Time Series Data Analysis

Erekle Magradze

Director of Engineering at MaxinAI
Associate Professor at Ilia State University

DataFest Tbilisi 2020
16.12.2020

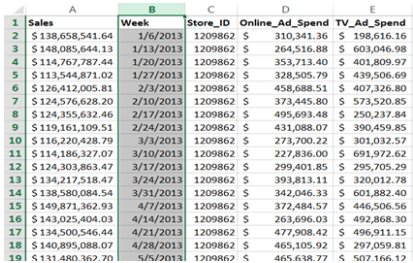
www.maxinai.com

Outline

- EDA – Exploratory Data Analysis
- Ways of Prediction
- Linear Regression – the baseline
- Autoregressive Models
- Model Accuracy Evaluation
- FB Prophet & Neural Prophet

EDA – Raw Data

- Use any cloud storage – and create sharable location, where you will be able to store the raw data in the format you have got it!
- Time Series Data – timestamp and data?



	A	B	C	D	E
1	Sales	Week	Store_ID	Online_Ad_Spend	TV_Ad_Spend
2	$ 138,658,541.64	1/6/2013	1209862	$ 310,341.36	$ 198,616.16
3	$ 148,085,644.13	1/13/2013	1209862	$ 264,516.88	$ 603,046.98
4	$ 114,767,787.44	1/20/2013	1209862	$ 353,713.40	$ 401,809.97
5	$ 113,544,871.02	1/27/2013	1209862	$ 328,505.79	$ 439,506.69
6	$ 126,412,005.81	2/3/2013	1209862	$ 458,688.51	$ 407,326.80
7	$ 124,576,628.20	2/10/2013	1209862	$ 373,445.80	$ 573,520.85
8	$ 124,355,632.46	2/17/2013	1209862	$ 495,693.48	$ 250,237.84
9	$ 119,161,109.51	2/24/2013	1209862	$ 431,088.07	$ 390,459.85
10	$ 116,220,428.79	3/3/2013	1209862	$ 273,700.22	$ 301,032.57
11	$ 114,186,327.07	3/10/2013	1209862	$ 227,836.00	$ 691,972.62
12	$ 124,303,863.47	3/17/2013	1209862	$ 299,401.85	$ 295,705.29
13	$ 134,217,518.47	3/24/2013	1209862	$ 393,813.11	$ 320,012.78
14	$ 138,580,084.54	3/31/2013	1209862	$ 342,046.33	$ 601,882.40
15	$ 149,871,362.93	4/7/2013	1209862	$ 372,484.57	$ 446,506.56
16	$ 143,025,404.03	4/14/2013	1209862	$ 263,696.03	$ 492,868.30
17	$ 134,500,546.44	4/21/2013	1209862	$ 477,908.42	$ 496,911.15
18	$ 140,895,088.07	4/28/2013	1209862	$ 465,105.92	$ 297,059.81
19	$ 131,480,362.70	5/5/2013	1209862	$ 465,638.77	$ 507,166.12

	A	B	C
1	Year	Quarter	Sales
2	2012	1	$165,000.00
3		2	$253,000.00
4		3	$316,000.00
5		4	$287,000.00
6	2013	1	$257,000.00
7		2	$308,000.00
8		3	$376,000.00
9		4	$351,000.00

Date	Ozone ($\mu g/m^3$)	Temperature (°C)	Relative humidity (%)	*n* deaths
1 Jan 2002	4.59	−0.2	75.7	199
2 Jan 2002	4.88	0.1	77.5	231
3 Jan 2002	4.71	0.9	81.3	210
4 Jan 2002	4.14	0.5	85.4	203
5 Jan 2002	2.01	4.3	93.5	224
6 Jan 2002	2.4	7.1	96.4	198
7 Jan 2002	4.08	5.2	93.5	180
8 Jan 2002	3.13	3.5	81.5	188
9 Jan 2002	2.05	3.2	88.3	168
10 Jan 2002	5.19	5.3	85.4	194
11 Jan 2002	3.59	3.0	92.6	223
12 Jan 2002	12.87	4.8	94.2	201

