# Malaria Commodity Demand Prediction Model

## Group I

## Using Time Analysis and Extreme Gradient Boosting

Theophilus Lincoln Owiti, Mercy Mburu, Ian Kiplangat, Teresa Chebet,
Pleasant View Jr.

# Business Logic

Tracking and restocking of Malaria commodities is currently done on the basis of 6-month averages computed by MoH officials from the KHIS system data. The averages utilized are not efficient in projecting a clear and desired picture of trends related to consumption such as seasonality, increases or decreases as well as other confounding factors.

KHIS through a representative shares problems that they have faced in trying to use the averages in restocking Malaria commodities. The representative showed a system based of KHIS and desired the same predictive outcome instead of the averages projected to create and efficient way of making decisions to restock Malaria commodities amidst demand.

**Need of client**

Therefore, KHIS needs a Time Analysis Predictive model that is able to monitor common patterns and trends and give our projections of what is expected as per areas of low demand and areas of high demand in order to have efficient restocking of Malaria commodities.

# Data Acquisition

Data is derived from the KHIS System that generates 6-month seasonal averages. The team asked for data from the KHIS System where it was then provided. The nature of the data received was in the form of a CSV file that pertained of the years between 2018 to 2022, each of these years with their monthly demand for each county in Kenya. The data set projected a period of about 5 years.

| periodid | periodnar | periodcod | perioddes | dataid | dataname | datacode | datadescr | Baringo C | Bomet Co | Bungoma | Busia C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-01 | Jan-22 | 202201 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2644 | 626 | 24500 | 464 |
| 2022-02 | Feb-22 | 202202 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3529 | 444 | 19890 | 325 |
| 2022-03 | Mar-22 | 202203 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2998 | 460 | 21699 | 354 |
| 2022-04 | Apr-22 | 202204 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2755 | 508 | 21538 | 317 |
| 2022-05 | May-22 | 202205 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2878 | 355 | 29388 | 486 |
| 2022-06 | Jun-22 | 202206 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3433 | 938 | 48301 | 668 |
| 2022-07 | Jul-22 | 202207 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2286 | 154 | 30519 | 582 |
| 2022-08 | Aug-22 | 202208 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2908 | 198 | 22585 | 396 |
| 2022-09 | Sep-22 | 202209 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 5001 | 183 | 20930 | 317 |
| 2022-10 | Oct-22 | 202210 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3934 | 135 | 20739 | 297 |
| 2022-11 | Nov-22 | 202211 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 2774 | 104 | 21332 | 354 |
| 2022-12 | Dec-22 | 202212 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3300 | 91 | 18250 | 287 |
| 2021-01 | Jan-21 | 202101 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3176 | 548 | 15612 | 48 |
| 2021-02 | Feb-21 | 202102 | Hot | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 10559 | 1222 | 24803 | 135 |
| 2021-03 | Mar-21 | 202103 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 4618 | 2002 | 21181 | 516 |
| 2021-04 | Apr-21 | 202104 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3911 | 1634 | 19911 | 360 |
| 2021-05 | May-21 | 202105 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3923 | 1050 | 30332 | 558 |
| 2021-06 | Jun-21 | 202106 | Wet | cPlWFYbB | MOH 743 Rev2020_Rapid Diagn | | | 3858 | 1265 | 42697 | 732 |

data

Figure 1.0: Table data from KHIS System

# Data Exploration & Analysis

In efforts to understand the big picture behind the demand for the above-mentioned commodity demand in Kenya. Power BI was utilized to visualize the overall occurrence of demands.

