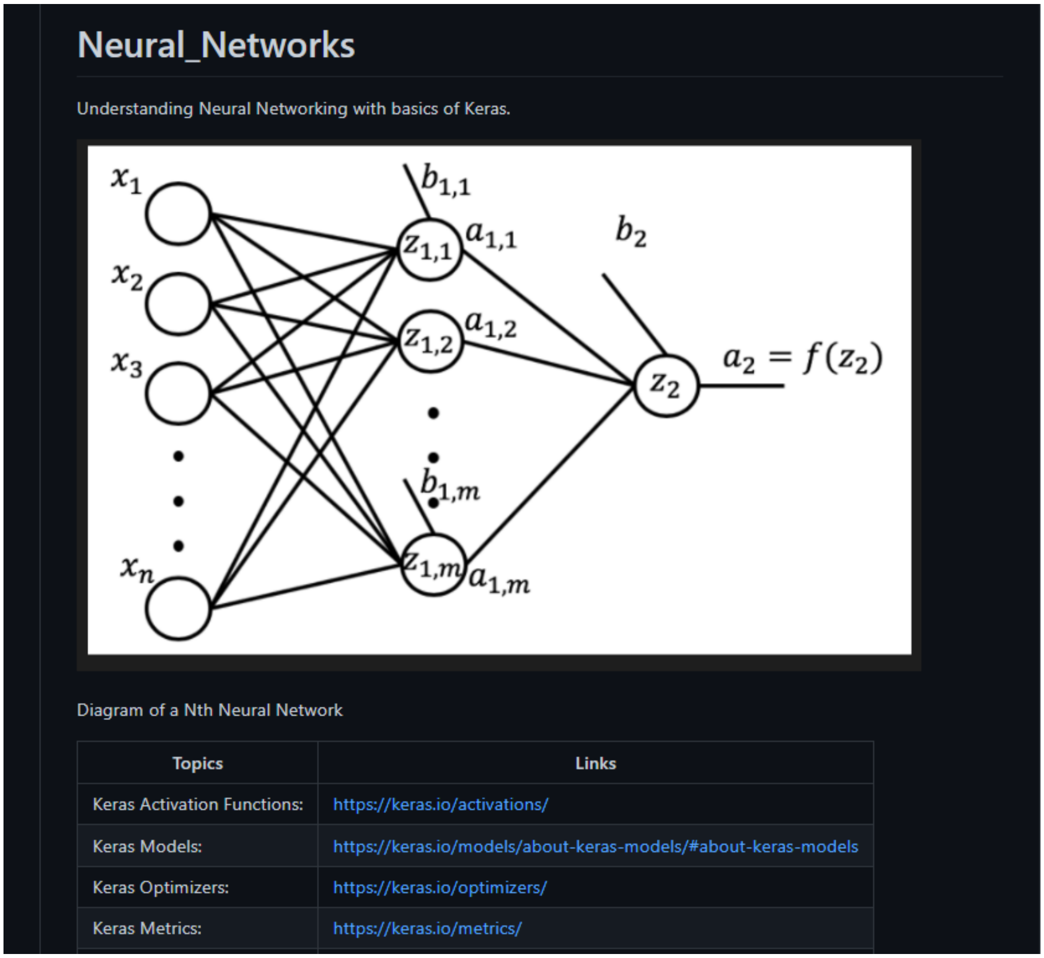
Astronomy has become a data-rich science; this transition is often labeled as: “Data Tsunami”. The data are not just increasing in size but also in complexity and dimensionality. Astroinformatics is a new field of science that has emerged from this technology-driven progress. Virtual Observatory, Machine Learning, Data Mining and Grid Computing are just a few examples of the new tools available to scientists. The new concept of data infrastructure named Virtual Observatory (VO) offers an ideal basis for dealing with distributed heterogeneous datasets.

The project aimed at using VO and Data Mining technologies to process data towards the classification of the host of faint radio sources. Multiwavelength data from X-ray, Ultra-Violet, Optical, and Infrared will be used to analyze and classify the host of faint radio population into different classes (i.e. Galaxies, Quasars, and Stars).

Figure 2.17: GalaxyML Repo

`https://github.com/ADGSTUDIOS/GalaxyML`

2.2.9.2 Neural Networks



Neural_Networks

Understanding Neural Networking with basics of Keras.

