Kerala Road Accident Analysis And Forecasting

Sathyajith KP
Naheela PK

01

What is Time Series Forecasting?

Time series Analysis & Forecasting

A time series is a sequence of observations taken at successive equally spaced points in time. A series of data points indexed (or listed or graphed) in time order.

Time Series Analysis and Forecasting is the process of understanding and exploring **Time Series** data to predict or forecast values for any given time interval. It predicts future events by analyzing the trends of the past.

- Time series analysis can be useful to see how a given asset, security, or economic variable changes over time.
- Time Series Forecasting relates to trend analysis, cyclical fluctuation analysis, and issues of seasonality.

As with all forecasting methods, success is not guaranteed.

02

DATASET

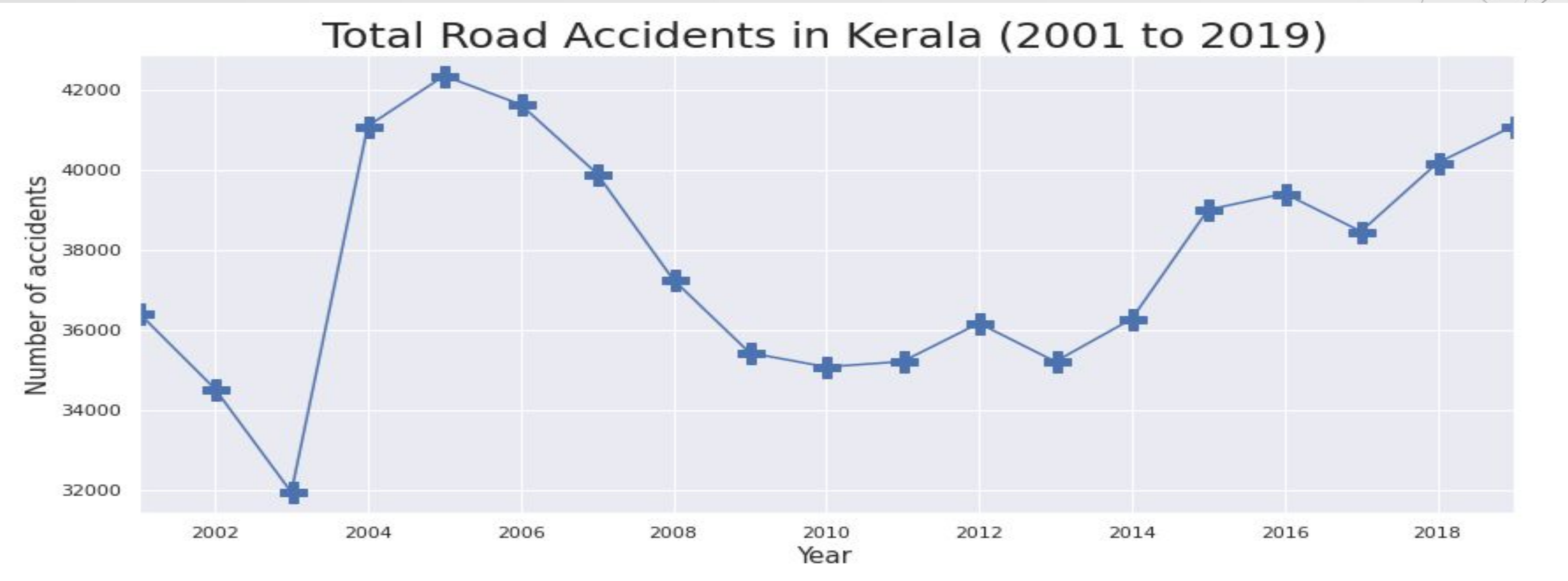
Kerala Road Accident 2001 to 2019

We collected our dataset from Official Webportal of Kerala Police : https://keralapolice.gov.in. The dataset provides number of accidents in the state monthly from the year 2001 to 2019.

data.csv

	STATE/UT	YEAR	JANUARY	FEBRUARY	MARCH	APRIL	MAY	JUNE	JULY	AUGUST	SEPTEMBER	OCTOBER	NOVEMBER	DECEMBER	TOTAL
0	Kerala	2001	3199	2935	2972	2912	2887	2829	2845	3012	3060	3048	3247	3493	36439
1	Kerala	2002	3072	2739	2643	3096	3113	2754	2731	2688	2835	2897	2908	3058	34534
2	Kerala	2003	2894	2485	2495	2570	2614	2526	2534	2569	2612	2852	2799	2997	31947
3	Kerala	2004	3661	3393	3477	3411	3239	3306	3164	3334	3399	3532	3425	3762	41103
4	Kerala	2005	3862	3422	3667	3565	3873	3407	3138	3501	3108	3505	3605	3710	42363

Visualization of the dataframe



The total number of accidents in each year from 2001-2019 is visualized.