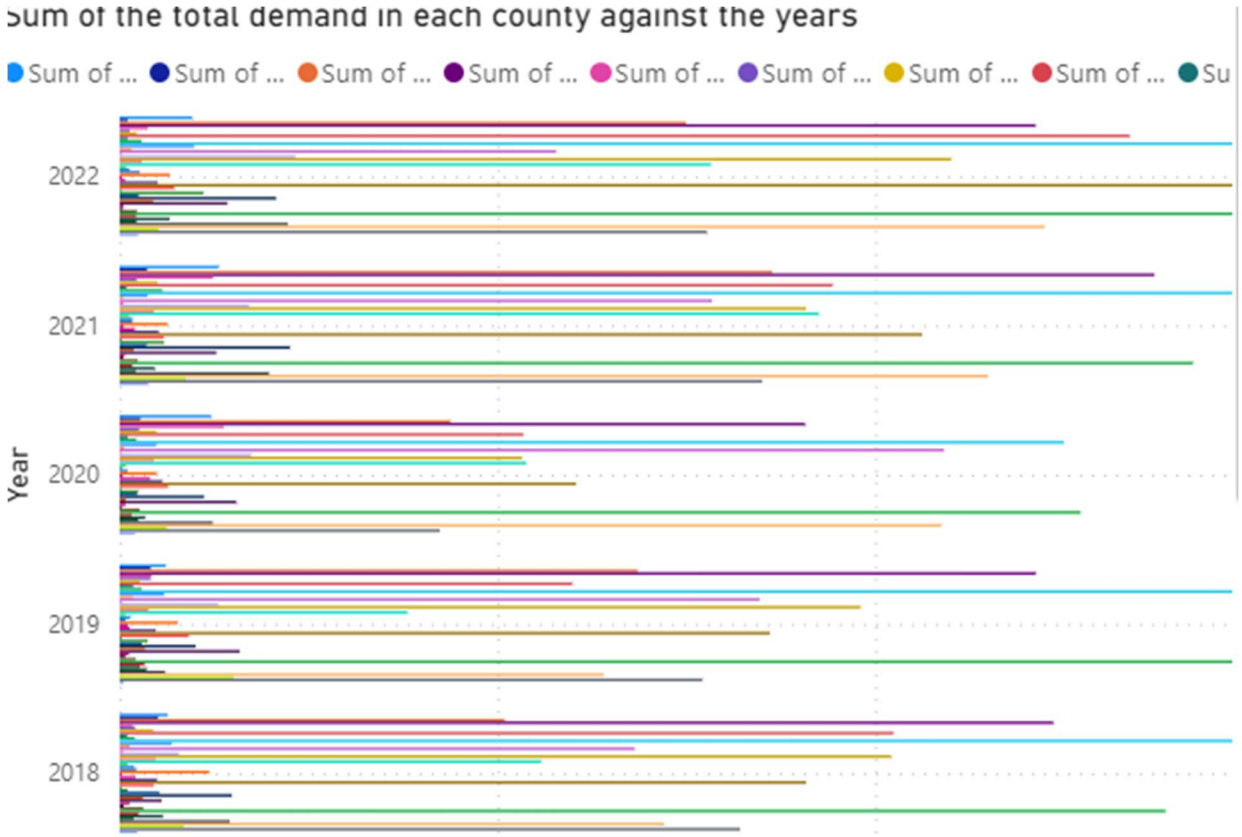



Figure 2.0: Power BI Generation of Bar Graphs for the whole Country

Model counties were appropriate to be chosen and Nakuru was a county of interest. The question that revolves around one's mind is that what was the current trend of Malaria for about 5 years in Nakuru?

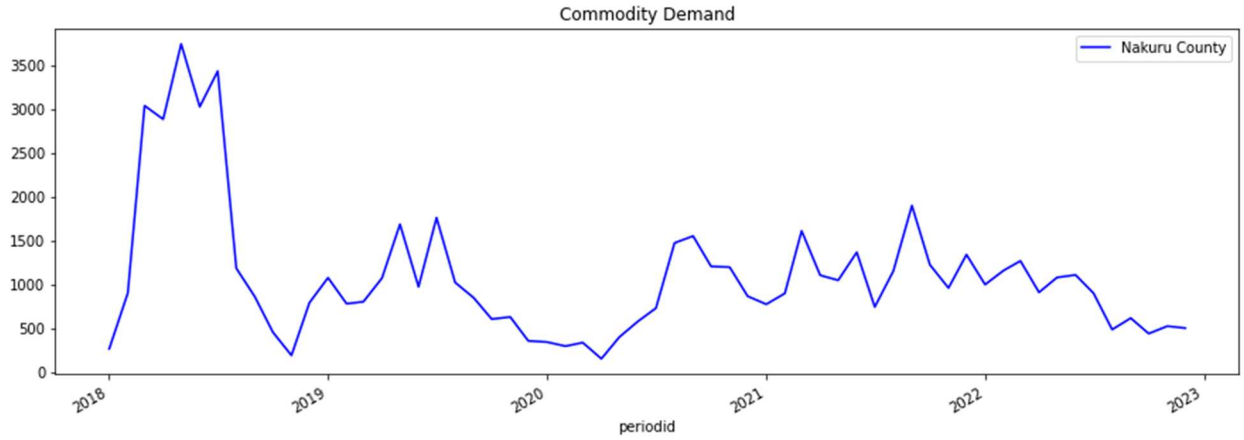



Figure 2.1: Nakuru Time Series Plot Generated by Python Pandas

From the data trend we observe that:

