Efficient storage and retrieval of horoscope data on a computer system.

A case study using Python and MongoDB

PRITHWIS MUKERJEE[1]

Praxis Business School, Kolkata, India

Abstract*: A horoscope chart consists of a lot of information that goes well beyond the date and time of birth or even the location of grahas in rashis. A competent analysis requires many other pieces of information like bhavs, lords, states of exaltations and debilitations, aspects and conjuncts. While processing, or calculating this data is not difficult, storing them in a manner that allows easy retrieval based on specific parameters is a non-trivial problem. In this paper we demonstrate how horoscopes can be converted into instances of objects (in this case JSON documents) with high data dimensions using the Python programming language and then stored on the MongoDB platform for selective retrieval.*

Keywords: horoscope, software, database, mongodb, json, python

Introduction

Astrology is based on the analysis and detection of correlated patterns. Causation and correlation are two different ways of studying related phenomena. Switching on an oven is the obvious cause for a cake to be baked but the consumption of flour, sugar, eggs, butter and electricity in the kitchen is strongly correlated to the appearance of a cake. These six events are part of a pattern and the presence of the first five will be correlated to or lead to a prediction of the presence of the sixth. Viewed from this perspective of pure correlation, astrology is no different from data science where a pattern of data - in this case the positions of planets at birth – is used to calculate the possibility of certain outcomes.

Regular data science uses a set of digital data, preferably numbers, to predict a certain outcome. For example, data regarding age, income, gender, ownership of vehicles, cars, bank account and other personal information can lead to the prediction of whether or a person will repay a loan (or purchase a car). This is a straightforward data science or more precisely, a machine learning problem. There is a lot of mathematics that can justify these predictions, even though there is no

[1] prithwis@praxis.ac.in

direct cause:effect relationship between any of these data points and the event that is being predicted. The presence of past data is used to establish significant correlations between these events and forms the basis of successful predictions.

When it comes to astrological data, there is a major difference and a significant difficulty. The data used in machine learning, say the value of age, or income, generally has the same significance across all samples used to identify the correlations. But in the case of astrology, the position of a *graha*, say Mars in Mesh *rashi*, has a different significance, depending on say (but not limited to), (a) the *bhav* number of the same Mesh *rashi* for this person, (b) whether the same *bhav* is aspected by another *graha*, (c) whether the same *bhav* is aspected by the *lord* of another *bhav*, (d) whether the *graha* Mars is conjuncted by another *graha* or (e) or by another *lord* and so on. So the location of a *graha* in one *rashi* is not just a single point of data, (like the value of age or income in data science) but just one component of a piece of data of a higher dimensionality, that needs to be factored in while establishing correlations. [We assume that astrologers reading this article will understand the meaning of *graha, rashi, bhav, lord* etc. while others will read these as additional dimensions of data] In machine learning and data science this problem is usually addressed with dummy variables that are not present in the original data set. Our approach uses a similar strategy by employing dummy variables as explained later.

Traditionally, this complex, multi-dimensional data is represented as a visual artefact, the horoscope chart, that astrologers use to spot patterns. Given the very large number of dimensions involved, this is not easy. Though the process is totally algorithmic, or rule based, it is quite laborious and only an experienced astrologer can determine and identify the higher dimensional data with speed and accuracy. Those who cannot do so, fail to spot all the patterns and end up with erroneous predictions. This is where we are naively tempted to use the processing power of modern computers, but as we know, the accuracy of the output of a computer program depends almost entirely on the quality of data with which it works. A human astrologer works with visual data available in the form of a horoscope chart, but a computer cannot use the image of a chart to look for patterns. To be of any use, the visual chart of a horoscope must be converted into a set of numbers that can be stored and processed on a machine. The data that describes a horoscope can be defined at multiple levels of complexity. This article describes a mechanism for handling and processing this high-dimension data using a combination of Python programs in Jupyter notebooks and the MongoDB database.

Data Structures

At the lowest level, L0, a horoscope is uniquely identified by just **five** pieces of information: Date and Time of birth, Latitude and longitude of the place of birth and the TimeZone offset of the local time from UTC (or Universal Coordinated Time).

Based on this L0 data, we can use an ephemeris to calculate the next level, L1, *astronomical* data that consists of the longitude of 10 *grahas* (5 true planets, *sun*, *moon* and three special virtual objects, namely *lagna*, *rahu* and *ketu*) 5 of the 10 *graha*s can have temporary retrograde motion so in addition to their longitude, we need to record this fact. So, level L1 data consists of **fifteen** variables, where 10 are numeric values of longitude and 5 are Boolean variables (that can take a value of *true*, if the *graha* is retrograde and *false*, if they are not).