- In 2001 total accidents where around 36000 and in 2019 it is around 41000.
- There was a hike from 2004-2006 (42000) and then became normal to 36000 and then gradually increased year by year obviously as number of vehicles increased.

For Time series Analysis And Forecasting

We transformed our dataset to a form with three attributes which specifies year, month and number of accidents for better analysis of the dataset.

df.csv

	Year	Month	No of accidents
0	2001	January	3199
1	2001	February	2935
2	2001	March	2972
3	2001	April	2912
4	2001	May	2887

03

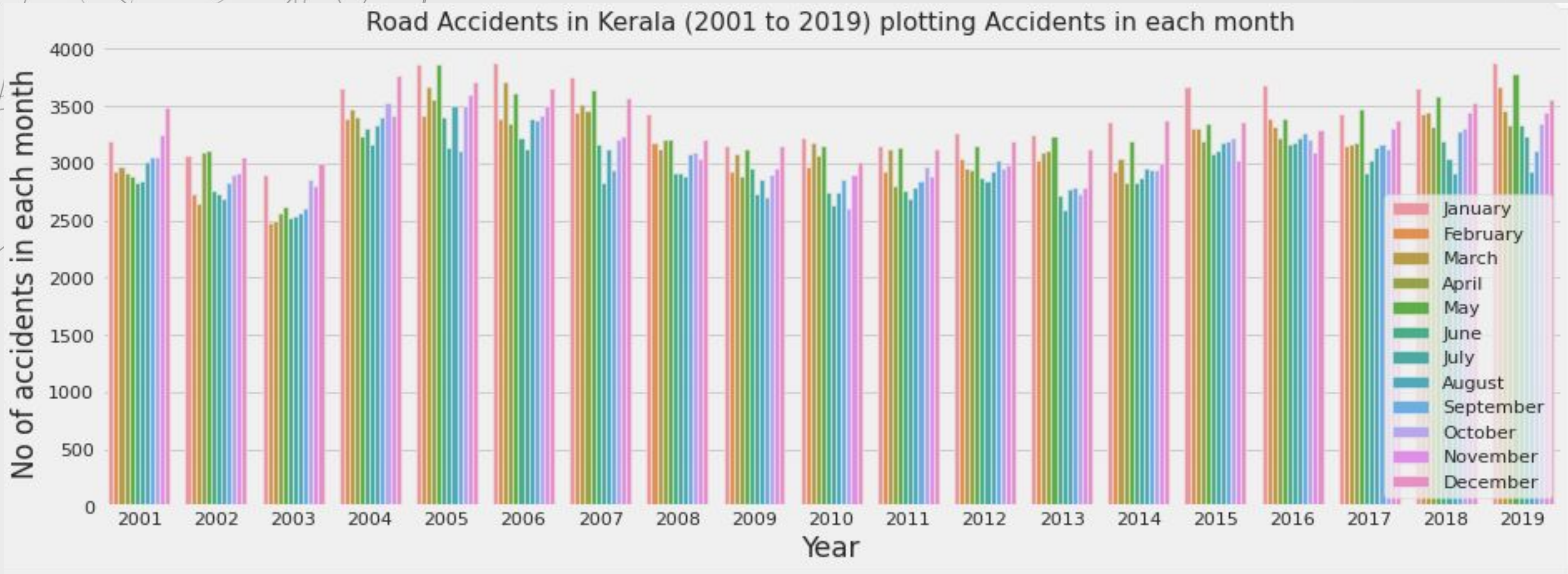
DATA ANALYSIS : VISUALIZATION

Line Plot

Average number of accidents in each year is plotted in the graph.

- In 2001 average accident was around 3000 and in 2019 it was around 3400.
- Unexpected decrease in 2003 and hike in 2004-2006 ranged 3400 to 3500.
- From 2014 there is a gradual increase as year pasess.

