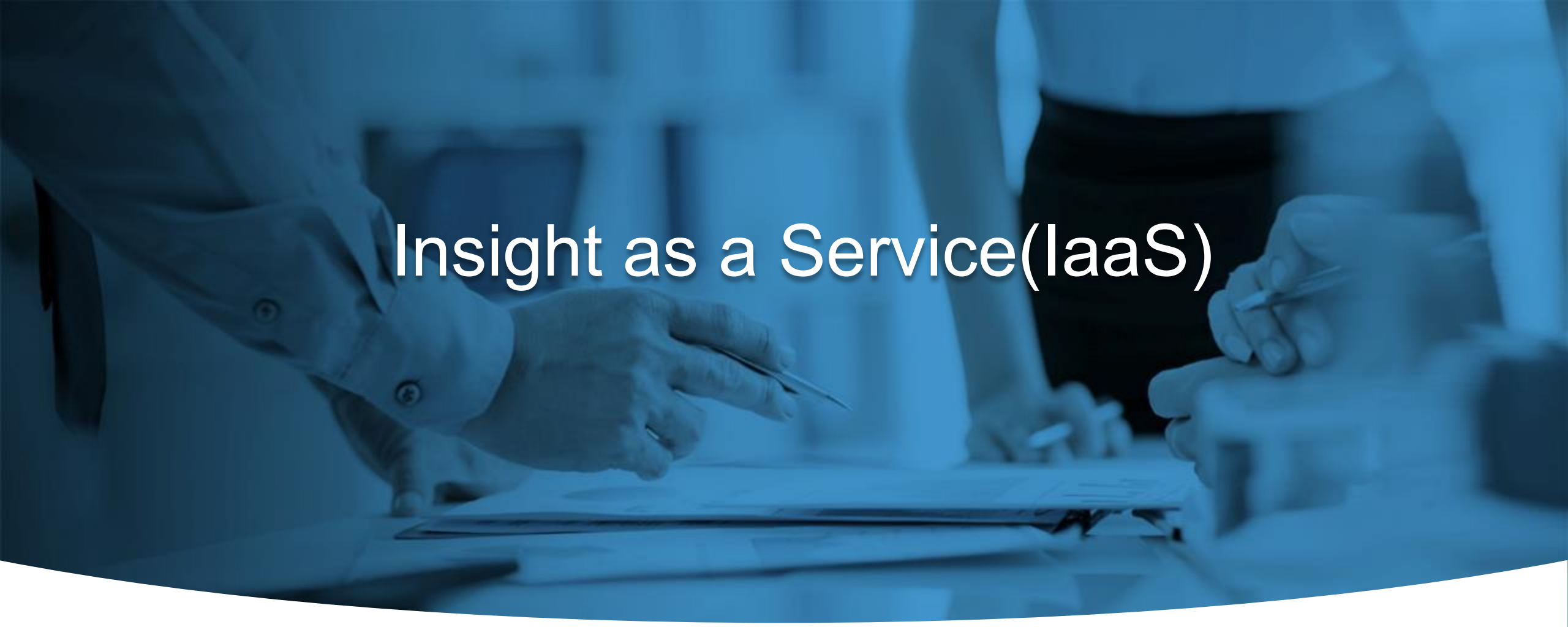


Insight as a Service(IaaS)

ASSURANT®

Objective

- The objective of IaaS project is to build a framework that works with time series data, to derive actionable piece of insights from it.

Dataset Used:

- The dataset used for the project is GEICO Dataset which is an insurance time series dataset with several KPI's like policies sold, premium earned etc. for every state of the US. The data set is of Monthly frequency from 2017-December till 2023- May.

Data Understanding

KPI's involved in the dataset

- **Average Premium Selected**
- **Cancellations**
- **Churn Rate**
- **Earned Premium Selected**
- Incurred Losses Selected
- Inforce Selected
- Loss Ratio Selected
- Policies Issued Selected
- Premium Selected
- Remorse (30 days) Selected
- Remorse (60 days) Selected
- Remorse (90 days) Selected
- Renewals Selected
- Reported Frequency Selected
- Expirations
- Growth Rate
- Net Adds
- Reinstatements
- YTD Policies Issued
- Day 366 Persist.
- Persisting Customers

Final Output

Here is the final product of the complete framework:

FL:

- An anomaly is detected for Remorse (60 days) Selected at 2023-04-30 where we obtained an increase of 22.12% compared to last month.
- Incurred Losses Selected seems to be almost constant from 2020-06-30 to 2021-01-31 but then sharply increases from 2021-06-30 to 2023-05-31.
- The Renewals Selected is forecasted to decreased by 25.73% by the next 6 month compared to today.

NJ:

- Unit Count first sharply increases from 2022-02-28 to 2022-09-30 but then sharply decreases from 2022-10-31 to 2023-05-31.

- An anomaly is detected for Remorse (60 days) Selected at 2023-04-30 where we obtained an increase of 19.51% compared to last month.
- Unit Count first sharply increases from 2022-02-28 to 2022-09-30 but then sharply decreases from 2022-10-31 to 2023-05-31.

NY:

- An anomaly is detected for Incurred Losses Selected at 2023-04-30 where we obtained an increase of 31.42% compared to last month.
- An anomaly is detected for Loss Ratio Selected at 2023-04-30 where we obtained an increase of 30.37% compared to last month.