Based on this purely *astronomical* L1 data, astrologers calculate their own *astrological* L2, data based on the principles of the *kaal-chakra* (or zodiac) where longitudes in level L1 data are replaced by corresponding *rashi* numbers (or names) that show the *rashi* where each *graha* resides.

The conversion of level L0 data to level L1 and then level L2 is quite straightforward and there are computer programs and websites that will do this calculation without any difficulty. *Computerised* Horoscope charts usually present this level L2 data in a variety of styles, namely, the Bengal Style, the North Indian Style (where in both cases, the *rashis* are arranged counterclockwise) and the South Indian style (where the *rashis* are arranged clockwise).

Newspaper columns and websites that publish monthly and weekly predictions do so on the basis of just *one, single* component of L2 data – the *rashi* position of either the Sun (in 'western' astrology) or the Moon (in 'Indian' astrology) – and hence are grossly inaccurate.

This is where we introduce the dummy variables in the form of level L3 data.

High Dimension Data

From Level L2, a series of increasingly complicated set of level L3, data points need to be calculated for *positions*, *aspects* and *conjuncts* :

Basic:

L3.1 : 12 *bhav* numbers - Based on the position of the *lagna*, each *rashi* gets a *bhav* number
L3.2 : 12 *bhav lords* - Each *bhav* gets a *graha* identified as its *lord* based on its *rashi*
L3.3 : 9 Locations of planets in terms of the *bhav* where the *graha* resides
L3.4 : 12 Locations of *lords* in terms of the *bhav* where the *lord* resides

Positions:

L3.5 : 9 Booleans that specify whether a *graha* is exalted
L3.6 : 9 Booleans that specify whether a *graha* is debilitated
L3.7 : 9 Booleans that specify whether a *graha* is at MoolTrikana

L3.8 : 9 Booleans that specify whether a *graha* is in Friendly *rashi*
L3.9 : 9 Booleans that specify whether a *graha* is in Hostile *rashi*
L3.10 : 9 Booleans that specify whether a *graha* is in Neutral *rashi*
L3.11 : 12 Booleans that specify whether a *lord* is exalted
L3.12 : 12 Booleans that specify whether a *lord* is debilitated
L3.13 : 12 Booleans that specify whether a *lord* is at MoolTrikana
L3.14 : 12 Booleans that specify whether a *lord* is in Friendly *rashi*
L3.15 : 12 Booleans that specify whether a *lord* is in Hostile *rashi*
L3.16 : 12 Booleans that specify whether a *lord* is in in Neutral *rashi*

Aspects:

L3.17 : 10 *sets* of *graha* Aspects where each set consists of *graha*s aspecting one *graha*
L3.18 : 10 *sets* of *graha* AspectedBY where each set consists of *grahas* that are aspected by one *graha*. Note that A aspecting B does not necessarily imply that B aspects A. There are rules that govern aspects.
L3.19 : 12 *sets* of *bhav* Aspects where each set consists of *graha*s aspecting a *bhav*

Conjunct:

L3.20 : 10 *sets* of *graha* - *graha* Conjuncts where each set consists of *graha*s that conjunct each *graha*
L3.21 : 12 *sets* of *lord* - *graha* Conjuncts where each set consists of *grahas* that conjunct each *lord*
L3.22 : 12 *sets* of *lord* - *lord* Conjuncts where each set consists of *lord*s that conjunct each *lord*

Beginning with only 5 pieces of data in level L0, we move to 15 pieces of data in level L1 and L2 and then 22 additional pieces of data in level L3. This transformation from L0 to L3 is done as per the rules given in Rao & Rao [1]. The critical issue here is that unlike L0, L1, L2 data, L3 data structures are not elemental (that is, consisting of just a single number or Boolean) but often collections or *sets* of associated data. *So, in effect, L3 data has more than the 22 components that* are listed here. This means that a *horoscope is not merely a pattern drawn on a piece of two-dimensional paper but is an instance of an object of very high dimensionality*.

Storing and processing this kind of an object that has data with high dimensionality needs programs that are an order of magnitude more complicated than most *Computer Horoscope* programs that are generally available.