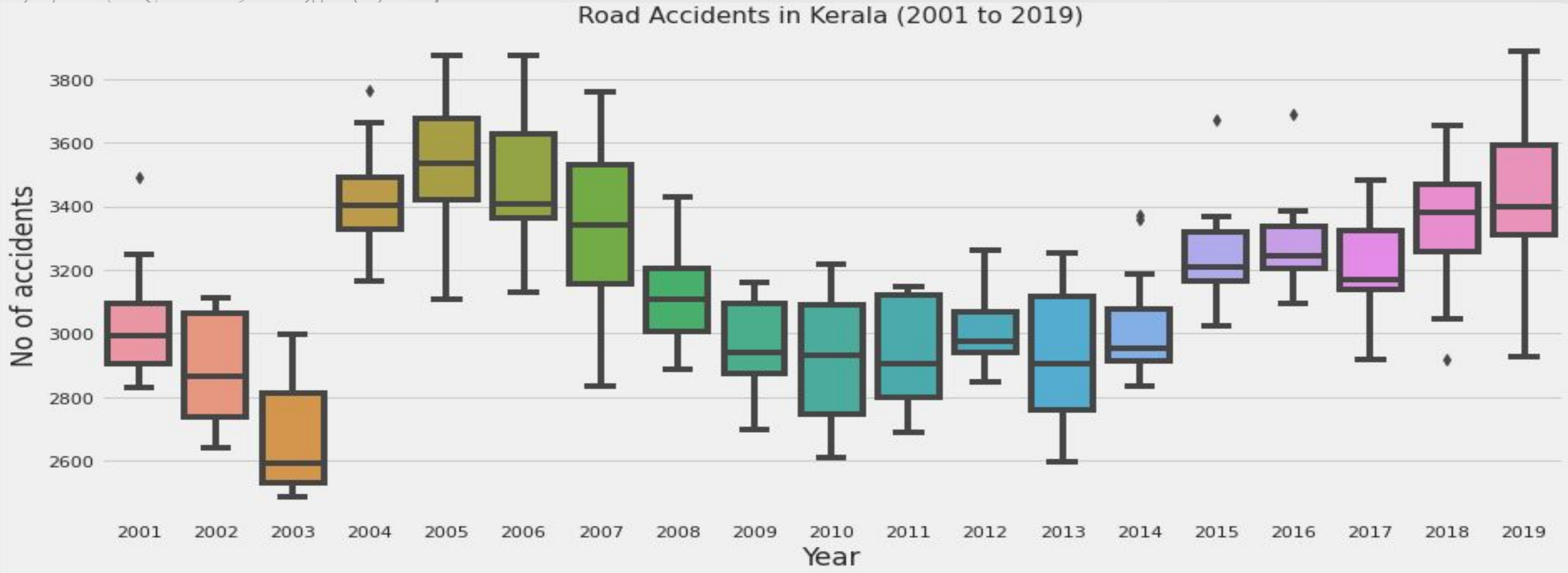
Bar plot



Number of accidents in each year month wise is plotted in the graph.

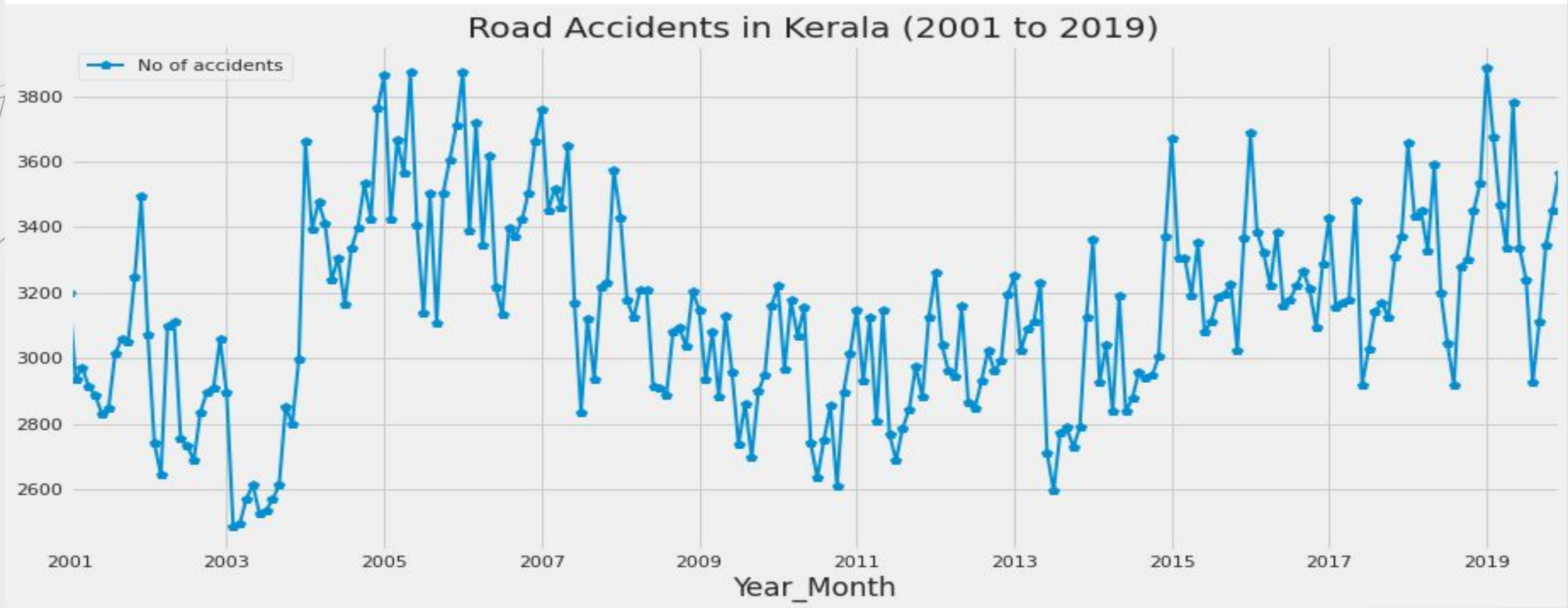
- In the month of january and december there was always an increase.
- In the month of July and august there was always a decrease.