Diagram of a Nth Neural Network

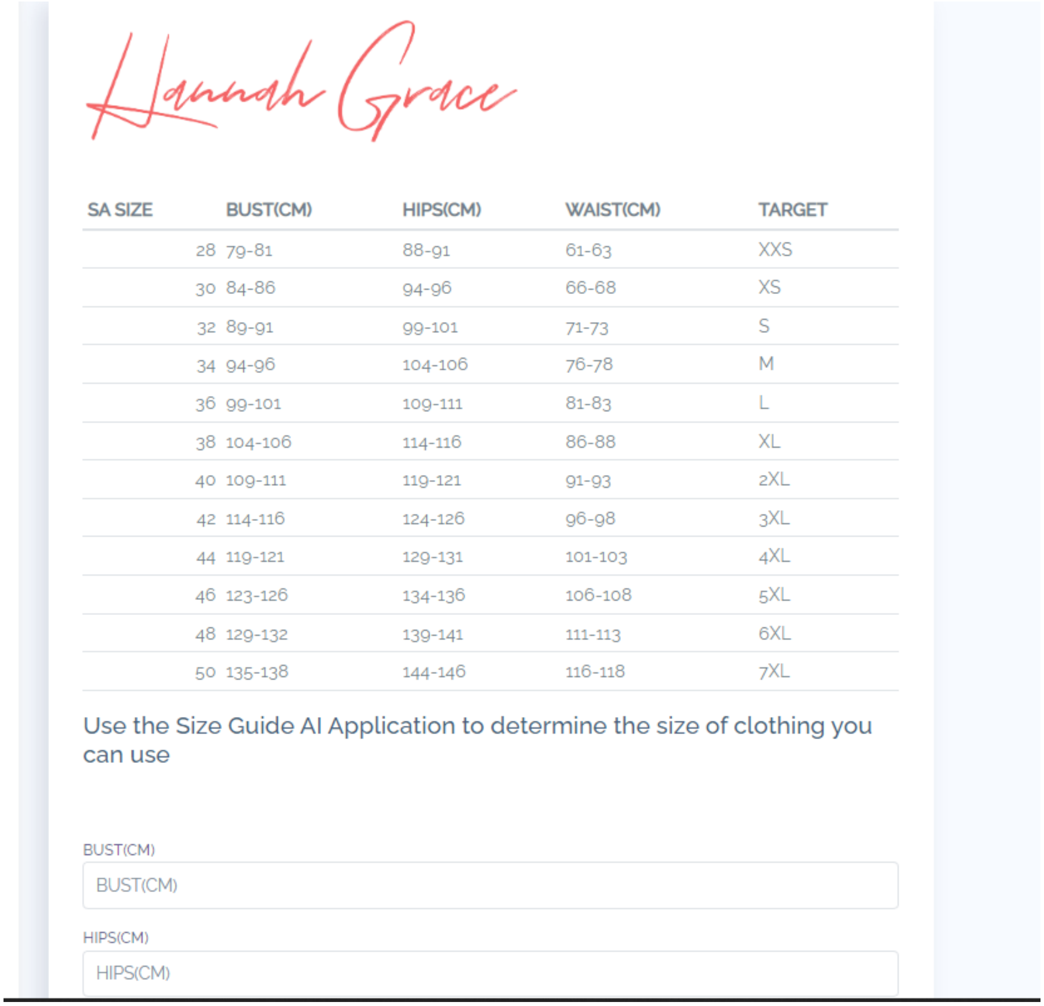
Topics	Links
Keras Activation Functions:	https://keras.io/activations/
Keras Models:	https://keras.io/models/about-keras-models/#about-keras-models
Keras Optimizers:	https://keras.io/optimizers/
Keras Metrics:	https://keras.io/metrics/

Figure 2.18: Neural Networks Repo

If you want to play with Neural Networks got great notebooks to play with @ `https://github.com/adgsenpai/Neural_Networks`

2.2.9.3 Machine Learning Application in ECommerce

Clothing Size Guide AI Application used for predicting clothing size given a users BUST(CM) and HIPS(CM)



Hannah Grace

SA SIZE	BUST(CM)	HIPS(CM)	WAIST(CM)	TARGET
28	79-81	88-91	61-63	XXS
30	84-86	94-96	66-68	XS
32	89-91	99-101	71-73	S
34	94-96	104-106	76-78	M
36	99-101	109-111	81-83	L
38	104-106	114-116	86-88	XL
40	109-111	119-121	91-93	2XL
42	114-116	124-126	96-98	3XL
44	119-121	129-131	101-103	4XL
46	123-126	134-136	106-108	5XL
48	129-132	139-141	111-113	6XL
50	135-138	144-146	116-118	7XL

Use the Size Guide AI Application to determine the size of clothing you can use

BUST(CM)

BUST(CM)

HIPS(CM)

HIPS(CM)

Figure 2.19: Hannah Grace Clothing Size AI Application

you can play with it @ `https://www.hannahgrace.co.za/size-guide/` this app has an accuracy of 100%

2.2.10 Forecasting with Prophet

2.2.10.1 What is Forecasting?

Forecasting is the process of predicting the future based on historical and current data. These can then be compared (resolved) to what actually happens. For example, a company might forecast revenue for the coming year and then compare it to actual results. Prediction is a related but broader term. Forecasting can refer to formal statistical methods that use *time series*, cross-sectional, or longitudinal data, as well as less formal judgmental methods or the prediction and resolution process itself. In hydrology, for example, the terms ”forecast” and ”forecasting” are sometimes used to refer to estimates of values at specific future times, whereas the term ”prediction” is used to refer to more general estimates. We can use forecasting in digital twins, businesses and the stock market. We will run time series analysis on the data to build a digital twin of any plant or simulation. A time series analysis is a method of analysing a collection of data points over a period of time. Instead of recording data points intermittently or randomly, time series analysts record data points at consistent intervals over a set period of time. If you refer back to Linear Regression, we will build a forecasting model equation to sketch a time series analysis forecasting graph based on concepts of *Linear Regression*.