Python and MongoDB

Parashar21 is a collection of python programs, developed on the Google Colab platform, that convert L0 data into L3 data. But the derived level L3 data needs to be stored in a manner that will allow easy and intuitive access. Storing even level L2 data in a computer system poses significant problems.

Horoscope charts, that are a visual representation of level L2 data, can of course be stored as image files in png or jpg format. However, this conversion to an image format leads to an immediate loss of information, because an image file cannot be identified, and retrieved, on the basis of the position of a *graha* in a particular *rashi*. For example, if we want to retrieve horoscopes that have the moon in *Meen rashi* and sun in *Makar rashi*, it cannot be done if the chart is stored as a simple image file. This is because the image file only stores spatial information of the proximity of the phrases 'Mo' and 'Su' to certain squares and triangles merely *on the diagram* without the semantic information that connects the phrases 'Mo' and 'Su' to the *grahas* moon and sun, or the squares to *Meen* or *Makar rashis*.

Level L2 data can of course be stored in a traditional flat file or a relational database like Oracle or MySQL without loss of information but when we move to level L3, the storage and retrieval becomes extremely complicated. This is because relational databases can only store elemental data, and not sets of data, in any field. Requirements to retrieve horoscopes that have, for example, moon *in the first bhav* and *lagna aspected by saturn*, would be almost impossible to fulfill. But this is precisely the kind of pattern that we are looking for.

This is where we introduce MongoDB, a very popular and widely used document storage platform as the database of choice. MongoDB stores information as *documents* in the *JSON* format that allows sets to be defined. MongoDB and the JSON format are very widely used in the modern software industry, and we will now demonstrate their utility in astrology through a representative case study.

Case study

The Astro-Databank Wiki [2] has a database of horoscope related information collected by Lois Rodden that is available in the public domain. This database consists of html pages for every individual that contains level L0 information along, six additional data fields containing data about their vocation (or profession) and a quality tag that indicates the estimated level of accuracy of the date and time of birth. From these html pages we extracted 33,662 pieces of (the most accurate) AA rated horoscope data along with three of the six vocation parameters and converted them into a CSV text file.

The Parashar21 python programs were used to convert this data level L0 data into level L3 information and this was stored in a MongoDB database as 39,667 JSON documents.

On this database, two kinds of retrieval tests were performed:

1. Random horoscopes were selected, based on arbitrary parameters and computed values of the 22 odd level L3 variables and printed for manual comparison against the corresponding visual charts. No errors were detected in the samples that were chosen for review. Since no errors were detected, we assume that the computation is correct, even though, as in the case of all software, absence *of evidence (of errors) is no evidence of absence (of errors).*
2. The query facilities of MongoDB were used to locate horoscopes that meet certain requirements. Samples from the retrieved horoscopes were printed and it was found that the queries were indeed giving correct results as shown below:

We first give the English version of the query followed by the same query using the MongoDB query language:

—----------

Retrieve charts that have -
Lagna aspected by *Saturn*

{"GAspectedBy2.La": {"$in": ["Sa"]}}

9991 Charts selected from 39667. Random 5 charts printed.

—----------

Retrieve charts that have -
Lagna aspected by *Saturn AND*
Exalted *Jupiter*

{"$and": [{"exaltG.Ju": {"$eq": true}}, {"GAspectedBy2.La": {"$in": ["Sa"]}}]}

874 Charts selected from 39667. Random 5 charts printed.

—----------

Retrieve charts that have -
Lagna aspected by *Saturn AND*
Exalted *Jupiter AND*
Sun conjunct with *Mercury*

{"$and": [{"exaltG.Ju": {"$eq": true}}, {"GAspectedBy2.La": {"$in": ["Sa"]}}, {"GConjunctsG2.Su": {"$in": ["Me"]}}]}

440 Charts selected from 39667. Random 5 charts printed.

—----------

Retrieve charts that have -
Lagna aspected by *Saturn AND*
Exalted *Jupiter AND*
Sun conjunct with *Mercury AND*
Moon in *Bhav* 1

{"$and": [{"exaltG.Ju": {"$eq": true}}, {"GAspectedBy2.La": {"$in": ["Sa"]}}, {"GConjunctsG2.Su": {"$in": ["Me"]}}, {"GrahaBhava.Mo": {"$eq": 1}}]}