Box plot



It plots the maximum, minimum, average, the range and outliers in number of accidents in each year.

Line plot



Plots number of accidents in each year month wise & we can notice a certain pattern in each year.

04

Forecasting Models

1. Sarimax model

- **Seasonal AutoRegressive Integrated Moving Average with eXogenous** regressors model that explicitly supports univariate time series data with a seasonal component.
- SARIMAX is essentially a linear regression model that uses a seasonal ARIMA-type model for residuals.
- It is modified SARIMA modeling, as a result of an a posteriori modification of the SARIMA model, and ANN-based modeling.
- The SARIMAX time series forecasting method is supported in Python via the **Statsmodels** library.

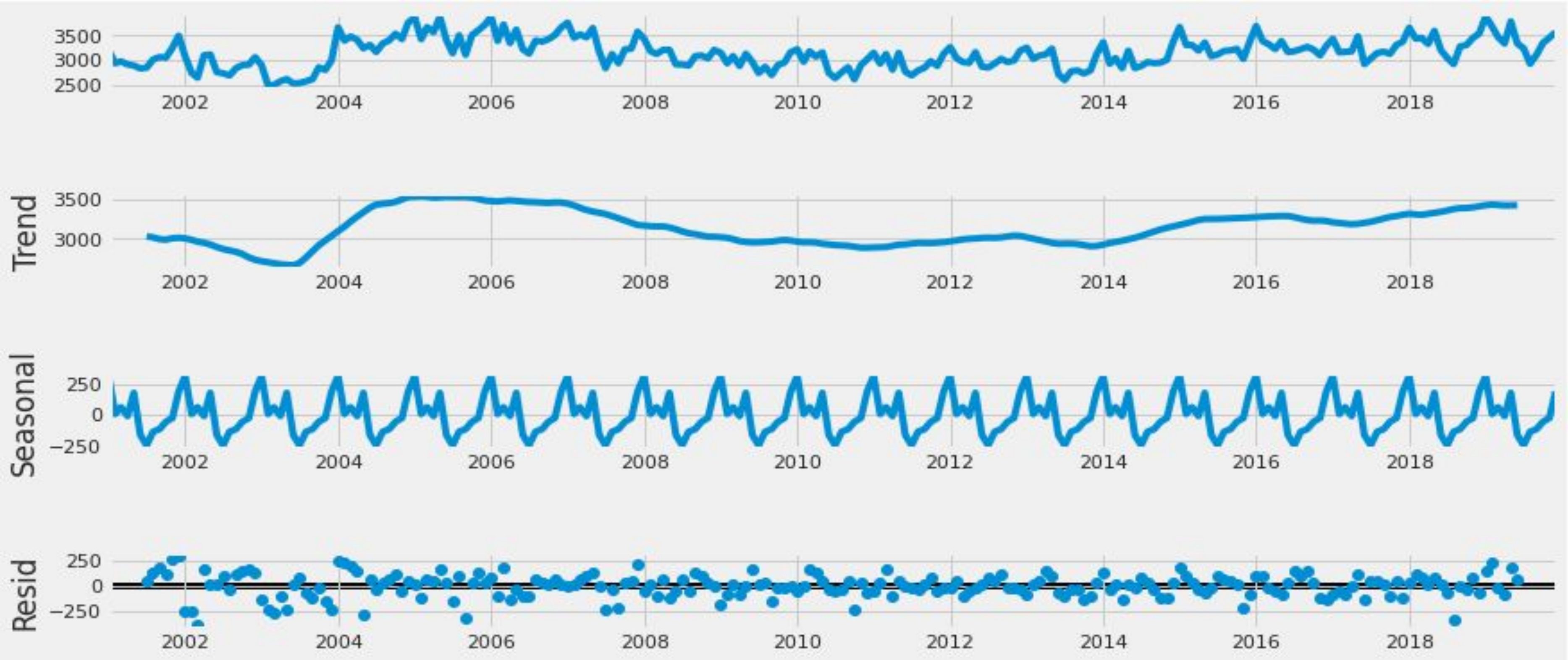
Import libraries

```
#Sarimax Model
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
```