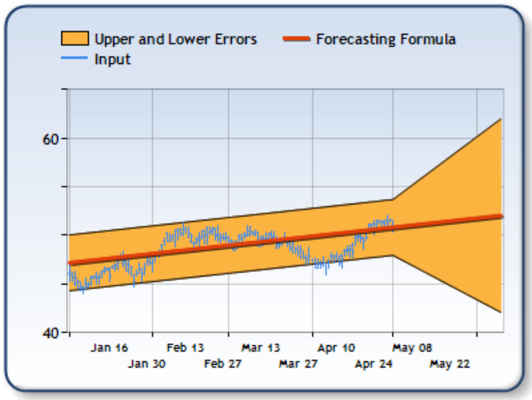


Figure 2.20: Time Series Graph

2.2.10.2 What is Prophet?

Prophet is a module for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well. In

2017, Facebook (now called Meta) released Prophet, an open-source forecasting tool in Python and R. The demand for high-quality forecasts often outpaces the analysts producing them. This situation was the motivation behind building a tool like Prophet that makes it easier for both experts and non-experts to deliver high-quality forecasts. Prophet has gained massive popularity among other forecasting tools. Prophet also tops among other Python time series packages when ranked by monthly downloads. It is supported in the Python and R language. Producing high-quality forecasts is a challenge. Businesses usually resort to either of the two options to create forecasts — they use completely automatic forecasting techniques or are entirely dependent on human analysts to produce high-quality forecasts. Both the options have their own drawbacks. In the case of complete automation, forecasting techniques produced are brittle and inflexible; incorporating useful assumptions becomes difficult. On the other hand, it is a challenge to find analysts who can create forecasts because it requires people with specialized data science skills with substantial experience. To remediate these challenges, Prophet attempts to offer the best of both worlds. Users are not left at the complete mercy of an automatic process; instead, an analyst with no training in time series can help improve and tweak forecasts using easily interpretable parameters. According to Meta, Prophet offers a straightforward way to create a 'reasonable and accurate forecast'. Further, the forecasts made by Prophet are customisable in a way that is intuitive to even non-experts.

2.2.10.3 How does Prophet work mathematically? + Prophet models explained mathematically.

The Prophet procedure is an additive regression model with four main components — a piecewise linear logistic growth curve trend; a yearly seasonal component modelled using Fourier series; a weekly seasonal component created using dummy variables; a user-provided list of important holidays. The procedure makes use of a decomposable time series model with three main model components: trend, seasonality, and holidays. This is an important technique for all types of time series analysis, especially for seasonal adjustment. It seeks to construct, from an observed time series, a number of component series (that could be used to reconstruct the original by additions or multiplications) where each of these has a certain characteristic or type of behaviour. Similar to a generalized additive model (GAM), with time as a regressor. In statistics, a generalized additive model (GAM) is a generalized linear model in which the linear response variable depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions. Prophet fits several linear and non-linear functions of time as components. In its simplest form;

$$y(t) = g(t) + s(t) + h(t) + e(t) \tag{2.21}$$

$g(t)$ - trend models non-periodic changes (i.e., growth over time)
$s(t)$ - seasonality presents periodic changes (i.e., weekly, monthly, yearly)
$h(t)$ - ties in effects of holidays (on potentially irregular schedules ≥ 1 day(s))
$e(t)$ - covers idiosyncratic changes not accommodated by the model

In other words, the procedure's equation can be written;

$$y(t) = piecewise_{trend}(t) + seasonality(t) + holiday_{effects}(t) + noise(t) \tag{2.22}$$

2.2.10.4 Trends

The procedure provides two possible trend models for $g(t)$, "a saturating growth model, and a piecewise linear model."

2.2.10.5 Saturating Growth Model

If the data suggests promise of saturation — i.e., one is wrestling constraints like: Power station capacity levels gigawatts (GW), Water Production Capacity (Megalitre), — setting growth= "*logistic*" is the move. Typical modelling of these nonlinear, saturating trends is basically accomplished;

$$g(t) = \frac{1}{1 + e^{-\frac{t-t_0}{\tau}}} \tag{2.23}$$

where t_0 is the time of the first data point, tau is the time constant, and t is the time.

alternatively in Prophet

$$g(t) = \frac{1}{1 + e^{-k(t-m)}} \tag{2.24}$$

where: C is the carrying capacity k is the growth rate m is an offset parameter There are two primary aspects of growth at Facebook (fluctuating carrying capacity and volatile rate of change) that are not captured in this simplified equation, though.

Carrying Capacity vs Time

First, as with many scalable business models carrying capacity is not constant — as "the population increases the capacity demands scale for plants the plants need to cater for the ever-growing population", so does the growth ceiling." Accounting for this is done by replacing the fixed capacity C with a time-varying capacity C(t).