Time	Type	Dur	nKey	Ins	Del	nFix	DFix	ScSpan	Turn	HCross
385053	8	1439	0	0	0	0	0	0	0	0.0000
386492	2	699	0	0	0	1	1	0	0	1.7265
387192	4	782	1	1	0	0	0	0	0	0.0000
387974	2	1557	0	0	0	4	3	114	1	2.1819
389532	4	993	2	1	1	0	0	0	0	0.0000
390525	2	305	0	0	0	2	2	103	0	1.1910
390830	6	180	0	0	0	1	1	0	0	0.6554
391010	4	787	2	2	0	0	0	0	0	0.0000
391797	8	1169	0	0	0	0	0	0	0	0.0000
392966	2	1003	0	0	0	2	2	3	0	1.5730
393969	1	698	0	0	0	2	2	31	0	1.5590
394667	5	1237	0	0	0	3	3	19	0	1.3310
395904	6	1930	3	3	0	2	1	0	0	2.5902
397834	2	1301	0	0	0	2	2	101	0	2.1583
399135	8	1943	0	0	0	0	0	0	0	0.0000
401078	2	739	0	0	0	1	1	0	0	1.5230
401817	4	315	1	1	0	0	0	0	0	0.0000
402132	2	596	0	0	0	2	2	1	0	1.4486
402729	1	9297	0	0	0	15	12	31	8	2.3061
412027	2	488	0	0	0	1	1	0	0	1.3742
412516	4	1331	2	2	0	0	0	0	0	0.0000
413847	2	862	0	0	0	2	1	0	0	2.8657
414710	4	1345	2	2	0	0	0	0	0	0.0000
416056	4	1153	1	1	0	0	0	0	0	0.0000
417209	2	999	0	0	0	1	1	0	0	3.0425
418209	4	700	1	1	0	0	0	0	0	0.0000
418909	6	450	0	0	0	2	1	0	0	1.7265
419359	4	658	2	2	0	0	0	0	0	0.0000
420017	8	1247	0	0	0	0	0	0	0	0.0000
421264	2	332	0	0	0	1	1	0	0	2.6012
421596	1	607	0	0	0	1	1	0	0	0.6500
422204	2	3117	0	0	0	8	3	8	1	2.4995
425321	6	366	0	0	0	1	1	0	0	2.6012
425687	4	960	2	2	0	0	0	0	0	0.0000

Which timestamp should we use????

Use the time stamp which can be used as an index for all data sources

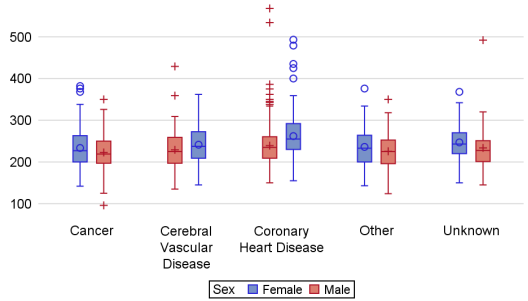
EDA – Raw Data – Python Pandas DFs

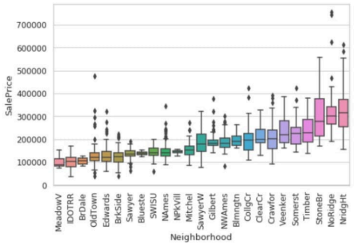
- Most of the data management software are capable to export their data in CSV format
- You can read CSV file as a python pandas Data Frame
  pd.read_csv(…)
- Once you have the Data Frames, you can merge them easily e.g.
- There are lots of short tips and tricks pages (like https://queirozf.com/entries/pandas-dataframes-merge-join-examples) to quickly merge your files in one

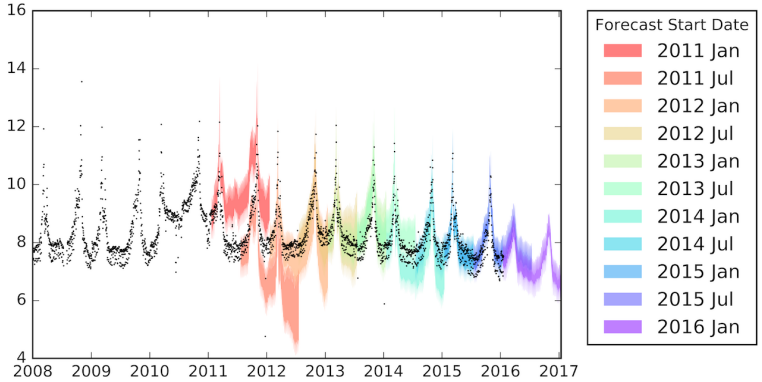
```
import pandas as pd

df_employees_sal = pd.DataFrame({
    'year':[1980,1981,1980,1981,1980,1981,1980,1981],
    'id':[1,1,2,2,3,3,4,4],
    'name':['alice','alice','bob','bob','charlie','charlie','david','david'],
    'salary':[30000,30000,40000,41000,35000,40000,45000,45000],
    'company_id':[1,1,2,2,1,1,2,2]})

df_companies_rev = pd.DataFrame({
    'year':[1980,1981,1980,1981],
    'id':[1,1,2,2],
    'name':['bell labs','bell labs','xerox','xerox'],
    'revenue':[1130000,1130000,5000000,500000]})

pd.merge(
    df_employees_sal,
    df_companies_rev,
    left_on=['year','company_id'],
    right_on=['year','id']
)
```

We have merged data, what's next?