Clusters:

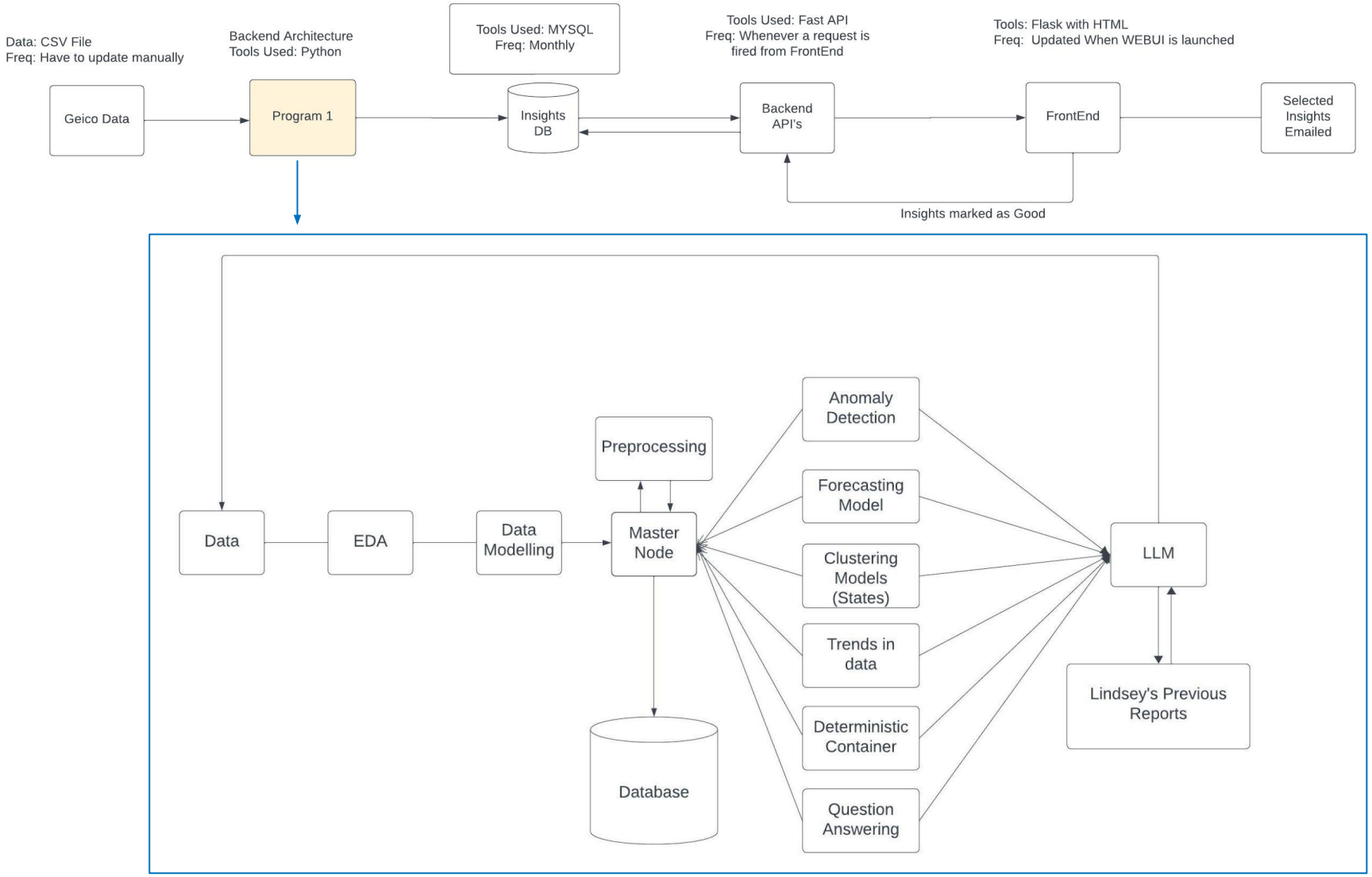
- For the last month(May), the column Average Premium Selected the high performing states are MS(504.19) AL(407.64) TN(406.04) LA(346.83) IL(318.68) amd the low performing states are ND(112.48) NJ(146.49) OR(152.97) VA(156.57) MA(157.49)

Glimpse of the dataset

- The dataset is in the form of pivot tables, I have run a script to convert it to csv dataset since pandas don't provide any explicit support for pivot tables
- Here is the link to the Original Dataset
- The dataset is converted to CSV file for preprocessing and applying different algorithms

Architecture

End to End Architecture



Architecture of Individual components

- Each module works in a sperate container individually and has a uvicorn web server on its own to handle incoming and outgoing requests. This ensures that modules can be used as plug and play into the existing architecture.
- Each docker container of the modules can be called any number of times in parallel that will ensure scalability of the architecture and the code.
- Since each module itself is a docker container we can ensure that we can add further modules without much change in the existing architecture

Algorithmic Implementations

ASSURANT®

Preprocessing

These are the following things that are implemented in the preprocessing steps:

1. Type cast the data frame into appropriate format. This involves parsing the datetime and rounding off float values to 3 decimal places.
2. Checks for DTW Distance between time series and eliminates similar time series. This will ensure that we are not processing the similar columns again and again
3. Detecting the NaN values in the dataframe and eliminating them by linear interpolation.

Master Node

I have followed a worker slave architecture to ensure that I can call any of container any number of times from the container.

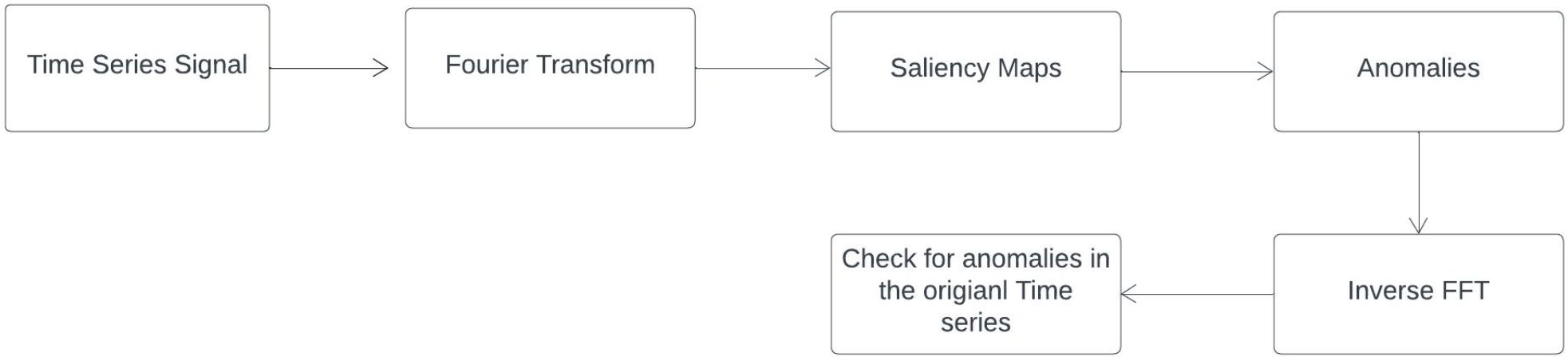
Since in the dataset you must have seen that dataset is of 50 different US States, and we need to detect anomalies, forecast, trends etc.