Rate of Change vs Time
Second, the market does not allow for stagnant technology. Advances like those seen over the past decade in handheld devices, app development, and global connectivity, virtually ensure that growth rate is not constant. Because this rate can quickly compound due to new products, the model must be able to incorporate a varying rate in order to fit historical data.

Suppose there are S changepoints at times $\delta j, j = 1, ..., S$

Prophet defines a vector of rate adjustments;

$$\delta j \in \mathbb{R}^S \tag{2.25}$$

where: δj is the rate adjustment for the jth changepoint S is the number of changepoints

The rate at any time t is then the base rate k, plus adjustments up to that time;

$$k(t) = k + \sum_{j=1}^{S} \delta j \left\{ \frac{t - \delta j}{1 + \delta j} \right\} \tag{2.26}$$

This is represented more cleanly by defining a vector;

$$a(t) \in \{0, 1\}^S \tag{2.27}$$

Described by Meta
"The rate at time t is then $k + a(t)^T \delta$ When the rate k is adjusted, the offset parameter m must also be adjusted to connect the endpoints of the segments. The correct adjustment at changepoint j is easily computed as"

$$\delta j = \frac{m - \delta j - 1}{1 + \delta j - 1} \tag{2.28}$$

At last, the piecewise growth= "*logistic*" model is reached;

$$g(t) = \frac{1}{1 + e^{-k(t-m)}} \tag{2.29}$$

In application, the logistic growth model presented here is a special case of generalized logistic growth curves — which is only a single type of sigmoid curve — allowing the relatively straightforward extension(s) of this trend model to other families of curves.

2.2.10.6 Linear Trend with Changepoints

The second — much simpler and default — trend model is a simple Piecewise Linear Model with a constant rate of growth. It is best suited for problems without a market cap or other max in sight, and is set via $growth =' linear'$. Modelling the linear trend is easily realized with Prophet. In fact, not adjusting anything usually does the trick;

$$g(t) = (k + a(t)^T \delta)t + (m + a(t)^T \gamma) \tag{2.30}$$

where: k is the growth rate δ has the rate adjustments m is the offset parameter

and, to make the function continuous, γj is set to:

$$\gamma j = \frac{m - \gamma j - 1}{1 + \gamma j - 1} \tag{2.31}$$

I think in Metas case could be interpreted as:

$$-s_j \delta_j \tag{2.32}$$

2.2.10.7 Automatic Changepoint Selection

If known, the changepoints s_j can be specified by the user as dates of product launches and other growth-altering events (one example in the plants, measurements in the labs at a specific date), or, by default, changepoints may be automatically selected given a set of candidates.
"Automatic selection can be done quite naturally with the formulation in either model by putting a sparse prior on δ." - Meta
Often, it is advisable to specify a large number of changepoints (e.g., one per month for a several years history, in plants we typically have weekly data or daily data that's a lot of data, I worked with a dataset of 1 million samples of a plant) and use the prior:

$$\delta j \sim \mathcal{L}(0, \tau) \tag{2.33}$$

where:
τ directly controls the flexibility of the model in altering its rate.

Critical note: a sparse prior on the adjustments δ has no impact on the primary growth rate k, so as τ progresses to 0 the fit reduces to standard (not-piecewise) logistic or linear growth.

2.2.10.8 Trend Forecast Uncertainty

When the model is extrapolated past the history to make a forecast, the trend $g(t)$ will have a constant rate; the uncertainty in the forecast trend is estimated by extending the generative model forward.

The generative model for the trend is that there are;

- S changepoints
- over a history of T points
- each of which has a rate change $\delta j \sim \mathcal{L}(0, \tau)$

Simulation of future rate changes (that emulate those of the past) is achieved by replacing τ with a variance inferred from data.

"In a fully Bayesian framework this could be done with a hierarchical prior on τ to obtain its posterior, otherwise we can use the maximum likelihood estimate of the rate scale parameter" - Meta

$$\lambda = \frac{1}{S} \sum_{j=1}^{S} |\delta_j| \tag{2.34}$$

"Future changepoints are randomly sampled in such a way that the average frequency of changepoints matches that in the history" - Meta

$$\forall j > T, \left\{ \begin{array}{ll} \delta_j = 0 & w.p. \frac{T-S}{T} \\ \delta_j \sim \mathcal{L}(0, \tau) & w.p. \frac{S}{T} \end{array} \right\} \tag{2.35}$$

Thus, uncertainty in the forecast trend is measured by assuming the future will see the same average frequency and magnitude of rate changes that were seen in the history. Once λ has been inferred from the data, this generative model is deployed to "simulate possible future trends and use the simulated trends to compute uncertainty intervals." Prophet's assumption that the trend will continue to change with the same frequency and magnitude as it has in the history is fairly strong, so don't bank on the uncertainty intervals having exact coverage. As τ is increased the model has more flexibility in fitting the history and so training error will drop. Even so, when projected forward this flexibility is prone to produce wide intervals. The uncertainty intervals are, however, a useful indication of the level of uncertainty, and especially an indicator of over fitting.