- Save the merged dataframe as a CSV file somewhere in the cloud storage – if you mess up with the dataframe, you will be able to start over again quickly
- Fill the gaps – use e.g. Python Pandas ffill, bfill functions, or you can do the rolling average filling of NA-s.
- Plot the data!
  Use boxplots for thousands of records
  Use the dot plot for hundreds of data





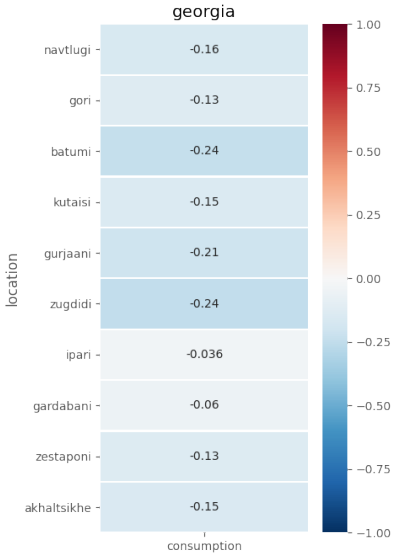


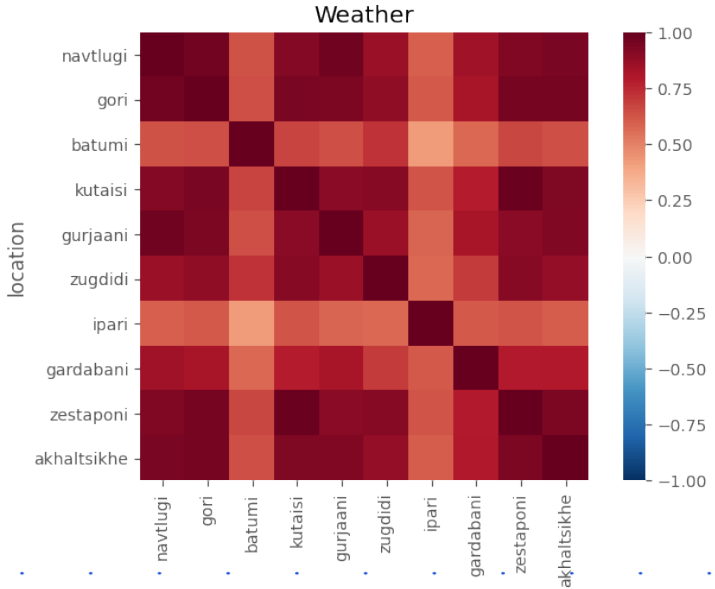
Why we need visualization?

We will see if there are outliers, anomalies, value ranges and may be some pattern, relationship of different factors/variables/regressors

Correlations

- Search for correlations among variables you have
- Use heatmaps
- Check the relationship against a single (target) variable





Even if you plan to use the Linear regression for the forecasting
You should use only non correlated Variables to get a good result – So, before starting the analysis we should
Pick only non-correlated or less-correlated variables

Always save the plots somewhere as a file, it will save your time when you need to prepare the report, which is also key part of the data scientists' job

End of EDA Section

Ways of Prediction

- Analyzing a single variable and trying to model and forecast it

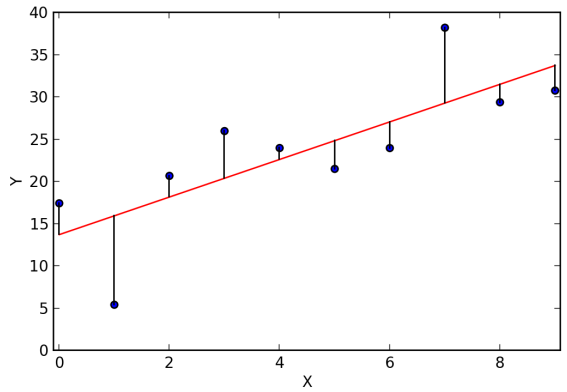
$$\{t_0, t_1, t_2, ..., t_{n-2}, t_{n-1}, t_n\} \rightarrow t_{n+1}$$