- Between 2018 – 2019, there was a sharp rise of Malaria in Nakuru County and the middle of 2018 there was the highest spike. This conclusion is based on the fact that more Malaria commodities were on demand. The outbreak of Malaria in Nakuru at this time might have been due to most people not sleeping under a mosquito net and proper drainage to discourage breeding of mosquitoes were not destroyed as required.
- Approximately 4 months towards the end of the year there was a sharp decline in demand and then a steady fall and in the last 2 months a steady increase occurred but lower than in the middle of the year.
- Same trend followed in 2019 but with lower Malaria commodity demand. This means that if the government was holding awareness campaigns on Malaria it means that the campaigns had started taking shape in sensitizing people against Malaria.
- Up to 2022 a prominent trend of spikes in the middle of the year occurs but at lower rates than in 2018.
- The seasonal trend becomes spikes in the middle of the year and this can be attributed to the fact that spikes occur due to rainy period that encourage breeding of mosquitoes due to stagnant water that is usually present at this time.
- Also, the spikes are rampant towards June/July. But commodity demand has lowered over the years.

Fig 2.2: Trend in 2018 in Malaria commodity demand

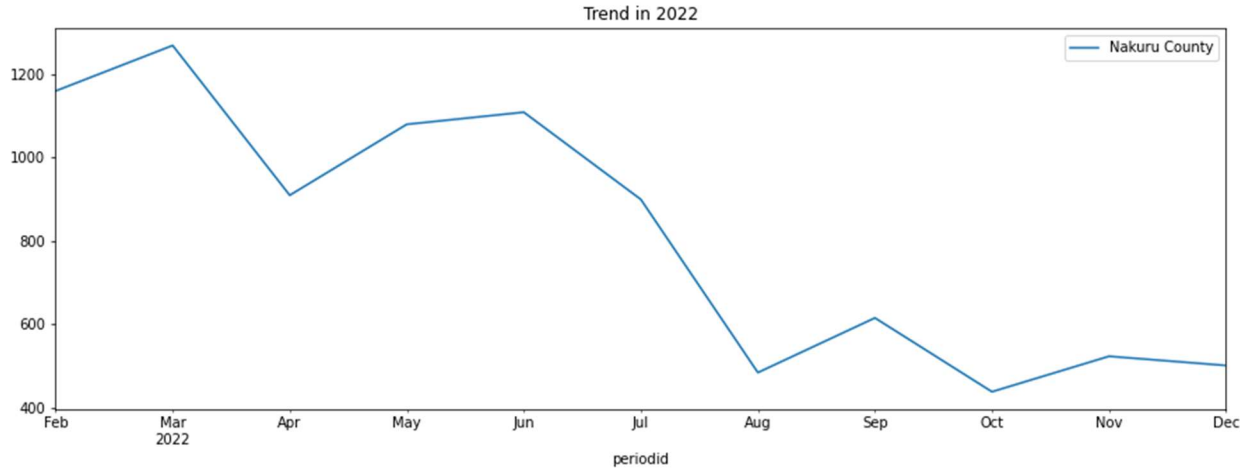
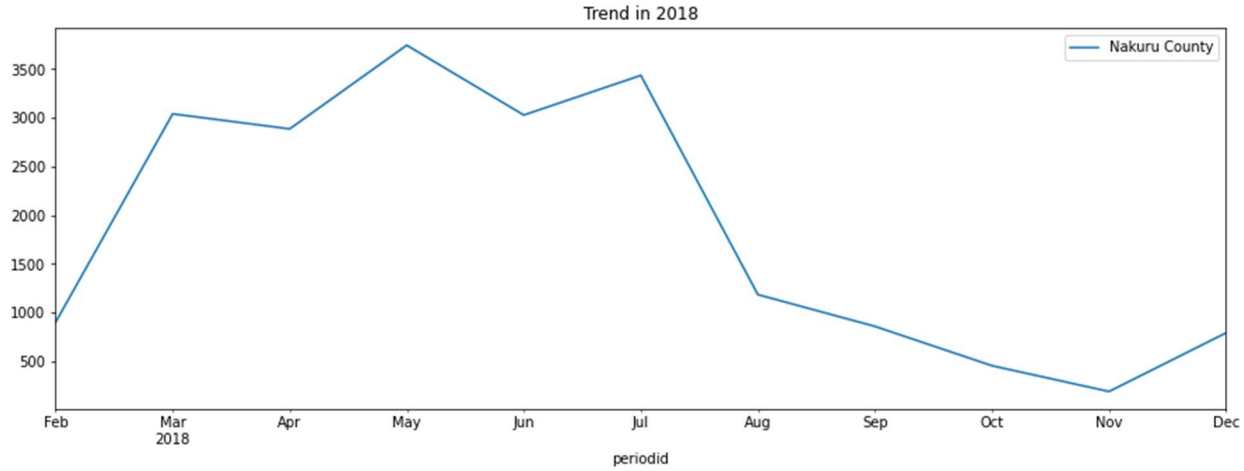