Introducing a master container allows us to have call each of the modules repeatedly and have more granular insights from the dataset.

Anomaly Detection

The algorithm used for detecting anomalies in the dataset is Spectral Residue Analysis.

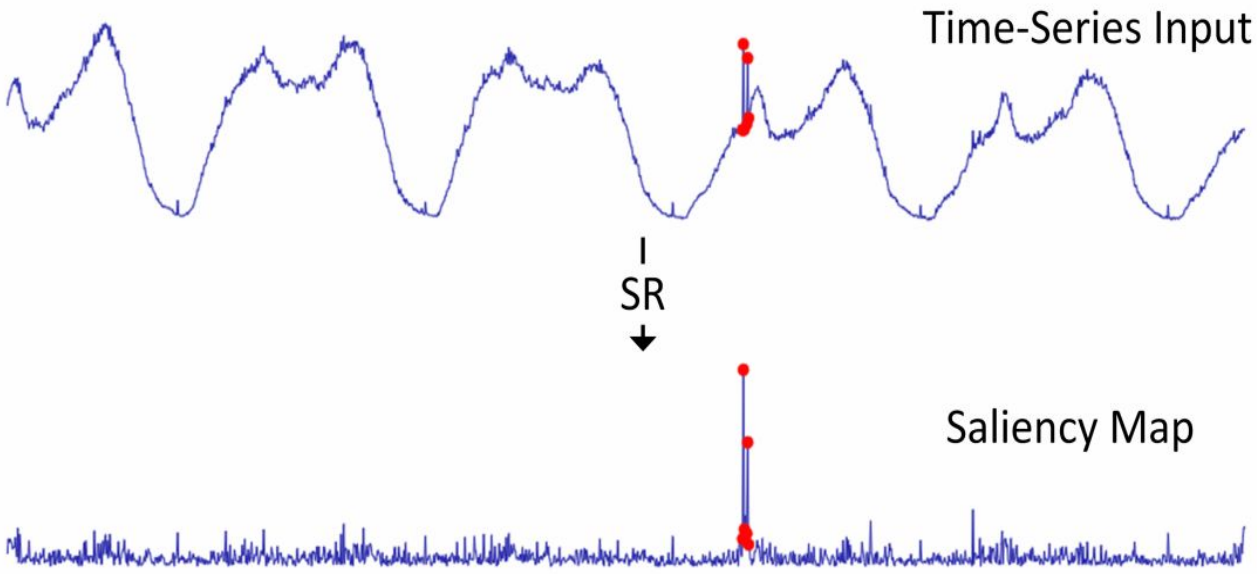
The algorithm works in the following format



Algorithmic Details:

1. $A(f) = \text{Amplitude}(F(x))$
2. $P(f) = \text{Phrase}(F(x))$ and $L(f) = \log(A(f))$
3. $AL(f) = h_q(f) \cdot L(f)$
4. $R(f) = L(f) - AL(f)$
5. $S(x) = F^{-1}(\exp(R(f) + iP(f)))$

Where F(.) is FFT, L(f) is the amplitude of the FFT, P(f) is the phase, F^{-1} is the inverse FFT, h(q) is one's matrix of size q=3, S(x) is the spectral residue. 'i' is the complex number.



De-anomalising time series

Since the rest of the modules will perform badly if the time series with anomalies are given to them, so we need to deanomalize it before performing any other functions

Least Square Method:

Algorithm Details: Since the anomalous point should behave in the similar manner corresponding to its neighbours:

1. Take a window of size k=4 on either side of the anomalous point
2. Fit Least square regression method to the points
3. Replace the value of anomaly with the predicted value

Output from Anomaly Detection

The output from the container is a data frame containing the anomalies and de-anomalized data for the time series dataset. It's a binary data frame for each of the KPI's called, where True represents that the datapoint is an anomaly

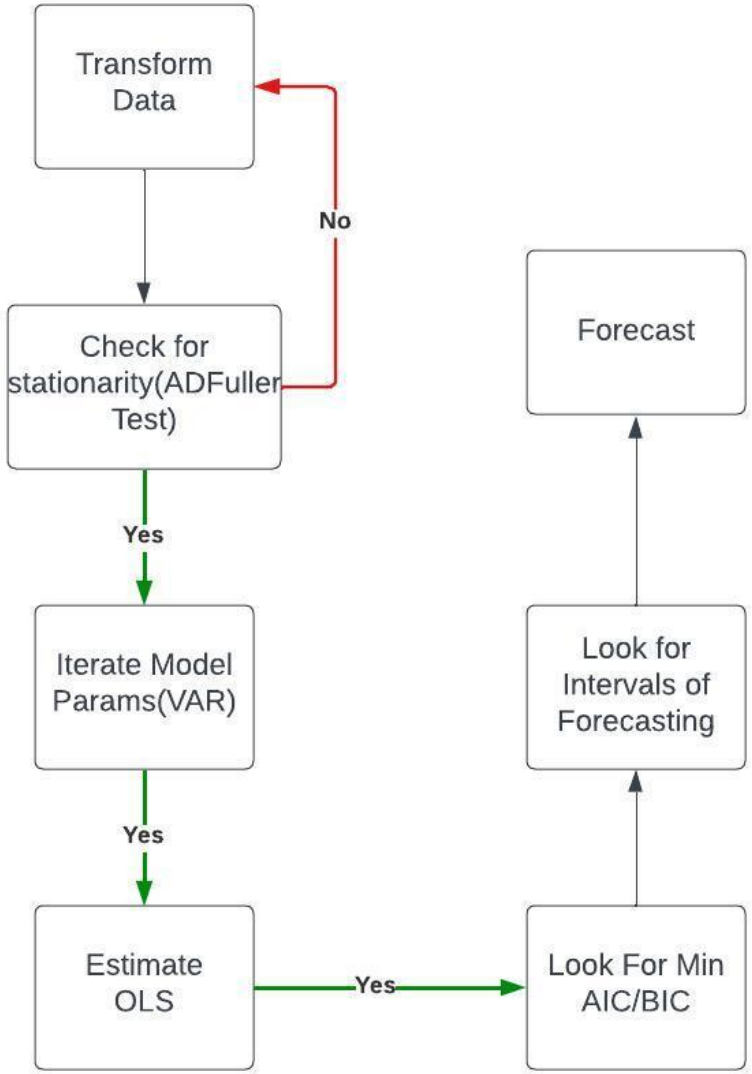
KPI-1	KPI-2	KPI-3	KPI-4	Date
False	False	False	False	2018-05-31
False	False	True	False	2018-06-30
False	False	False	False	2018-07-31