41 Charts selected from 39667. Random 5 charts printed.

—----------

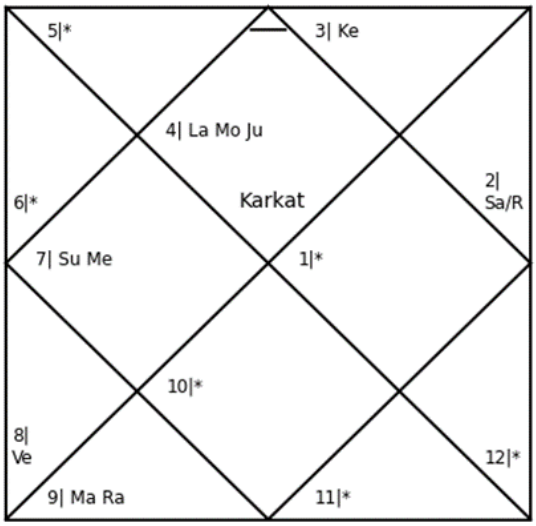
Retrieve charts that have -
Lagna aspected by *Saturn AND*
Exalted *Jupiter AND*
Sun conjunct with *Mercury AND*
Moon in *bhav* 1 *AND*
4th *lord* in *bhav* 5

{"$and": [{"exaltG.Ju": {"$eq": true}}, {"GAspectedBy2.La": {"$in": ["Sa"]}}, {"GConjunctsG2.Su": {"$in": ["Me"]}}, {"GrahaBhava.Mo": {"$eq": 1}}, {"LordBhav.4": {"$eq": 5}}]}

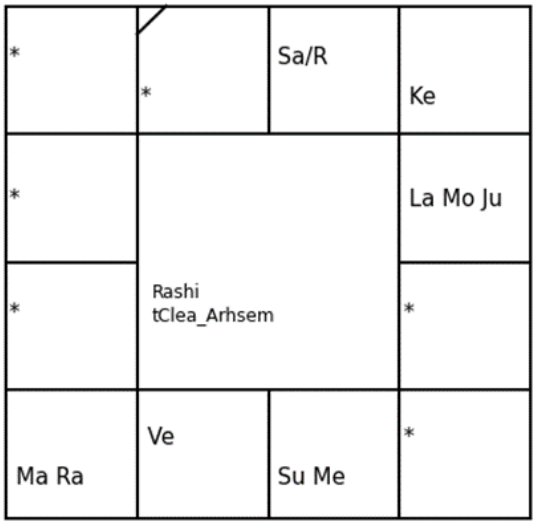
2 Charts selected from 39667. Random 2 charts printed.

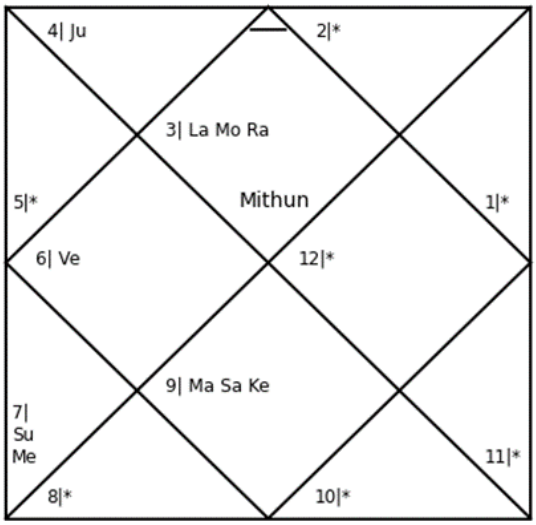
—----------

The two North India and South style charts generated by the program for two horoscopes retrieved from the MongoDB database from the last query with 5 conditions are shown here. The full reports generated by all 5 queries in three styles (Bengal, North India, South India) are available in the GitHub repository.

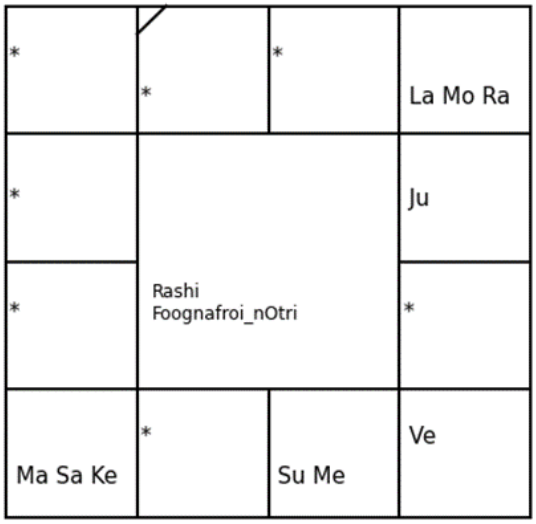


Rashi | tClea_Arhsem





Rashi | Foognafroi_nOtri



Discussion

The total number of horoscopes stored in the database was 39,667. In the five queries that were tested, the number of conditions were increased from 1 to 5. Based on these queries, the following number of horoscopes were retrieved. However, to save space, only 5 charts – chosen at random – were printed where the number retrieved was more than 5.