Fig 2.3: Trend in 2022 in Malaria commodity demand



From the generation of features after doing out time analysis. An interesting fact comes out in terms of trends that occur in each month in Nakuru per month.

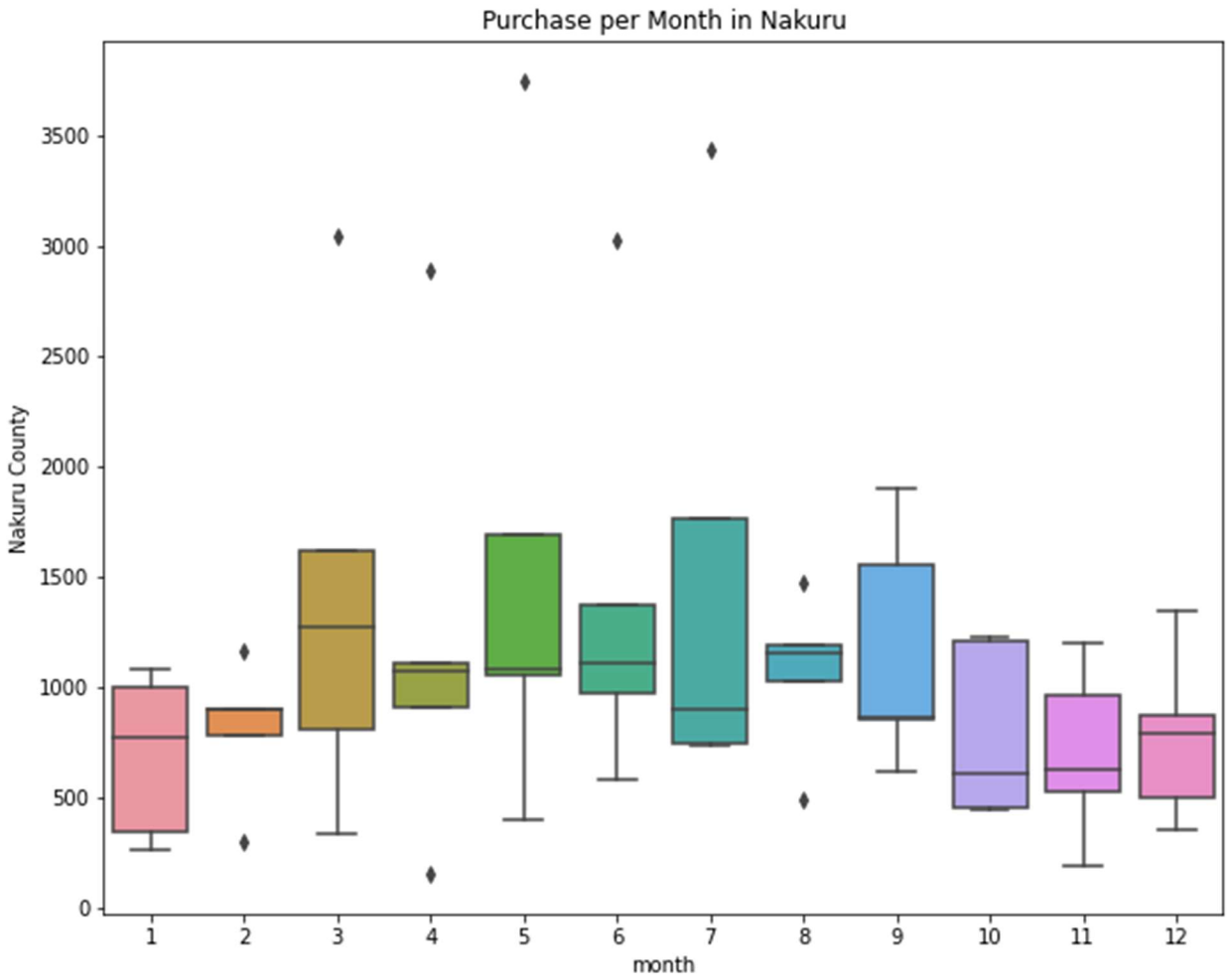



Fig 2.4: Box Plot for purchase per month in Nakuru

As we have determined it is March or May that demand gets up this can be due to the wet conditions that occur at that time and following the box plot it shows between July to October there is still a fairly significant rise in demand.

It therefore important that most restocking should be done starting early as March to avoid havoc during high spikes. It is important to note that the spikes are not uniform but the largest effect starts at around March.

# Data Cleaning

After doing a quick analysis of the data, during cleaning the CSV file:

- We use pandas to check for null values using the **isnull()** function.
- We then use **dropna()** to drop the null values that are present in our data.
- If our values are needed we fill them in with averages.

All this is possible through the pandas library.

```
dataframe=pd.read_csv('data.csv',usecols=['periodid','Nakuru County']).dropna()
#retieve 4 rows
dataframe.head(4)
```

| | periodid | Nakuru County |
|---|---|---|
| 0 | 2022-01 | 998 |
| 1 | 2022-02 | 1159 |
| 2 | 2022-03 | 1268 |
| 3 | 2022-04 | 909 |

Figure 3.0: Dropping null values using pandas dropna() function