- Use additional features/variables/regressors/predictors/exogenous variables (may be you know one more name of them ☺) for better modelling and higher forecasting accuracy

$$\{t_0, t_1, t_2, ..., t_{n-2}, t_{n-1}, t_n, ..., x_0, x_1, x_2, ..., x_{n-2}, x_{n-1}, x_n, ..., y_0, y_1, y_2, ..., y_{n-2}, y_{n-1}, y_n, ..., z_0, z_1, z_2, ..., z_{n-2}, z_{n-1}, z_n\} \rightarrow t_{n+1}$$

Linear Regression the baseline

- Linear regression approximates the values with line, this is the simplest possible model we can have




```
from sklearn import linear_model
```

```
X = df
y = target["MEDV"]
```

```
lm = linear_model.LinearRegression()
model = lm.fit(X,y)
```

```
predictions = lm.predict(X)
print(predictions)[0:5]
```

Autoregressive Models

- ARIMA, SARIMA (Seasonal ARIMA) belongs to autoregressive or regressive models class
- ARIMA – Auto Regressive (AR) Integrated Moving Average (MA)
- ARIMA is characterized by three terms

  p is the order of the AR term
  q is the order of the MA term
  d is the number of differencing required to make the time series stationary

The best way to start the modelling the time series is to make them first stationary

Definition: ***A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time.***
Reason: ARIMA uses own lags as predictors (features), since ARIMA is a linear model, it's features should be non-correlated – stationarity guarantees that!
How? The most common way to make the time series stationary is to subtract previous value to the current value, if the time series are complex, it might need more then one iteration of substation. Thus ***d*** is the value of minimum numbers of subtractions to make the time series data stationary.

p and *q* values

- ***p*** is the order of the AR term – AR models only depend on previous values

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

- ***q*** is the order of the MA term – MA models only depend on lagged forecast errors

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

- ARIMA model predicts the value with the linear combination of lags (up to p lags) and linear combination of lagged forecast errors (up to q lags)

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

Auto ARIMA

```python
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm

df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/wwwusage.csv', names=['value'], header=0)

model = pm.auto_arima(df.value, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

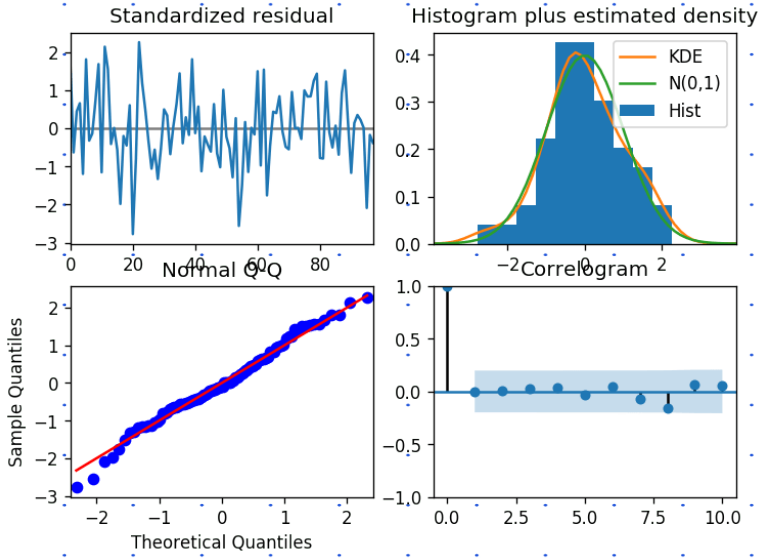
ADF - Augmented Dickey-Fuller Test - Against null hypothesis that data is non-stationary