Seasonal decomposition

We can find the trend as well as outliers
In our dataset.

```
decomposition = seasonal_decompose(monthly, freq=12)
fig = plt.figure()
fig = decomposition.plot()
fig.set_size_inches(14,6)
```



Fitting sarimax model

Fitting model

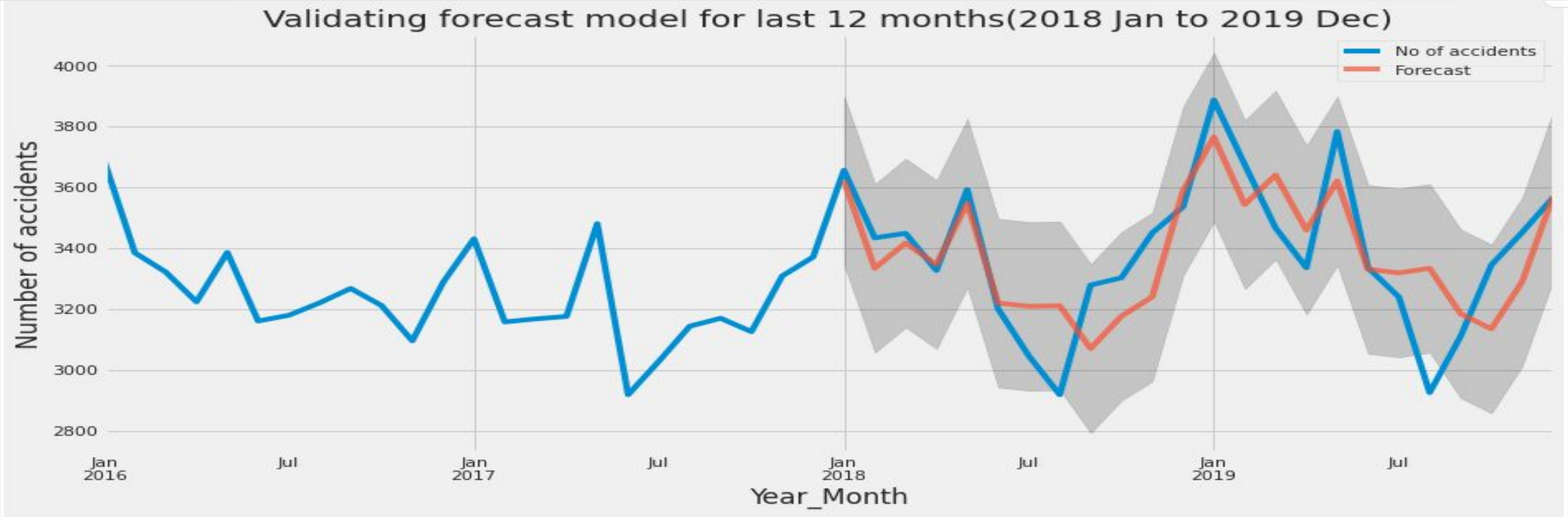
```
# Applying Seasonal ARIMA model to forcast the data
mod = sm.tsa.SARIMAX(monthly['No of accidents'], trend='n',\
                    order=(0,1,1), \
                    seasonal_order=(1,1,1,12),\
                    enforce_stationarity=False,\
                    enforce_invertibility=False)
results = mod.fit()
print(results.summary())
```

summary

```
                                  SARIMAX Results
==========================================================================================
Dep. Variable:                   No of accidents   No. Observations:                  228
Model:            SARIMAX(0, 1, 1)x(1, 1, 1, 12)   Log Likelihood               -1282.600
Date:                           Fri, 08 May 2020   AIC                           2573.200
Time:                                   07:01:28   BIC                           2586.413
Sample:                               01-01-2001   HQIC                          2578.547
                                    - 12-01-2019
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.4424      0.047     -9.404      0.000      -0.535      -0.350
ar.S.L12      -0.1015      0.074     -1.365      0.172      -0.247       0.044
ma.S.L12      -0.7335      0.089     -8.253      0.000      -0.908      -0.559
sigma2      1.998e+04   1545.047     12.929      0.000    1.69e+04     2.3e+04
===================================================================================
Ljung-Box (Q):                       31.13   Jarque-Bera (JB):               177.78
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               0.49   Skew:                             0.55
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.47
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Validating model for last 12 months(2018 Jan to 2019 Dec)




```
pred = results.get_prediction(start=pd.to_datetime('2018-01-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = monthly['2016':].plot(label='observed')

pred.predicted_mean.plot(ax=ax, label='Forecast', alpha=.7, figsize=(14, 6))
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_title('Validating forecast model for last 12 months(2018 Jan to 2019 Dec)')
ax.set_xlabel('Year_Month')
ax.set_ylabel('Number of accidents')
plt.legend()
plt.show()
```

Measuring Accuracy : MSE and RMSE values closer to zero are better