This data frame is then parsed by the master and compared to the original data frame to get the actual anomalies and the de-anomalized dataset is then parsed for further use.

Forecasting

For Forecasting I have used VAR model:

1. Since for forecasting we needs to have a stationary time series I did first order differencing until I obtained a stationary series.
2. Fit a VAR Model to the equation
3. Check for Minimum AIC to get the best model
4. I predicted the next 12 months of data for each KPI for each state using the forecasting module



The output from the container is a data frame containing the anomalies for the time series dataset. It's a binary data frame for each of the KPI's called, where True represents that the datapoint is an anomaly

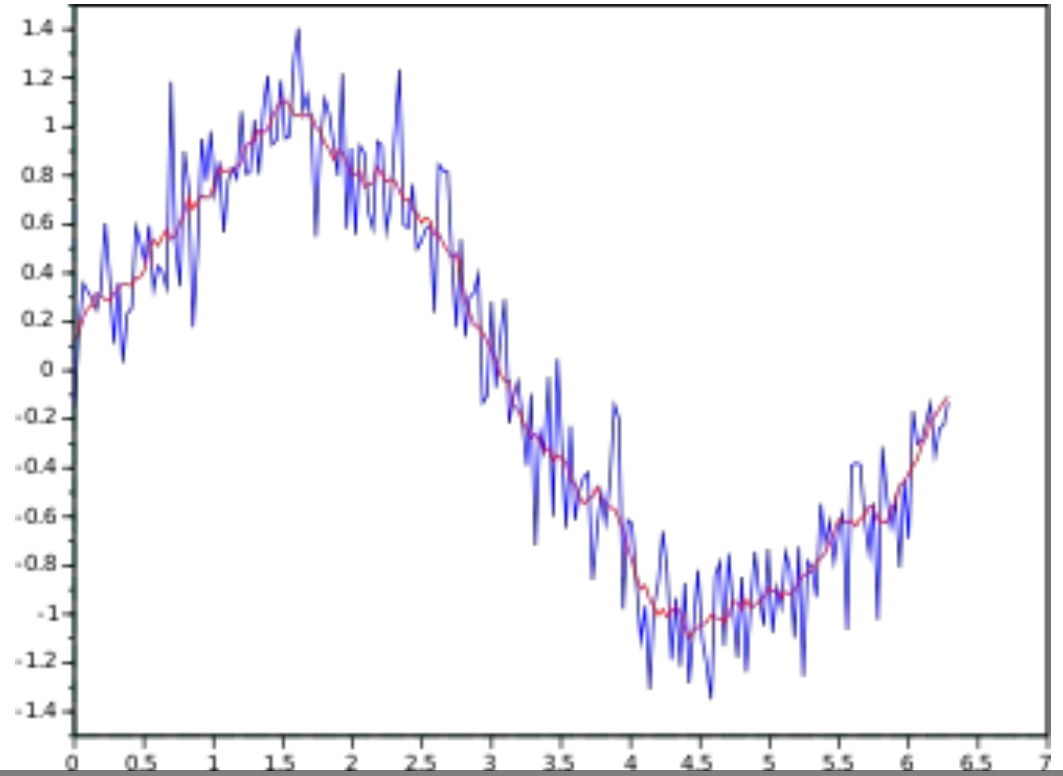
KPI-1	KPI-2	KPI-3	Months After
470.16	51	21.36	1
472.19	54	22.36	2
478.30	49	21.09	3