```
#> Fit ARIMA: order=(1, 2, 1); AIC=525.586, BIC=535.926, Fit time=0.060 seconds
#> Fit ARIMA: order=(0, 2, 0); AIC=533.474, BIC=538.644, Fit time=0.005 seconds
#> Fit ARIMA: order=(1, 2, 0); AIC=532.437, BIC=540.192, Fit time=0.035 seconds
#> Fit ARIMA: order=(0, 2, 1); AIC=525.893, BIC=533.648, Fit time=0.040 seconds
#> Fit ARIMA: order=(2, 2, 1); AIC=515.248, BIC=528.173, Fit time=0.105 seconds
#> Fit ARIMA: order=(2, 2, 0); AIC=513.459, BIC=523.798, Fit time=0.063 seconds
#> Fit ARIMA: order=(3, 2, 1); AIC=512.552, BIC=528.062, Fit time=0.272 seconds
#> Fit ARIMA: order=(3, 2, 0); AIC=515.284, BIC=528.209, Fit time=0.042 seconds
#> Fit ARIMA: order=(3, 2, 2); AIC=514.514, BIC=532.609, Fit time=0.234 seconds
#> Total fit time: 0.865 seconds
#>                             ARIMA Model Results
#> ==============================================================================
#> Dep. Variable:                      D2.y   No. Observations:                   98
#> Model:                 ARIMA(3, 2, 1)   Log Likelihood                -250.276
#> Method:                       css-mle   S.D. of innovations              3.069
#> Date:                Sat, 09 Feb 2019   AIC                            512.552
#> Time:                        12:57:22   BIC                            528.062
#> Sample:                             2   HQIC                           518.825
#>
#> ==============================================================================
#>                  coef    std err          z      P>|z|      [0.025      0.975]
#> ------------------------------------------------------------------------------
#> const          0.0234      0.058      0.404      0.687      -0.090       0.137
#> ar.L1.D2.y     1.1586      0.097     11.965      0.000       0.969       1.348
#> ar.L2.D2.y    -0.6640      0.136     -4.890      0.000      -0.930      -0.398
#> ar.L3.D2.y     0.3453      0.096      3.588      0.001       0.157       0.534
#> ma.L1.D2.y    -1.0000      0.028    -36.302      0.000      -1.054      -0.946
#>                                     Roots
#> =============================================================================
#>                  Real          Imaginary           Modulus         Frequency
#> -----------------------------------------------------------------------------
#> AR.1            1.1703           -0.0000j            1.1703           -0.0000
#> AR.2            0.3763           -1.5274j            1.5731           -0.2116
#> AR.3            0.3763           +1.5274j            1.5731            0.2116
#> MA.1            1.0000           +0.0000j            1.0000            0.0000
#>
```

www.maxinai.com

Model Evaluation

```
model.plot_diagnostics(figsize=(7,5))
plt.show()
```



Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.

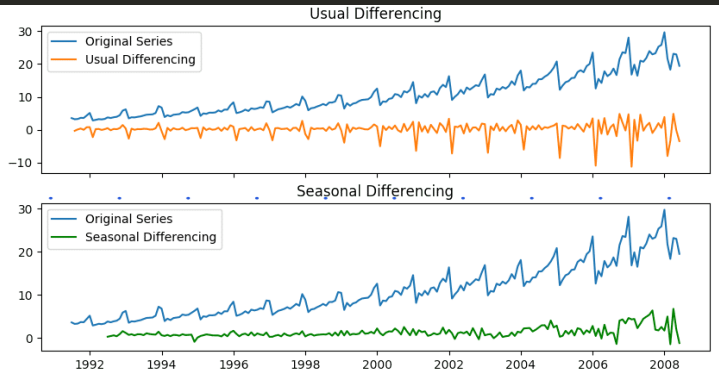
Top Right: The density plot suggest normal distribution with mean zero.

Bottom left: All the dots should fall perfectly in line with the red line. Any significant deviations would imply the distribution is skewed.

Bottom Right: The Correlogram, aka, ACF plot shows the residual errors are not autocorrelated. Any autocorrelation would imply that there is some pattern in the residual errors which are not explained in the model. So you will need to look for more X's (predictors) to the model.

Seasonal Stationarity

```
# Import
data = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'], index_col='date')

# Plot
fig, axes = plt.subplots(2, 1, figsize=(10,5), dpi=100, sharex=True)

# Usual Differencing
axes[0].plot(data[:], label='Original Series')
axes[0].plot(data[:].diff(1), label='Usual Differencing')
axes[0].set_title('Usual Differencing')
axes[0].legend(loc='upper left', fontsize=10)