2.2.10.9 Seasonality

The seasonal component $s(t)$ provides an adaptability to the model by allowing periodic changes based on sub-daily, daily, weekly and yearly seasonality. "Business time series often have multi-period seasonality as a result of the human behaviours they represent. For instance, a 5-day work week can produce effects on a time series that repeat each week, while vacation schedules and school breaks can produce effects that repeat each year. To fit and forecast these effects we must specify seasonality models that are periodic functions of t." Prophet relies on Fourier series to provide a malleable model of periodic effects. P is the regular period the time series will have (e.g., P = 365.25 for yearly data or P = 7 for weekly data, when time is scaled in days). Approximate arbitrary smooth seasonal effects are therefore tied in with a standard Fourier series;

$$s(t) = \sum_{n=1}^{N} a_n \cos\left(\frac{2\pi n}{P}t\right) + \sum_{n=1}^{N} b_n \sin\left(\frac{2\pi n}{P}t\right) \tag{2.36}$$

"Fitting seasonality requires estimating the $2N$ parameters $\beta = [a1, b1, \ldots, aN, bN]^T$. This is done by constructing a matrix of seasonality vectors for each value of t in our historical and future data, for example with yearly seasonality and N= 10" - Meta

$$X(t) = (\cos(\frac{2\pi(10)t}{365.25}), ..., \sin(\frac{2\pi(10)t}{365.25})) \tag{2.37}$$

Meaning the seasonal component is;

$$s(t) = X(t)\beta \tag{2.38}$$

In the generative model, Prophet takes $\beta \sim Normal(0, \sigma^2)$ to impose a smoothing prior on the seasonality. Truncating the series at N applies a low-pass filter to the seasonality, so, albeit with increased risk of overfitting, increasing N allows for fitting seasonal patterns that change more quickly. "For yearly and weekly seasonality, we have found N = 10 and N = 3 respectively to work well for most problems. The choice of these parameters could be automated using a model selection procedure such as AIC." Meta

2.2.10.10 Holidays and Events

Impact of a particular holiday on the time series is often similar year after year, making it an important incorporation into the forecast. The component $h(t)$ speaks for predictable events of the year including those on irregular schedules. To utilize this feature, the user needs to provide a custom list of events. Fusing this list of holidays into the model is made straightforward by assuming that the effects of holidays are independent.

South African Public Holidays in the year 2022	
Holiday name	Date
New Year's Day	Sat, 01 Jan 2022
Human Rights Day	Mon, 21 Mar 2022
Good Friday	Fri, 15 Apr 2022
Family Day	Mon, 18 Apr 2022
Freedom Day	Wed, 27 Apr 2022
Workers' Day	Mon, 02 May 2022
Youth Day (in South Africa)	Thu, 16 Jun 2022
National Women's Day	Tue, 09 Aug 2022
Heritage Day	Sat, 24 Sept 2022
Day of Reconciliation	Fri, 16 Dec 2022
Boxing Day	Mon, 26 Dec 2022
Christmas Day	Likely Mon, 26 Dec 2022

We could define a function called public holidays and generate a list of public holidays. We can implement an API maybe also to get the holidays with dates in any country at a given time interval. In layman's terms our equation and data structure (as a dictionary mathematically a set) will be described as below:

$$\begin{aligned} holidayAPI(countryName, year) = \{&i_0 HolidayName : i_0 HolidayDate \\ &, i_1 HolidayName : i_1 HolidayDate, \ldots \\ &, i_{lastindex} HolidayName : i_{lastindex} HolidayDate\} \end{aligned}$$

Our API can be described like that above returning all the holidays for a given year. Once we got an algorithm like that going.

$$h(t) = \sum_{t=startDate}^{endDate} \{holidayAPI\{countryName, t\}\} \tag{2.39}$$

We could define a function called public holidays and generate a list of public holidays. We can implement an API maybe also to get the holidays with dates in any country at a given time interval. In layman's terms our equation and data structure (as a dictionary mathematically a set) will be described as below:

We finalize the algorithm to get all the holidays during a time interval. But in machine learning we don't really need to know the holiday name so formally how we work this data mathematically is shown below formally we just use the past and future dates of the holiday. For each holiday i, let D_i be the set of past and future dates for that holiday. Then add an indicator function representing whether time t is during holiday i, and assign each holiday a parameter K_i which is the corresponding change in the forecast.

This is done in a similar way as seasonality by generating a matrix of regressors;

$$Z(t) = [1(t \in D_1), \ldots, 1(t \in D_L)]] \tag{2.40}$$

and taking,

$$h(t) = Z(t)\kappa \tag{2.41}$$

As with seasonality, Prophet uses a prior $\kappa - Normal(0, v^2)$.