# Feature Engineering

It is first of all important to do a time analysis where we have informed decision on which features/ variables we want to use. Then we use pandas to index our data according to the datetime after changing the first column- “periodid” to datetime datatype. Before this we can do a small test in our notebook to test for stationarity of our data.

```
dataframe['periodid']=pd.to_datetime(dataframe['periodid'])
dset_indexed=dataframe.set_index(['periodid'])
dset_indexed.head(4)
```

| periodid | Nakuru County |
|---|---|
| 2022-01-01 | 998 |
| 2022-02-01 | 1159 |
| 2022-03-01 | 1268 |
| 2022-04-01 | 909 |

Figure 4.0: Indexing our data according to the datetime column.

In a trial Python notebook, an Analysis of the data is done to find out if the trend present has an aspect of stationarity. We use the rolling mean and rolling standard deviation to vaguely determine if our data is stationary from a graph. To see if the trends that we have are consistent throughout the seasons we have. In this instance we use Python’s Statmodel.

From statsmodel we import dickey fuller test and use it to see the metrics and determine the nature of our data and give us our p-values of the data and tells us about our stationarity condition.

In the official notebook we then do a split for the test and train data. At this point it is wise not to use random split as we are dealing with a time series therefore we will use a range from the most recent years from 2021 as our test data set and early years as out train dataset. At this point we will split it using pandas **loc()** method.