Number of Conditions	Number of Charts Retrieved	Number of Charts Printed
1	9991	5
2	874	5
3	440	5
4	41	5
5	2	2

The 2 charts retrieved in the last query were examined visually and checked manually and it is noted that they do indeed meet all 5 conditions. Hence, we conclude that the scheme and the associated programs do meet the current requirements and expectations of the project.

Data and Tools Used

The primary data was sourced from the Astro-Databank Wiki Project [2] This has been collected, cleaned and reformatted into level L0 data and is stored as a CSV file in a publicly available Google Drive[2]. The Python programs are available at the GitHub repository[3] under open-source GNU GPL 3 license. The sample reports referred to in this paper are also available in the same repository[4]. The MongoDB database instance is hosted at Clever Cloud [3] but any other service provider can be used with its own user id and password.

Scope of Future Work

Now that we have a way to store and retrieve complex chart information in a format that allows for logical retrieval, there is scope for two kinds of future work.

First, after level L3, we can calculate, codify and store higher levels of data. For example, in level L4 we could calculate and store information about *kendras, trikons, panapharas* and *apoklimas* and in level L5 we could store information on different types of *yogs*, like *dhanyog* and *rajyog*. Extensions to the current python code and increase in the number of fields in the mongodb database would easily cater to these requirements of higher-level data.

But more important would be to build a database of 'tagged' horoscopes. The current Astro Bank database has six descriptive fields with information about vocation and vocation details for up to

[2] https://drive.google.com/uc?id=1dlvqcygJJh0CfyY_X2MLrWm-W9U0z_6l
[3] https://github.com/prithwis/parashar21
[4] https://github.com/prithwis/parashar21/tree/main/Sample%20Reports%202

three vocations. We would need to generalise this by tagging horoscopes with additional information like education, wealth, fame, health etc. Creating such a database would be the first step towards discovering and validating patterns of data that are correlated to life outcomes.

The astronomical observations that were made with the naked eye and recorded in a vast database by Tycho Brahe and his sister Sophia were the basis for the formulation of Kepler's Laws of Planetary motion, that in turn led to Newton's laws of gravitation and the emergence of European science in the eighteenth century. Today, the same laws can be deduced very easily and quickly from NASA data [6].

Similarly, a database built on the principles described in this article would lead to a resurgence of Hindu astrology in the twenty-first century. That is the goal of Parashar21.

Supplementary Information

In the GitHub repository, there are three main programs (or Colab Notebooks). The *Cast_Load* program reads basic L0 data from a CSV file, uses the pyswisseph [4] python package and generates the L3 JSON objects that are stored in a MongoDB database as well as in a JSON file. The *Pull_Print* program retrieves selective data from the MongoDB database and generates reports. For those not having access to a MongoDB instance, an alternative *Pull_Print_StandAlone* program is provided. This installs a local, non-persistent MongoDB instance within the Google Colab VM, reads in the JSON data, loads the data into the local MongoDB database and demonstrates the execution of the queries described in this paper. Running the *Pull_Print_StandAlone* program requires a modern browser and the knowledge of executing Python programs in a Jupyter Notebook format in a Google Colab VM environment.[5] The user should navigate to the notebook file, open it and then press the <open with Colab> button that is visible.

References:

[1] Rao, KN, Ashu Rao, K, "*Learn Hindu Astrology Easily*"
[2] https://www.astro.com/astro-databank/Main_Page
[3] https://www.clever-cloud.com/
[4] https://pypi.org/project/pyswisseph/
[5] https://www.mygreatlearning.com/blog/google-colab-tutorial/
[6] Springsteen, P., Keith, J., "*Rediscovering Kepler's laws using Newton's gravitation law and NASA data*", American Physical Society, Joint Spring 2010 Meeting of the Texas Sections of the APS, AAPT, and SPS, March 18-20, 2010, abstract id. J3.005, March 2010