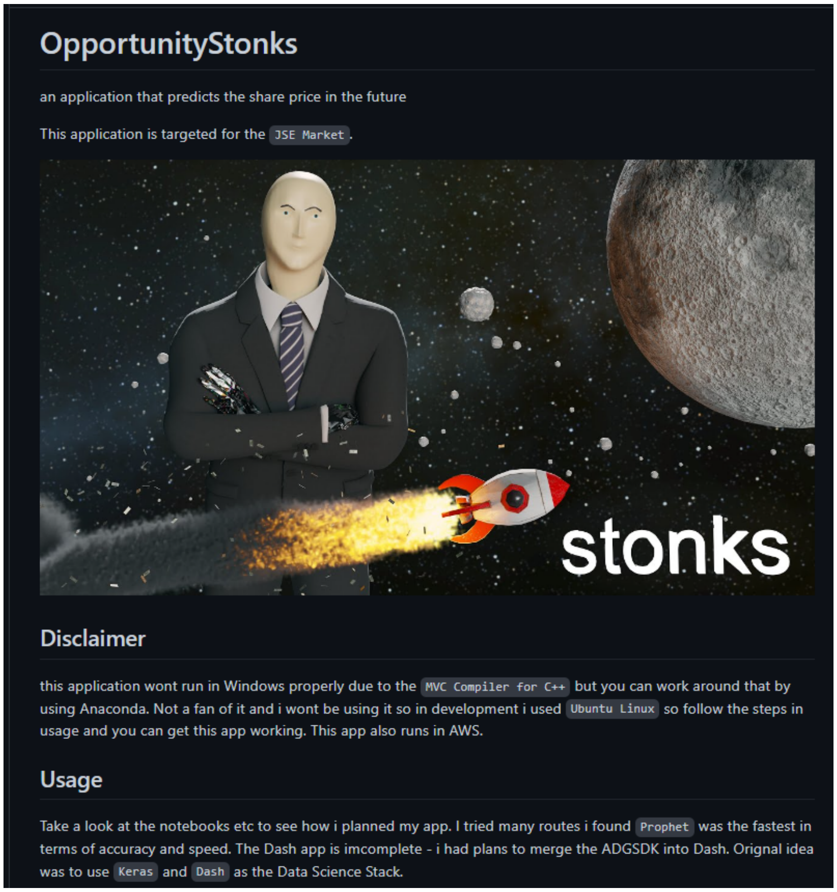
"It is often important to include effects for a window of days around a particular holiday, such as the weekend of Thanksgiving. To account for that we include additional parameters for the days surrounding the holiday, essentially treating each of the days in the window around the holiday as a holiday itself." - Meta

2.2.10.11 Prophet Conclusion

Ultimately, Prophet was engineered to help analysts with a variety of backgrounds produce more forecasts with less time invested towards doing so. This was achieved by sticking to a relatively plain model.
"We use a simple, modular regression model that often works well with default parameters, and that allows analysts to select the components that are relevant to their forecasting problem and easily make adjustments as needed." - Meta

2.2.10.12 Prophet Resources

Official Prophet Meta Website
`https://facebook.github.io/prophet/`
Official Developer Documentation
`https://facebook.github.io/prophet/docs/quick_start.html`
Blog
`https://research.facebook.com/blog/2017/02/prophet-forecasting-at-scale/`
Prophet Thesis Paper
`https://peerj.com/preprints/3190/`
I am not allowed to disclose my industrial work of a digital twin which I built of a plant but I do have an example which you can use on how I used prophet for building Opportunity Stonks.



OpportunityStonks

an application that predicts the share price in the future

This application is targeted for the `JSE Market`.