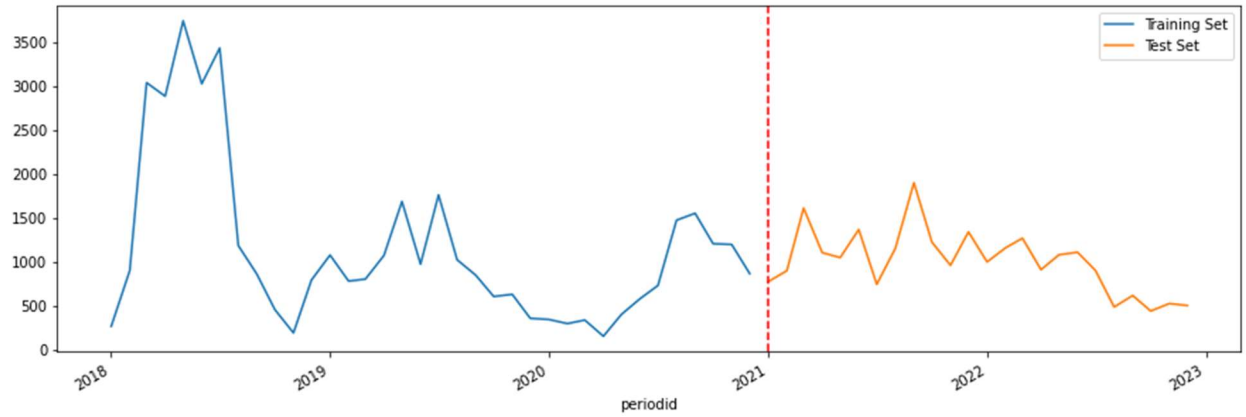



Figure 4.1: Graphical depiction of how we split our data.

We then go ahead and generate our features; we would now take from the indexed data the year and of course the months to determine seasonality.

```
def feature_generator(dF):
    dF=dF.copy()
    dF['year']=dF.index.year#year
    dF['month']=dF.index.month #we get month from our datetime index

    return dF


df=feature_generator(dset_indexed)
df.head()
```

| periodid | Nakuru County | year | month |
|---|---|---|---|
| 2022-01-01 | 998 | 2022 | 1 |

Figure 4.2: Getting our features as year and month using user defines function feature_generator()

# Model Development

Several models were used to test for proper prediction based on the new features we have adopted from our indexed dataset. The Machine Learning models of choice were: Linear Regression with TensorFlow Neural Networks using Sequential Layers, Gradient Boost Regressor and lastly Extreme Gradient Boosting.

Linear Regression resulted into large MSE Error and did not fit any of the curves in an optimal way and performed immensely poor during regularization with new data. Gradient Boost Regressor resulted into continuous underfitting and large RMSE and MSE Errors and had poor regularization that projected impossible values even after adding an appropriate learning rate and proper number of boosted trees.

Extreme Gradient Boosting resulted into acceptable regularized values that gave an error of about RMSE of 500 and an observation of overfitting was observed at the point of cross-validation data set. At this point a number of trees were reduced and an appropriate learning rate was adopted for proper learning. After a doing all checks to make the algorithm work better it was therefore concluded it fit the curve as proposed.

The Algorithm was derived from xgboost library in python and incorporated as a function with relevant parameters for optimal performance.

```
#model
reg=xgb.XGBRegressor(n_estimators=1000, early_stopping_rounds=50)
reg.fit(X_train,y_train,eval_set=[(X_train,y_train),(X_test,y_test)],verbose=100)
```

Python

```
[0]    validation_0-rmse:1127.72919    validation_1-rmse:882.03079
[55]   validation_0-rmse:0.86921       validation_1-rmse:678.80378
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=50,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=1000, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

Figure 5.0: XGBRegressor model

# Model Evaluation

We first of all see the importance of the features that we used and the pandas provides this functionality as shown in Figure 6.0.

```
fimportance=pd.DataFrame(data=reg.feature_importances_,index=reg.feature_names_in_, columns=['Importance'])
```

Python

This terms quater prediction as an inaccurate strategy for prediction of commodities. Meanwhile, the month features makes it to the spot light as of the highest importance.

```
fimportance.sort_values('Importance').plot(kind='barh',title='Feature Importance')
plt.show()
```

Python

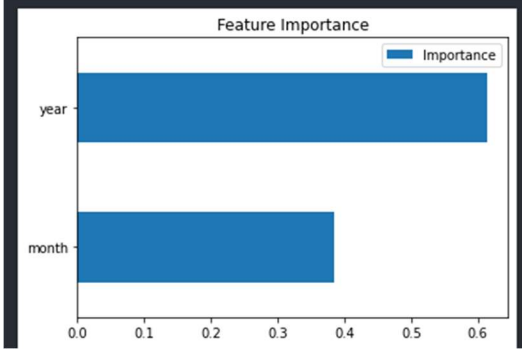



Figure 6.1: Feature importance

In evaluation we use **RMSE** as our main metric where we end up getting a result of 641.69 that was extremely high for other algorithms.

## Evaluating and Getting the performance

We will use either RMSE or MSE