# Seasinal Dei
axes[1].plot(data[:], label='Original Series')
axes[1].plot(data[:].diff(12), label='Seasonal Differencing', color='green')
axes[1].set_title('Seasonal Differencing')
plt.legend(loc='upper left', fontsize=10)
plt.suptitle('a10 - Drug Sales', fontsize=16)
plt.show()
```

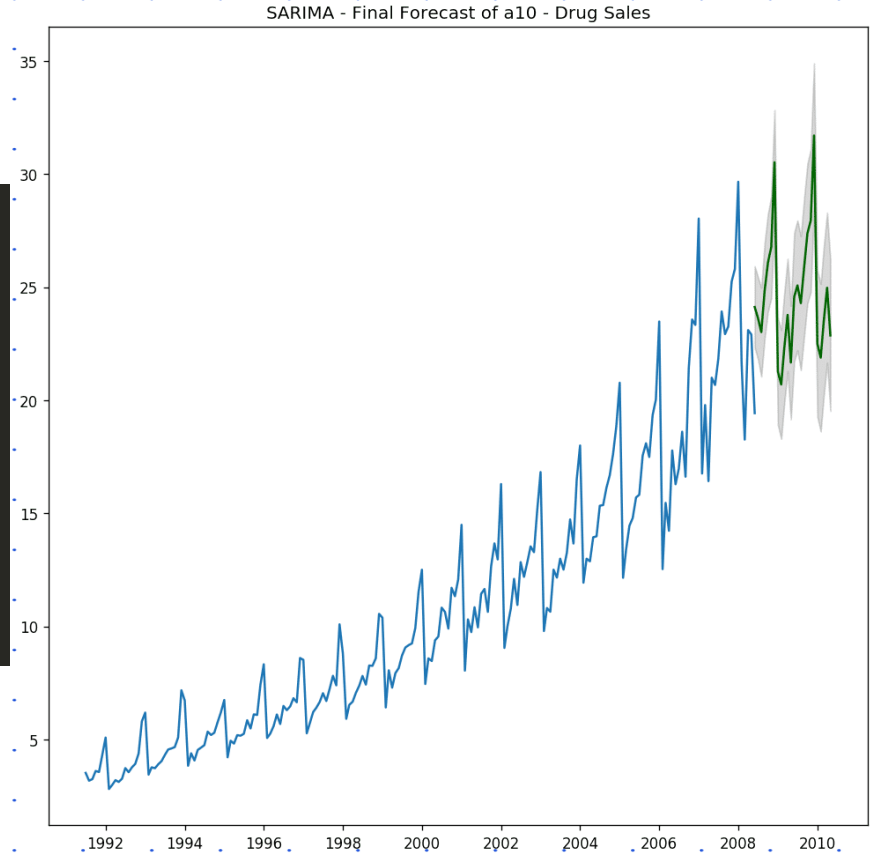


SARIMA

```
# !pip3 install pyramid-arima
import pmdarima as pm

# Seasonal - fit stepwise auto-ARIMA
smodel = pm.auto_arima(data, start_p=1, start_q=1,
                       test='adf',
                       max_p=3, max_q=3, m=12,
                       start_P=0, seasonal=True,
                       d=None, D=1, trace=True,
                       error_action='ignore',
                       suppress_warnings=True,
                       stepwise=True)
```

SARIMA Forecast

```python
# Forecast
n_periods = 24
fitted, confint = smodel.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = pd.date_range(data.index[-1], periods = n_periods, freq='MS')

# make series for plotting purpose
fitted_series = pd.Series(fitted, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.plot(data)
plt.plot(fitted_series, color='darkgreen')
plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("SARIMA - Final Forecast of a10 - Drug Sales")
plt.show()
```



Model Accuracy Evaluation

```python
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)                           # ME
    mae = np.mean(np.abs(forecast - actual))                  # MAE
    mpe = np.mean((forecast - actual)/actual)                 # MPE
    rmse = np.mean((forecast - actual)**2)**.5                # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]                 # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)                           # minmax
    acf1 = acf(fc-test)[1]                                    # ACF1
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse, 'acf1':acf1,
            'corr':corr, 'minmax':minmax})
```

Cross Validation for Time Series:

- Out of time Cross-Validation
- Stepwise Cross-Validation

www.maxinai.com

FB Prophet and Neural Prophet

Techniques available out of the box from Facebook

Prophet - https://facebook.github.io/prophet/
https://facebook.github.io/prophet/docs/quick_start.html

Works with single and multiple regressors (very similar approach to ARIMA, SARIMA models, takes into account exogenous factors like national holidays, weekends etc.)

Neural Prophet – Advanced non-linear model (shifted towards Deep Learning)
https://github.com/ourownstory/neural_prophet
https://towardsdatascience.com/neural-prophet-a-time-series-modeling-library-based-on-neural-networks-dd02dc8d868d

MaxinAI

That’s it

Questions:

erekle.magradze@maxinai.com

www.maxinai.com