```
monthly_forecasted = pred.predicted_mean
monthly_truth = monthly['2018-01-01':]
mse = ((monthly_forecasted -  monthly_truth["No of accidents"]) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

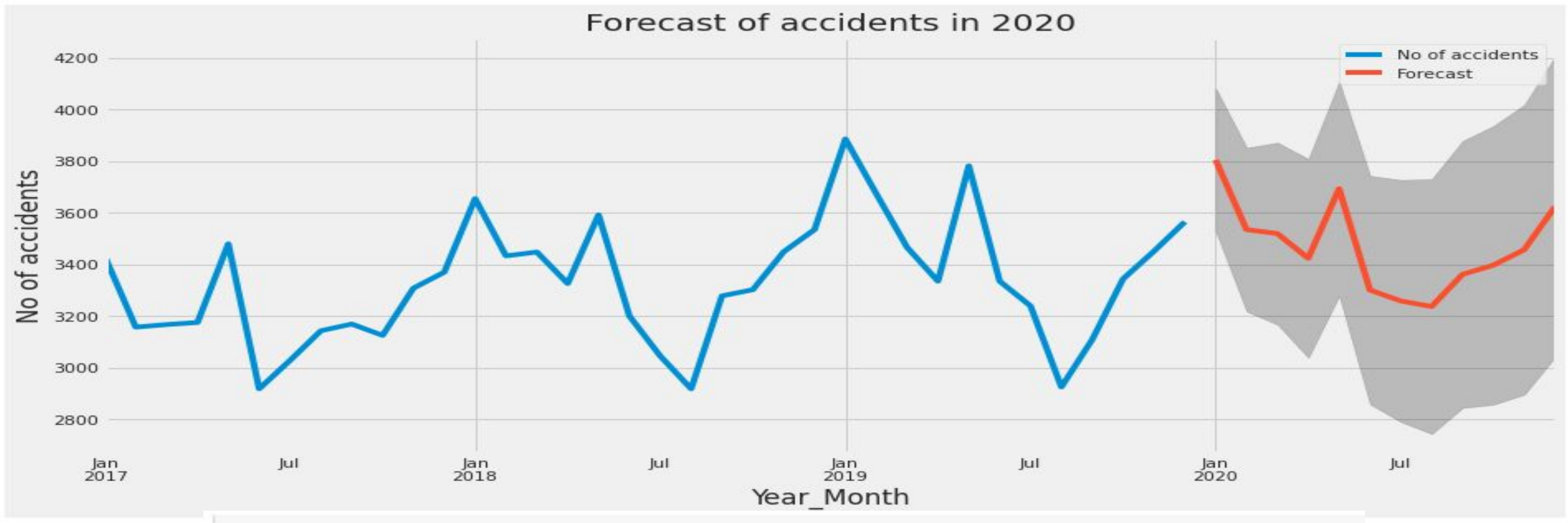
```
The Mean Squared Error is 24386.25
The Root Mean Squared Error is 156.16
```

Comparison of actual and forecasted accidents

```
--------------------------------
Actual accidents
--------------------------------
                 No of accidents
Year_Month
2018-01-01                  3655
2018-02-01                  3433
2018-03-01                  3448
2018-04-01                  3326
2018-05-01                  3592
2018-06-01                  3200
2018-07-01                  3045
2018-08-01                  2918
2018-09-01                  3278
2018-10-01                  3302
2018-11-01                  3449
2018-12-01                  3535
```

```
--------------------------------
Forcasted accidents
--------------------------------
2018-01-01    3621.088199
2018-02-01    3333.255854
2018-03-01    3416.293873
2018-04-01    3345.502574
2018-05-01    3546.698444
2018-06-01    3219.066757
2018-07-01    3208.207104
2018-08-01    3210.158254
2018-09-01    3069.720556
2018-10-01    3175.853495
2018-11-01    3239.120191
2018-12-01    3587.271986
Freq: MS, dtype: float64
```

FUTURE PREDICTION




```
pred_uc = results.get_forecast(steps=12)
pred_ci = pred_uc.conf_int()
ax = monthly['2017':].plot(label='observed', figsize=(14, 6))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_title('Forecast of accidents in 2020')
ax.set_xlabel('Year_Month')
ax.set_ylabel('No of accidents')
plt.legend()
plt.show()
```