```
error=np.sqrt(mean_squared_error(test['Nakuru County'],test['prediction']))
print(f'Root Mean Squared Root: {error:.2f}')
```

```
Root Mean Squared Root: 641.69
```

Figure 6.2: RMSE

We use this model to predict the test dataset and the graph shows how the prediction does. The prediction is not as effective as expected due to the data barriers faced and distribution. The major factors that affect the graph are the amount of data and the distribution of our data.

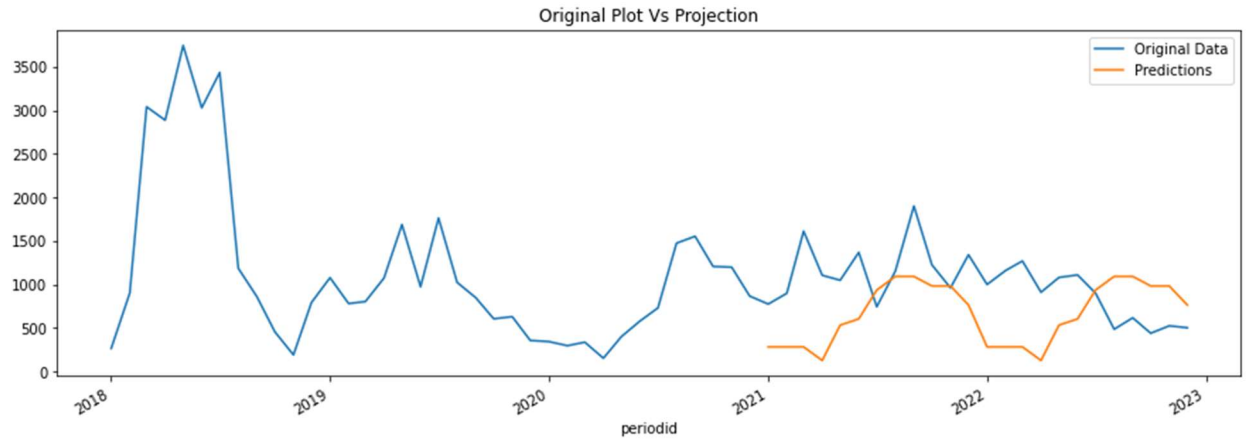



Figure 6.3: Test data prediction on graph.

Summary:

The metrics that was used for our model was the Root Mean Squared Error (RMSE) and Mean Squared Error (MSE), important thing to note is that the Regressor uses RMSE only by default because of the inconsistencies.

In accordance with the features, this terms quarter prediction as an inaccurate strategy for prediction of commodities. Meanwhile, the month features make it to the spot light as the highest importance.

We used a pandas feature that allows us to compute the importance of the features that we are to use for plotting.