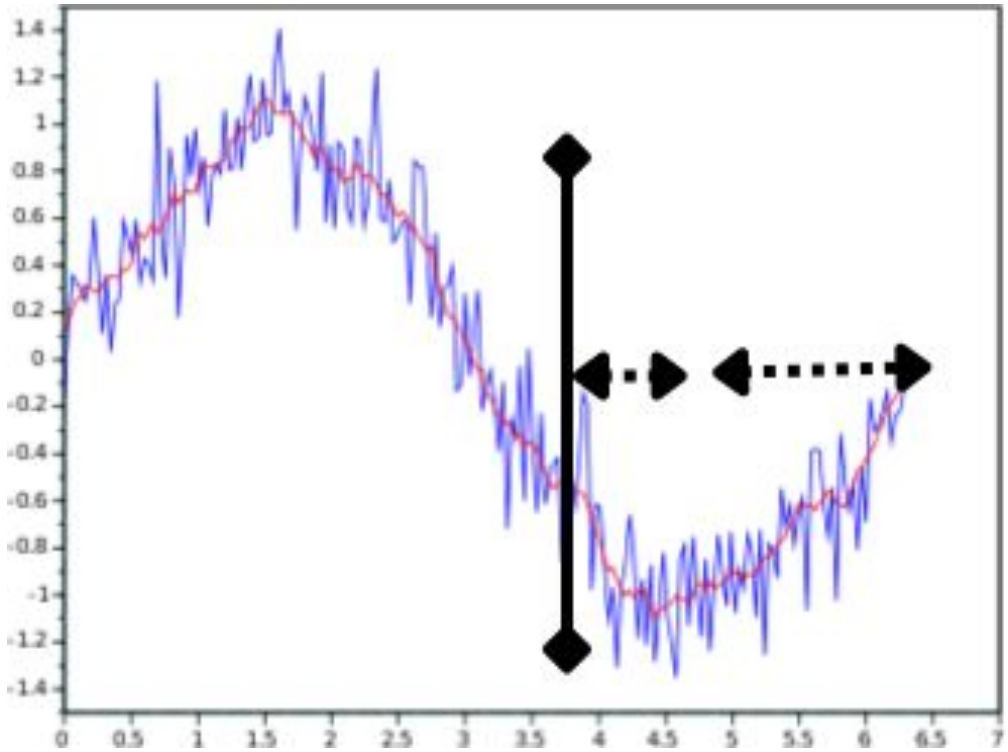
This data frame is then parsed by the master and compared to the original data frame to get the actual anomalies

Trends

- Trends in the data means the following:

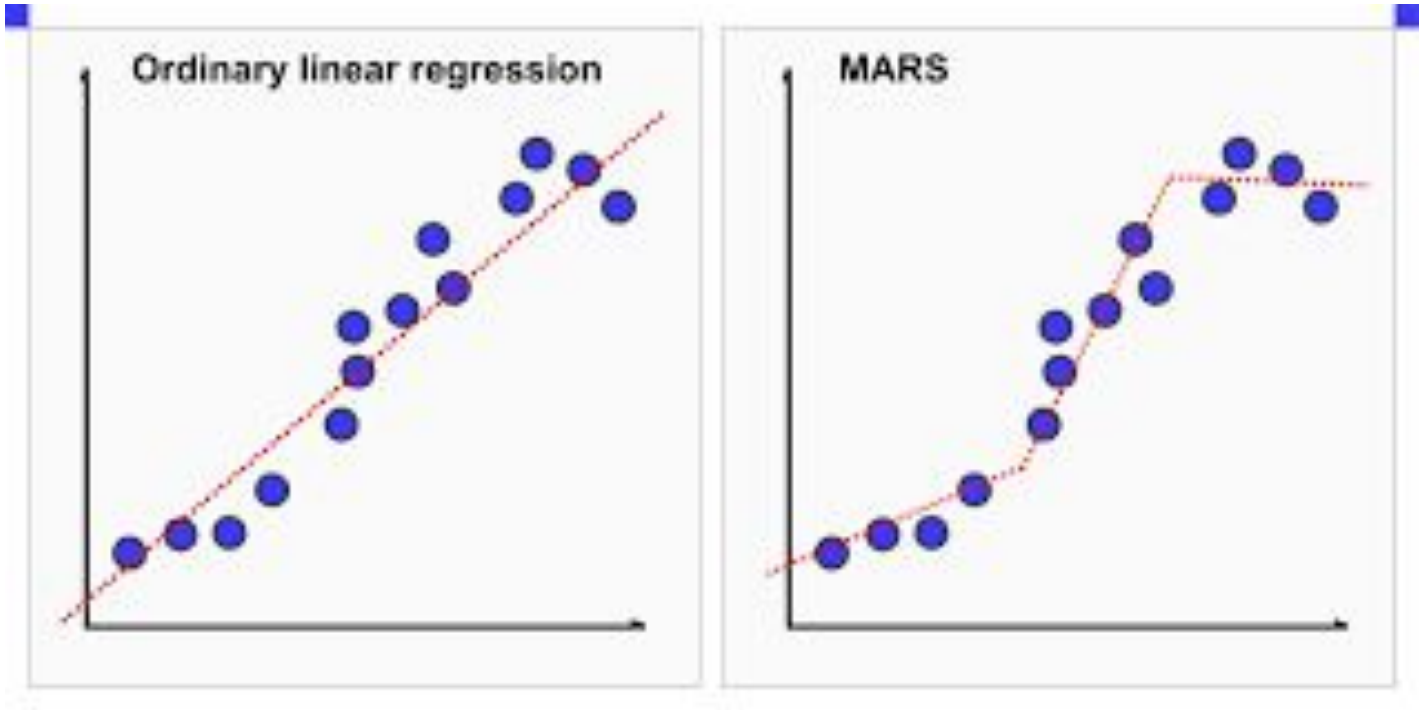
  The trend followed by the curve of any KPI:





Mars Algorithm:

Chop off the graph for the past 36 months of the data. Now draw a piecewise regression line on the given data and check its slope to check for the trends in a particular KPI for a state.



Now from the piece wise regression lines we can derive the slopes to check for trends

The output from the MARS model can be seen as a dictionary which contain the slopes of the and the values of the trends in the dataset:

Dictionary = {KPI1: [[date1, angle], [date2, angle], [date3, angle]], KPI2: [[date1, angle], [date2, angle]]}