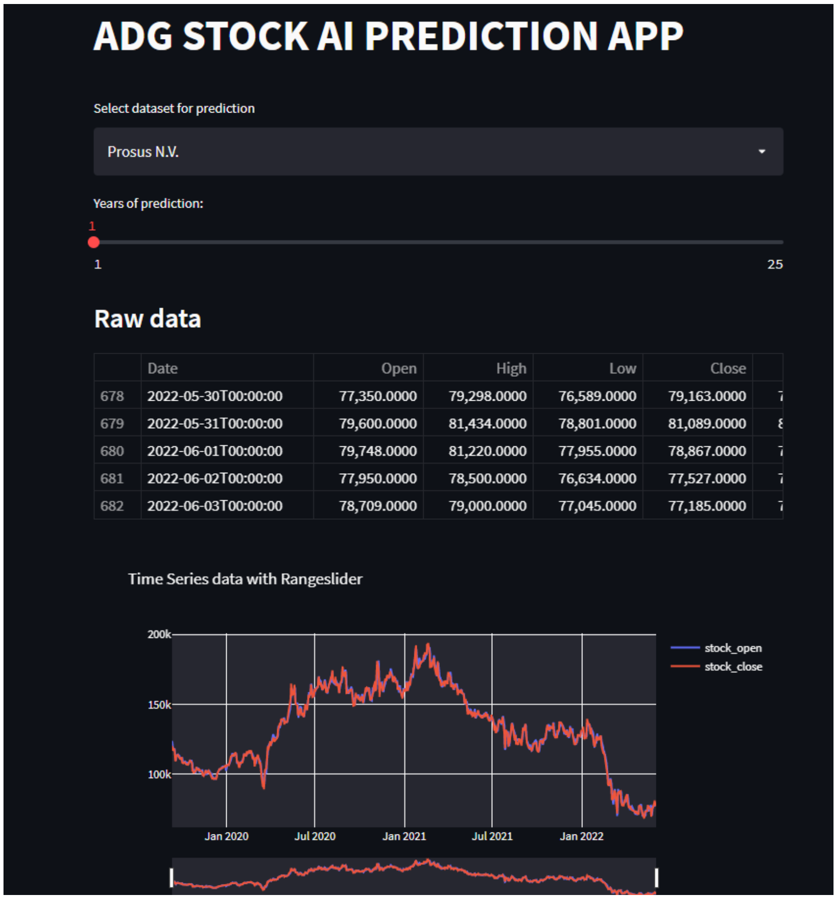
Disclaimer

this application wont run in Windows properly due to the `MVC Compiler for C++` but you can work around that by using Anaconda. Not a fan of it and i wont be using it so in development i used `Ubuntu Linux` so follow the steps in usage and you can get this app working. This app also runs in AWS.

Usage

Take a look at the notebooks etc to see how i planned my app. I tried many routes i found `Prophet` was the fastest in terms of accuracy and speed. The Dash app is imcomplete - i had plans to merge the ADGSDK into Dash. Orignal idea was to use `Keras` and `Dash` as the Data Science Stack.

Figure 2.21: GitHub Repo for Opportunity Stonks `https://github.com/adgsenpai/OpportunityStonks`



ADG STOCK AI PREDICTION APP

Select dataset for prediction

Prosus N.V.

Years of prediction:

1

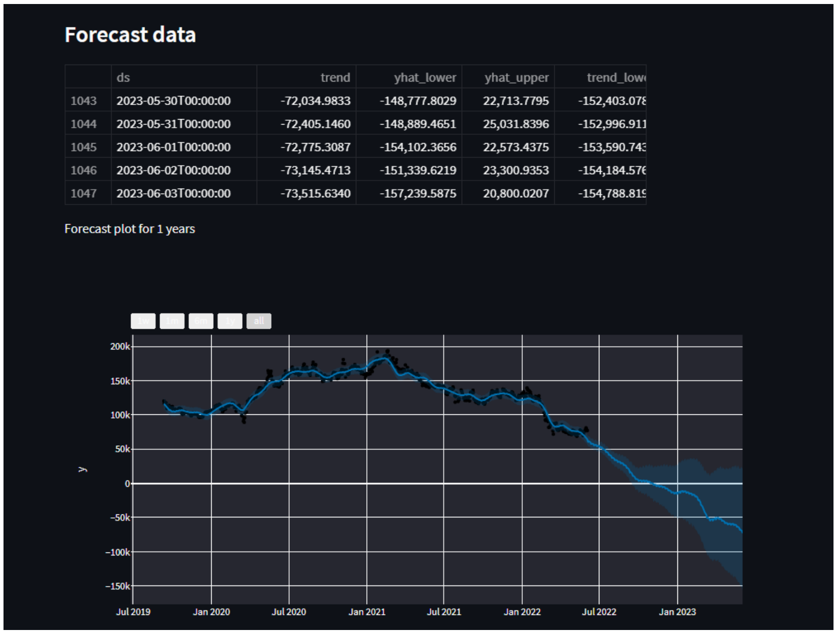
1 25

Raw data

	Date	Open	High	Low	Close	
678	2022-05-30T00:00:00	77,350.0000	79,298.0000	76,589.0000	79,163.0000	7
679	2022-05-31T00:00:00	79,600.0000	81,434.0000	78,801.0000	81,089.0000	8
680	2022-06-01T00:00:00	79,748.0000	81,220.0000	77,955.0000	78,867.0000	7
681	2022-06-02T00:00:00	77,950.0000	78,500.0000	76,634.0000	77,527.0000	7
682	2022-06-03T00:00:00	78,709.0000	79,000.0000	77,045.0000	77,185.0000	7

Time Series data with Rangeslider

Figure 2.22: Screenshot of Opportunity Stonks



Forecast data

	ds	trend	yhat_lower	yhat_upper	trend_low
1043	2023-05-30T00:00:00	-72,034.9833	-148,777.8029	22,713.7795	-152,403.078
1044	2023-05-31T00:00:00	-72,405.1460	-148,889.4651	25,031.8396	-152,996.911
1045	2023-06-01T00:00:00	-72,775.3087	-154,102.3656	22,573.4375	-153,590.743
1046	2023-06-02T00:00:00	-73,145.4713	-151,339.6219	23,300.9353	-154,184.576
1047	2023-06-03T00:00:00	-73,515.6340	-157,239.5875	20,800.0207	-154,788.819

Forecast plot for 1 years

Figure 2.23: Time Series Analysis/Forecast of Stock using Prophet @ Opportunity Stonks