XGBR is a Regression Model that works by building ensemble of decision trees, where each tree is trained to make prediction based on a subset of the available data. The final prediction is made by taking the average of the prediction from all the trees in the ensemble.

It was the used model as it worked better as compared to the others and the over-fitting was easier to identify using the cross-validation function.

# Model Deployment

The main strategy for deploying the model was to create a python module and use an local deployment then later on implement a flask deployment using proper routes to do county predictions and give out desired prediction that are appropriate for restocking.

Later recommendation is to use Azure Data Pipelines and create a model endpoint that can interact with an application API in order to give proper projections.

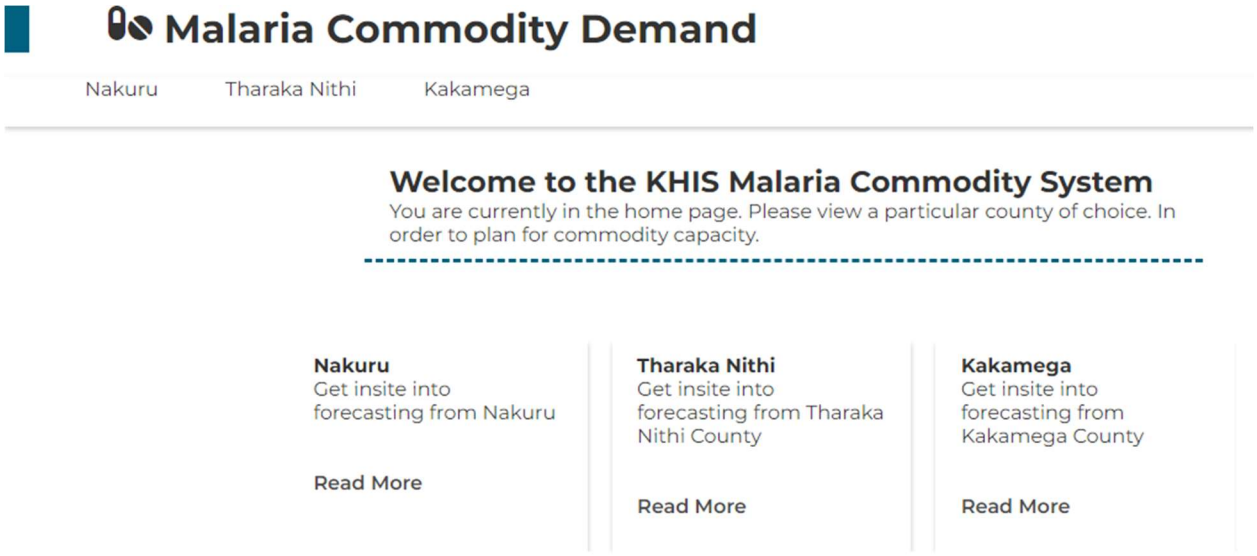



Figure 7.0: Deployment Homepage

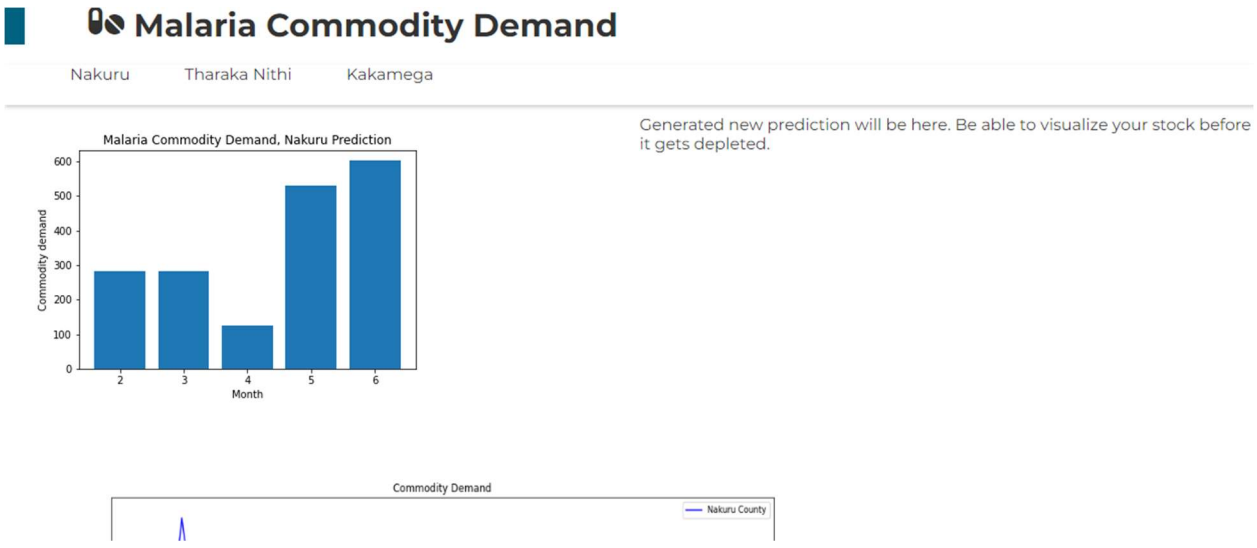



Figure 7.1: Nakuru Prediction page on the demands that are expected in months of 2023

# Challenges & Recommendation

- There was insufficient data leading to inaccurate predictions.
- There was a challenge in time where the Deployment phase needed a lot of testing.
- Insufficient time to properly understand the data since the data is the core of the model.