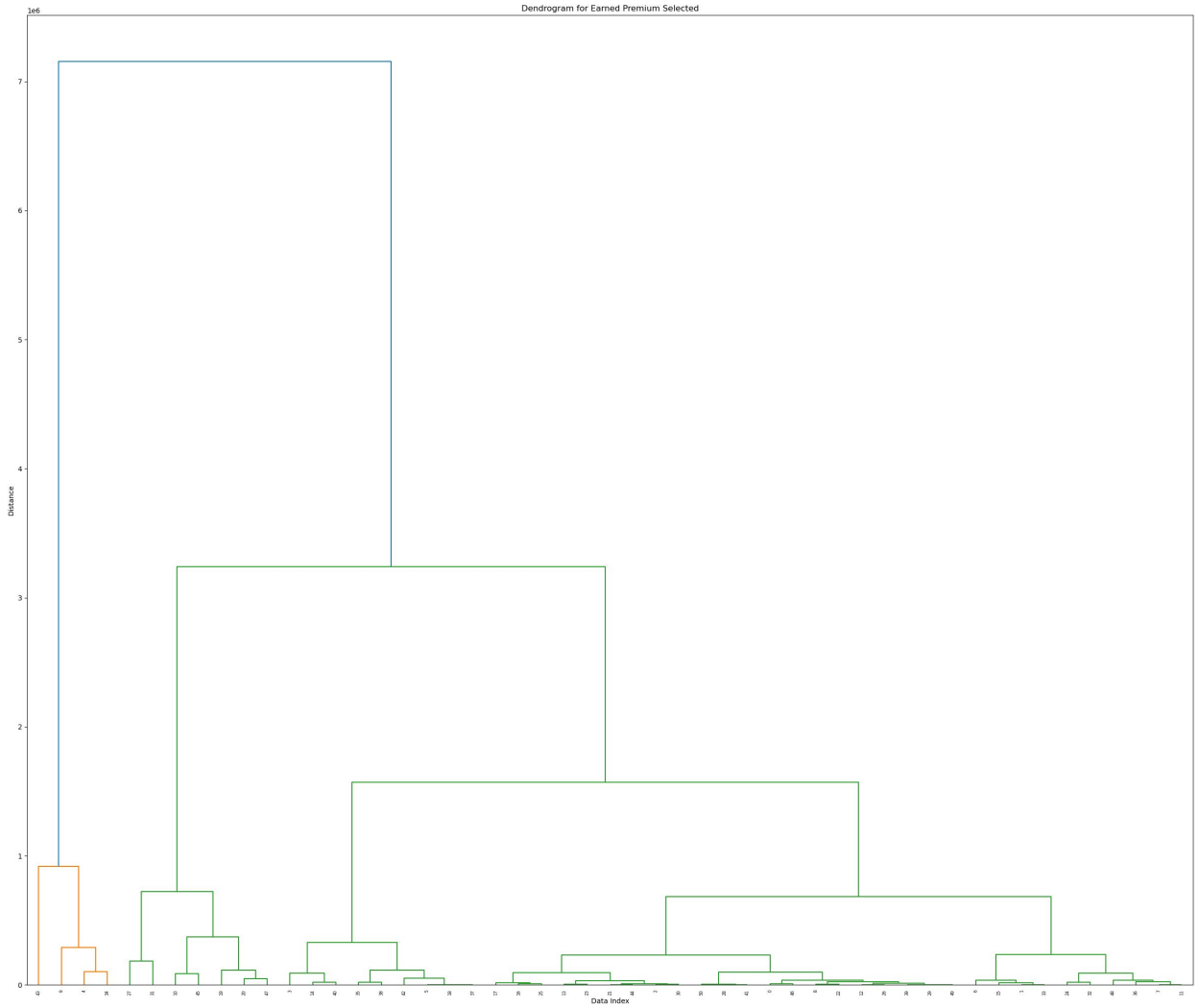
The output is then parsed and intercepted by the Master Model which then classifies the slope based on simple rules:

Angle:

Upper Angle	Lower Angle	Trend Value
-15	15	Almost constant
15	45	Gradually Increasing
-45	-15	Gradually Decreasing
45	90	Sharply Increasing
-90	-45	Sharply Decreasing

Clustering Model

The main purpose of the clustering module is to cluster similar states together based on a KPI. This will enable the business to see which states have similar kind of performance when it comes to accessing them based on an KPI. The KPI are mostly based on business rules given to me, like group similar states together that have similar values of Expiration/Inforce, or Average Premium Earned.



This will enable the business for which states work needs to done or which states are underperforming

The clustering algorithm is Hierarchical clustering, based on a KPI.

The output from the algorithm is a dictionary which looks something like:

{KPI1: {Cluster1: [States], Cluster2: [States]}, KPI2: {Cluster1: [States], Cluster2: [States]}}

The output from the model is then parsed by the master and then saved to the database

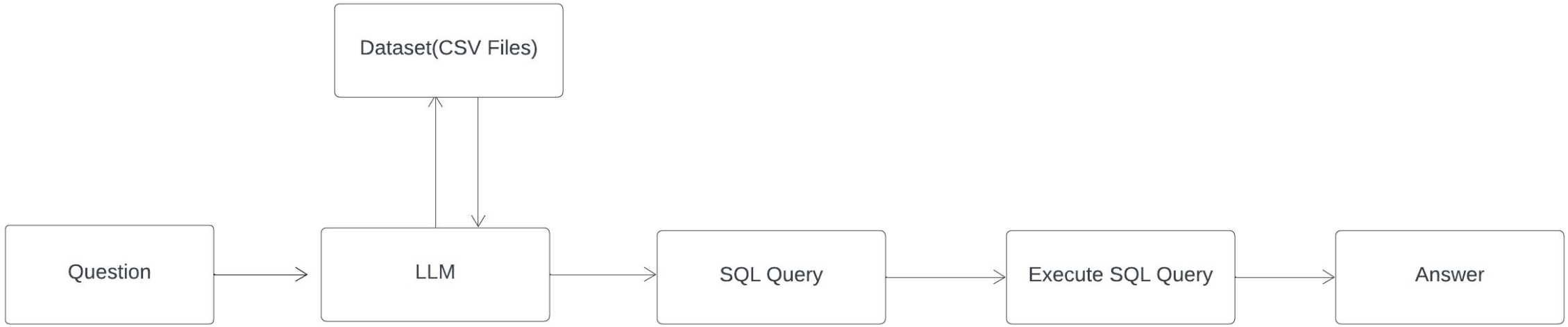
Static Model:

The module is very specific to the GEICO data. Since Lindsey have some comparison metrics in her report like YOY Change or MOM change, the module will enable us to calculate those metrics automatically and present at the webUI, saving manual effort.