FUTURE PREDICTION

- Here we forecast the accidents for the next 12 months.
- This parameter can me modified in the line “pred_uc = results.get_forecast(steps=12)” of the code.
- Since covid-19 pandemic & lockdown the prediction may go wrong so the prediction may be accurate till March 2020

```
forecast = pred_uc.predicted_mean
forecast.head(12)
```

```
2020-01-01    3804.661000
2020-02-01    3534.921443
2020-03-01    3519.431347
2020-04-01    3423.661089
2020-05-01    3693.416708
2020-06-01    3300.730473
2020-07-01    3258.462317
2020-08-01    3236.957846
2020-09-01    3361.451001
2020-10-01    3396.915507
2020-11-01    3457.096149
2020-12-01    3624.013140
Freq: MS, dtype: float64
```

2. Facebook Prophet Model

- Prophet is a procedure 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.
- Prophet is “framing the forecasting problem as a curve-fitting exercise” rather than looking explicitly at the time based dependence of each observation.

Import libraries

```
#Facebook Prophet Model
from fbprophet import Prophet
from fbprophet.plot import plot_plotly
from fbprophet.plot import add_changepoints_to_plot
```

Working with prophet model

The input to Prophet is always a dataframe with two columns: ds and y. The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

```
train_dataset = pd.DataFrame()
train_dataset['ds'] = pmonthly['Year_Month']
train_dataset['y'] = pmonthly['No of accidents']
train_dataset.head()
```

	ds	y
0	2001-01-01	3199
1	2001-02-01	2935
2	2001-03-01	2972
3	2001-04-01	2912
4	2001-05-01	2887

Fitting prophet and comparison with actual data and forecasted data

```
#BY default interval_width/ confidence factor is 80%
prophet = Prophet(interval_width=0.95)
prophet.fit(train_dataset)
```

Fitting model

Actual data

	ds	y
0	2001-01-01	3199
1	2001-02-01	2935
2	2001-03-01	2972
3	2001-04-01	2912
4	2001-05-01	2887
5	2001-06-01	2829
6	2001-07-01	2845
7	2001-08-01	3012
8	2001-09-01	3060
9	2001-10-01	3048
10	2001-11-01	3247
11	2001-12-01	3493

	ds	yhat	yhat_lower	yhat_upper
0	2001-01-01	3089.339148	2727.113623	3435.087391
1	2001-02-01	2789.072698	2457.104912	3107.608430
2	2001-03-01	2909.437462	2557.283929	3239.593010
3	2001-04-01	2851.012270	2528.316623	3188.022516
4	2001-05-01	2990.202933	2638.702075	3346.007163
5	2001-06-01	2690.621725	2346.117234	3032.850451
6	2001-07-01	2623.597571	2303.764335	2987.543560
7	2001-08-01	2734.264840	2395.039568	3095.956589
8	2001-09-01	2773.743657	2431.664987	3104.847340
9	2001-10-01	2832.954762	2497.623275	3177.281089
10	2001-11-01	2889.016836	2530.621977	3240.332081
11	2001-12-01	3112.471903	2768.060246	3466.641427

- **yhat- forecasted data**
- **Yhat_lower - lower range of forecasting**
- **Yhat_upper - upper range of forecasting**

Future Forecasting

```
future = prophet.make_future_dataframe(periods=12, freq='M')
forecast = prophet.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(12)
```