2.3 Building Digital Twins

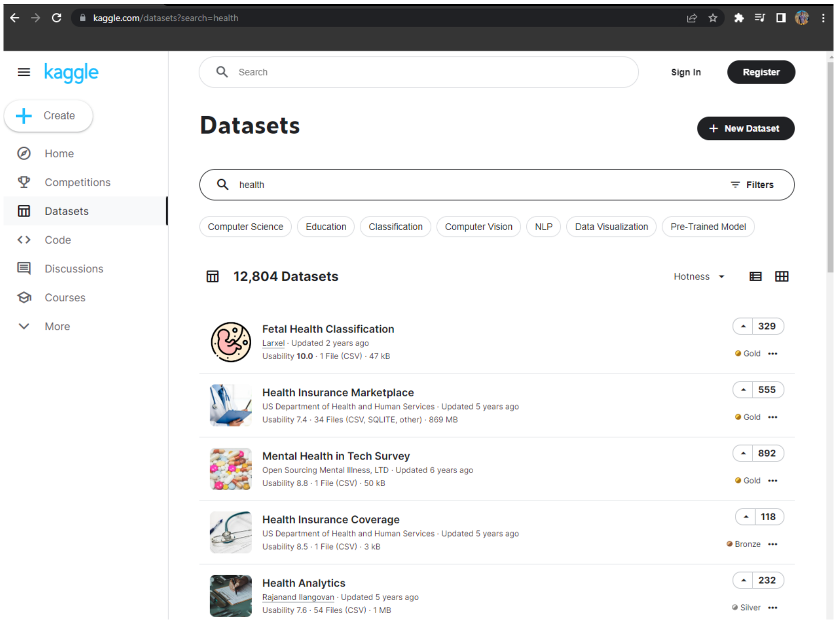
2.3.1 Revising Methodology to Build Digital Twins

The first step to building digital twins is that you need to collect your data depending on the type of digital twin you are building.

Examples:

- If you are building a digital twin of a plant – you will need to get nth number of records over a t period of that plant. You can be at the plant taking measurements or asking permission at a company for the datasets.
- If you are building a general digital twin of anything for fun or research. You can get a dataset publicly from the internet. A good start to any data science project is playing around with datasets implementing

algorithms and demo apps to get a grip on projects. A good website for datasets is `https://www.kaggle.com/datasets`



12,804 Datasets

Fetal Health Classification
Larxel · Updated 2 years ago
Usability **10.0** · 1 File (CSV) · 47 kB
329
Gold

Health Insurance Marketplace
US Department of Health and Human Services · Updated 5 years ago
Usability 7.4 · 34 Files (CSV, SQLITE, other) · 869 MB
555
Gold

Mental Health in Tech Survey
Open Sourcing Mental Illness, LTD · Updated 6 years ago
Usability 8.8 · 1 File (CSV) · 50 kB
892
Gold

Health Insurance Coverage
US Department of Health and Human Services · Updated 5 years ago
Usability 8.5 · 1 File (CSV) · 3 kB
118
Bronze

Health Analytics
Rajanand Ilangovan · Updated 5 years ago
Usability 7.6 · 54 Files (CSV) · 1 MB
232
Silver

Figure 2.24: Kaggle Website for datasets - query for ”Health”

The second step is that you must visualize your dataset, Learn about your parameters/variables. You can do that by sketching graphs it can be different types of graphs but different graphs might work better to visualize that relationship. The third step is that you must try different algorithms and play around with Machine Learning tools and Time Series Analysis Tools. Make sure your data is clean! The final step is that you can combine steps 2 and 3 into a cool looking dashboard for companies to play with on the web. Why the web? Because it is cross-platform.

2.3.2 Python Modules that can help you out when building digital twins

Pandas - Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

It is free software released under the three-clause BSD license. We normally use that for .xlsx files, .csv, .sql databases and etc.
Prophet – Forecasting, Time Series Analysis Module by Meta, as discussed above in the paper
Streamlit - Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
Keras - Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.
Tensorflow - TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.
Dash - Dash is a Python module or framework used to create many analytical web applications, and we can build analytical dashboards using a dash framework. With the help of the Dash module, we can easily create very fast and responsive web dashboards, which are also very good to look at (Having a Great User Interface).
Numpy - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
SciPy - SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.
Seaborn - Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframe and the Pandas library. The graphs created can also be customized easily
matplotlib.pyplot - is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Chapter 3

Conclusion

3.1 Conclusion

After completing my first digital twin, I realized it was simple if I had a strong mathematical/statistical background and previous industrial experience with Machine Learning. You can't create a digital twin unless you've worked with the technologies I've covered in this paper. These technologies include, for example (Building Dashboards, Machine Learning, Visualizing Data etc). Prior knowledge is recommended when creating digital twins (building or experimenting making machine learning apps) Aside from that, I've demonstrated that if you're a data scientist, programmer, or mathematician, creating digital twins is simple. You don't have to pay a software development company specializing in digital twins hundreds of thousands of dollars. Your data science team possesses all of the necessary skills to create this digital twin. I hope this paper has sparked your interest in creating digital twins.