The insights return will be in the form of dictionary like
{KPI1: {YOY_Change: value, MOM_Change: value}, KPI2: {{YOY_Change: value, MOM_Change: value}}}

Question Answering Module:

The main purpose of this modules is to enable the user to ask question based on the data we have. The workflow for the entire structure works like:



Different LLM's tried and tested

- I have tried multiple open-source LLM's for generating question from Lindsey's report and generating a SQL Query for the same:

Model Name	Parameter Size	Quantized	GPU-Size Required	Time Required for same query	Question Generation	SQL Query Generation
Stable Beluga	7B	No	21GB	4-5 sec	✓	❌
Stable Beluga	13B	Yes	24GB	8-9sec	✅	❌
Nous-Hermes-Llama2-13b	13B	Yes	24GB	8-9 sec	✅	❌
GPT 3.5 Turbo	175B	-	-	3-7 sec	✅	✅

SQL Query Template

Given an input question, first create a syntactically correct {dialect} query to run, then look at the results of the query and return the answer.

Use the following format:

Question: "Question here"

SQLQuery: "SQL Query to run"

SQLResult: "Result of the SQLQuery"

Answer: "Final answer here"

Only use the following tables: {table_info}

If someone asks for the table foobar, they really mean the employee table.

Question: {input}

Database

SQL Database Schema

Each container is going to run and store its data into a separate table whose schema is as follows:

Schema: https://drawsql.app/teams/aman-vishnoi/diagrams/aman

Backend Architecture

The backend architecture exposes some endpoint:

1. /get_insights: Get the top insights by using the three metrics:

a. By Inforce

b. By Distribution(select only those which are more important)

c. Most recent data This will return 30 insights based on the following metrics

2. /get_metrics_by_filters: This will return the insights based on the metrics and the above two. Only return 30 insights at max.

PPM Unified

Objective

- The objective of PPM Unified is to build ensemble of machine learning models that can accurately forecast the prices of IPad’s from 60 months of their launch

Dataset Used:

- The dataset used in the project is the PPM’s Data
- RNN+LSTM and ensemble of ARIMA model is used for the same purpose

Dataset Preprocessing(Imp):

- Missing months for a device is interpolated by Linear Interpolation
- Devices with less than 3 data points are dropped

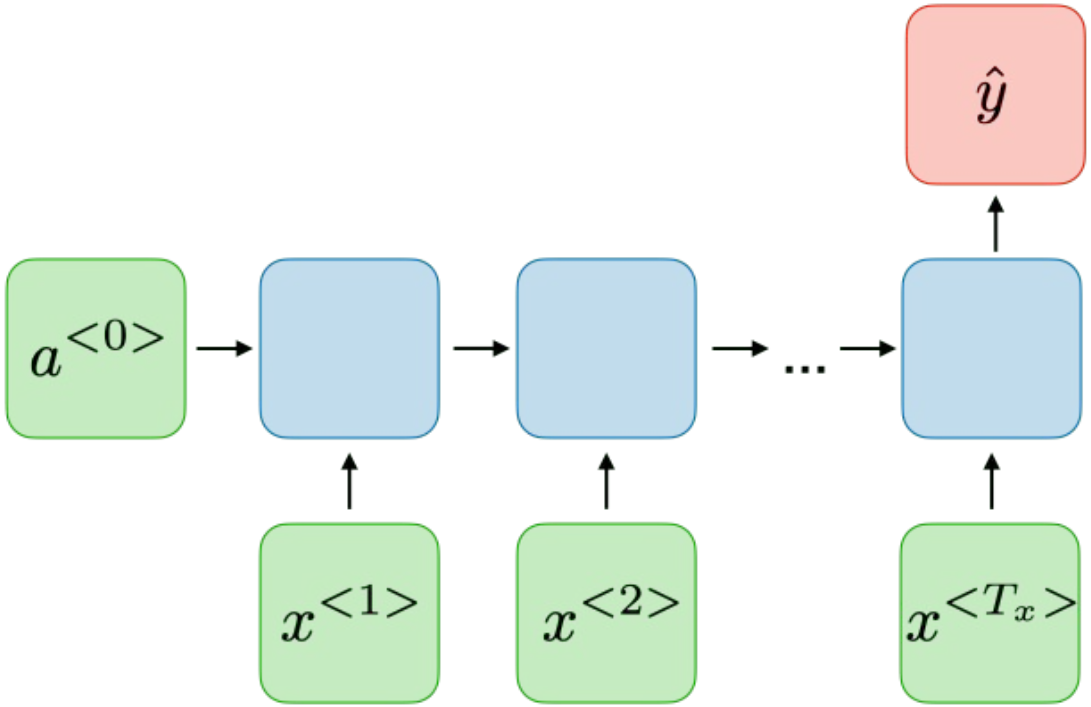
Features in the dataset

- Dataset typically consists features of particular device and its historical price.
  - Feature in the dataset:
    - Screen Size
    - CPU
    - Protocol
    - Generation
    - Carrier
    - Camera
    - **Historical Price**
    - Ports
    - Manufacturing Price
    - Battery
    - Capacity(RAM)