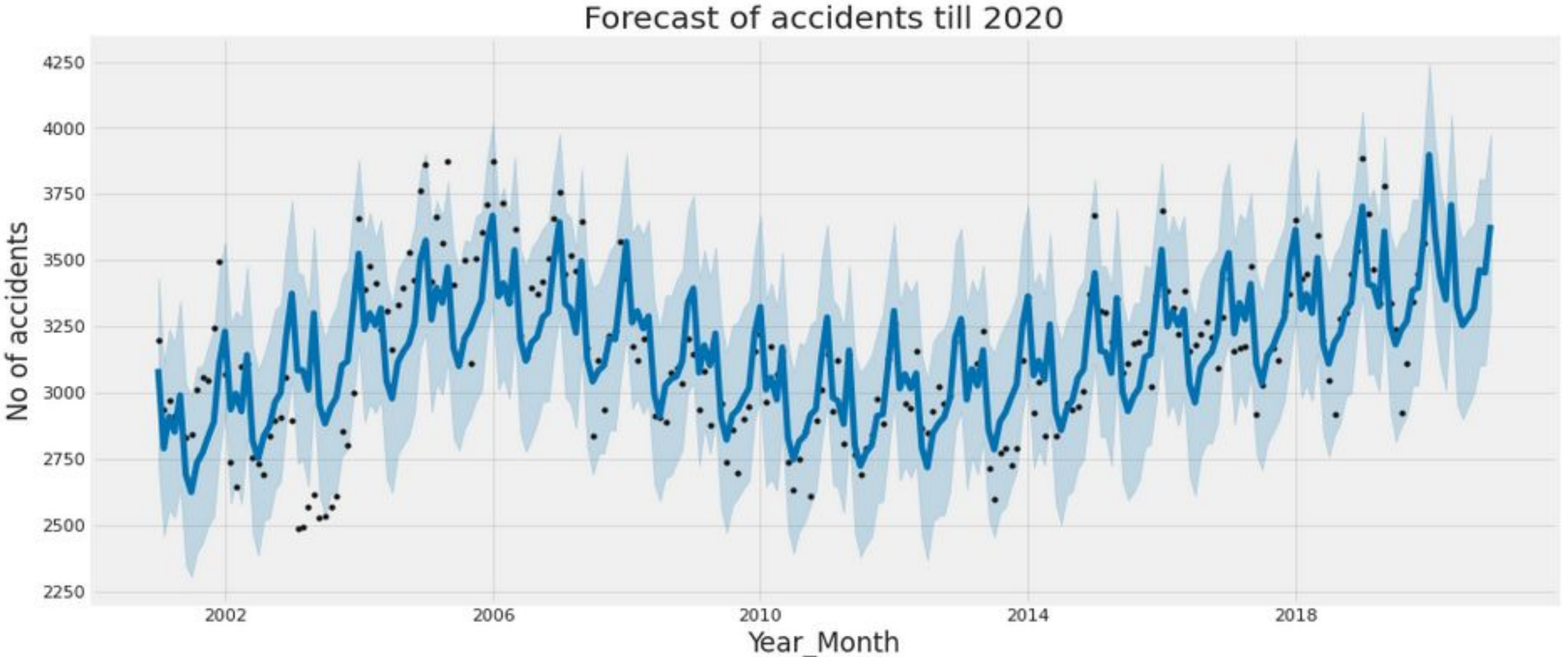
Future forecasting : 'periods' in the code define number of months

- **ds -date**
- **yhat-** forecasted data
- **Yhat_lower -** lower range of forecasting
- **Yhat_upper -** upper range of forecasting

	ds	yhat	yhat_lower	yhat_upper
228	2019-12-31	3897.910428	3560.185915	4235.743585
229	2020-01-31	3608.027858	3274.716234	3939.450815
230	2020-02-29	3435.772237	3067.050162	3772.566949
231	2020-03-31	3350.523598	3018.603919	3686.664133
232	2020-04-30	3708.802775	3369.395460	4048.233563
233	2020-05-31	3328.034627	2980.413363	3648.544196
234	2020-06-30	3253.235993	2917.837251	3602.223052
235	2020-07-31	3283.876179	2928.202086	3616.977513
236	2020-08-31	3317.167435	2953.306282	3690.699943
237	2020-09-30	3463.047361	3105.777414	3789.926751
238	2020-10-31	3452.169972	3106.922316	3798.508070
239	2020-11-30	3633.758823	3277.131658	3960.488958

Prophet plots the observed values of our time series (the black dots), the forecasted values (blue line) and the uncertainty intervals of our forecasts (the blue shaded regions).

```
fig = prophet.plot(forecast,xlabel="Year_Month",
                   ylabel="No of accidents",
                   figsize=(14, 6))
ax = fig.gca()
ax.set_title("Forecast of accidents till 2020")
plt.show
```



Details of plot using prophet

It's always nice to check how does the model perform on historical data.

- Deep blue line is forecasting number of accidents
- Black dots are actual number of accidents
- The light blue shade is 95% confidence interval around the forecast.
- From 2020 black dots are not visible as it shows only the future forecast
- Since covid-19 pandemic & lockdown the prediction may go wrong so the prediction may be accurate till March 2020

05

CONCLUSION

KERALA ROAD ACCIDENT DATASET

The Kerala Road Accident dataset has been analysed and been forecasted by SARIMAX and Facebook prophet model. Both the model provides almost the same accuracy.

- There are variations at many data points it is because we have considered only the number of accidents as the factor, there can be many factors for the accidents like road, government measures, disasters
- Our future prediction will go wrong from the month of april 2020 as covid-19 affected us.

As a future enhancement we would like to work on this dataset by considering most of the factors that lead to accidents

REFERENCES

- **Kerala Police Official Web portal:**
  https://keralapolice.gov.in/public-information/crime-statistics/road-accident

- **Medium : Towards Data science**
  1. https://towardsdatascience.com/how-to-forecast-sales-with-python-using-sarima-model-ba600992fa7d
  2. https://towardsdatascience.com/time-series-forecasting-with-a-sarima-model-db051b7ae459
  3. https://towardsdatascience.com/forecasting-with-prophet-d50bbfe95f91

CODES at Kaggle:

https://www.kaggle.com/jithu10/kerala-road-accidents-kerala-police-2001-2019

THANKS