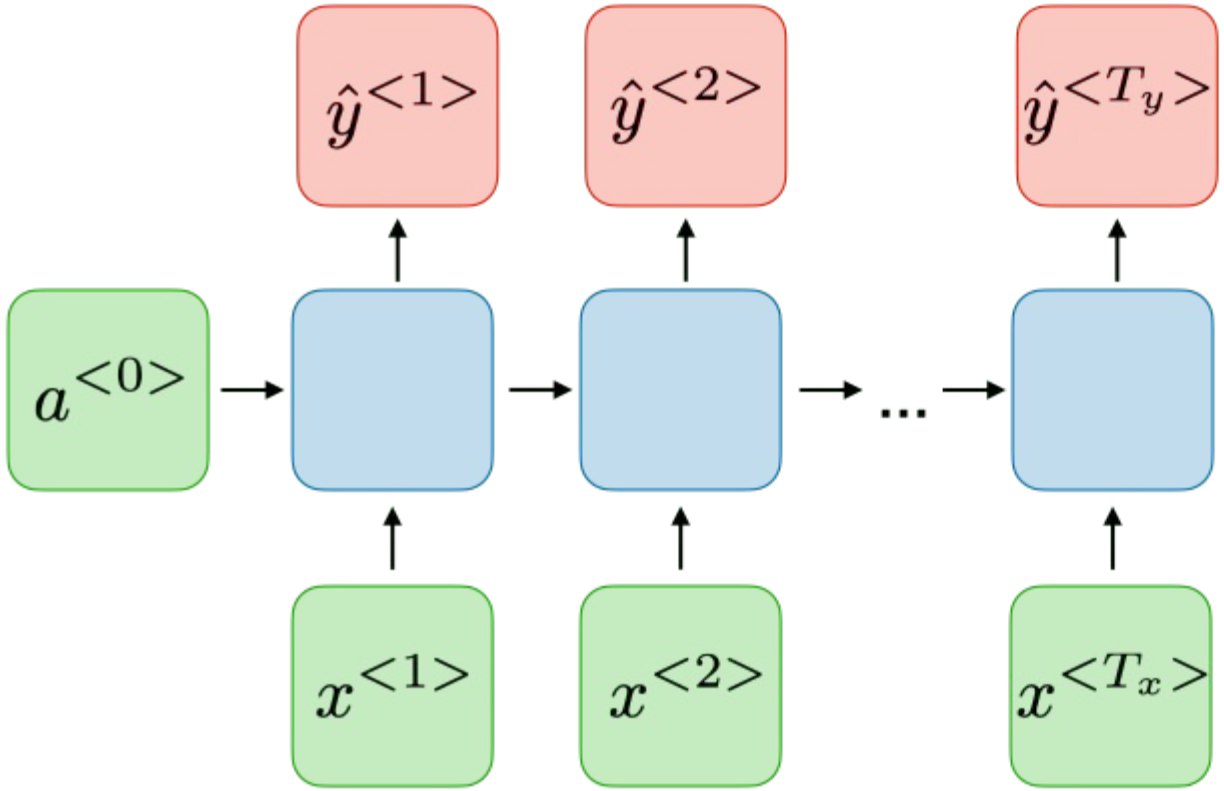
The entire data set is converted to numpy arrays to be feeded into the model, with a shape of (num_models, num_months, features)

Model Tried and Tested

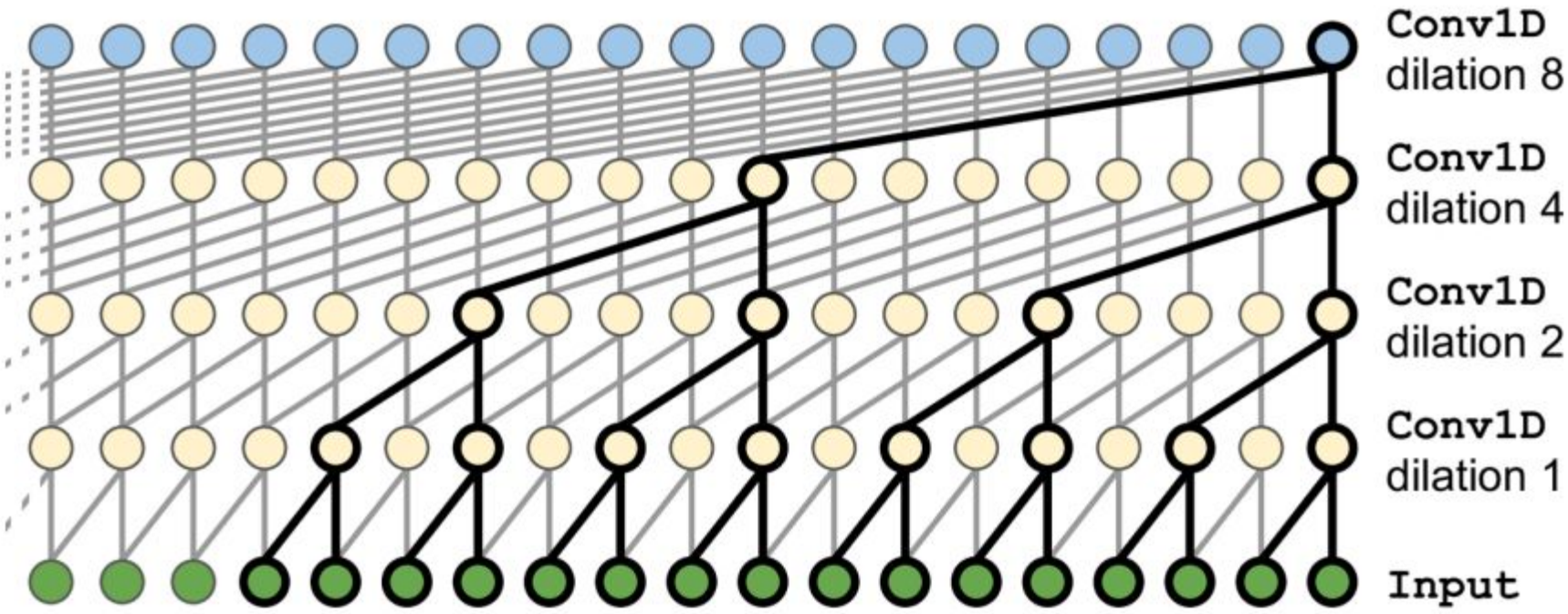
a). Many to One RNN and LSTM



b). Many to Many RNN and LSTM



c). WaveNet Architecture



Final Architecture decided:

Final Architecture is decided by using hyperparameter optimization using Keras Hyberband Algorithm.

Confidence Interval Generation

The uncertainty in the predictions can be because of two factors:

a. The uncertainty in the variables used in the predictions itself
b. The uncertainty because of human tendency to sell the mobile phones

Confidence intervals are generated by bootstrapping technique on the residuals generated for a particular iPad's device