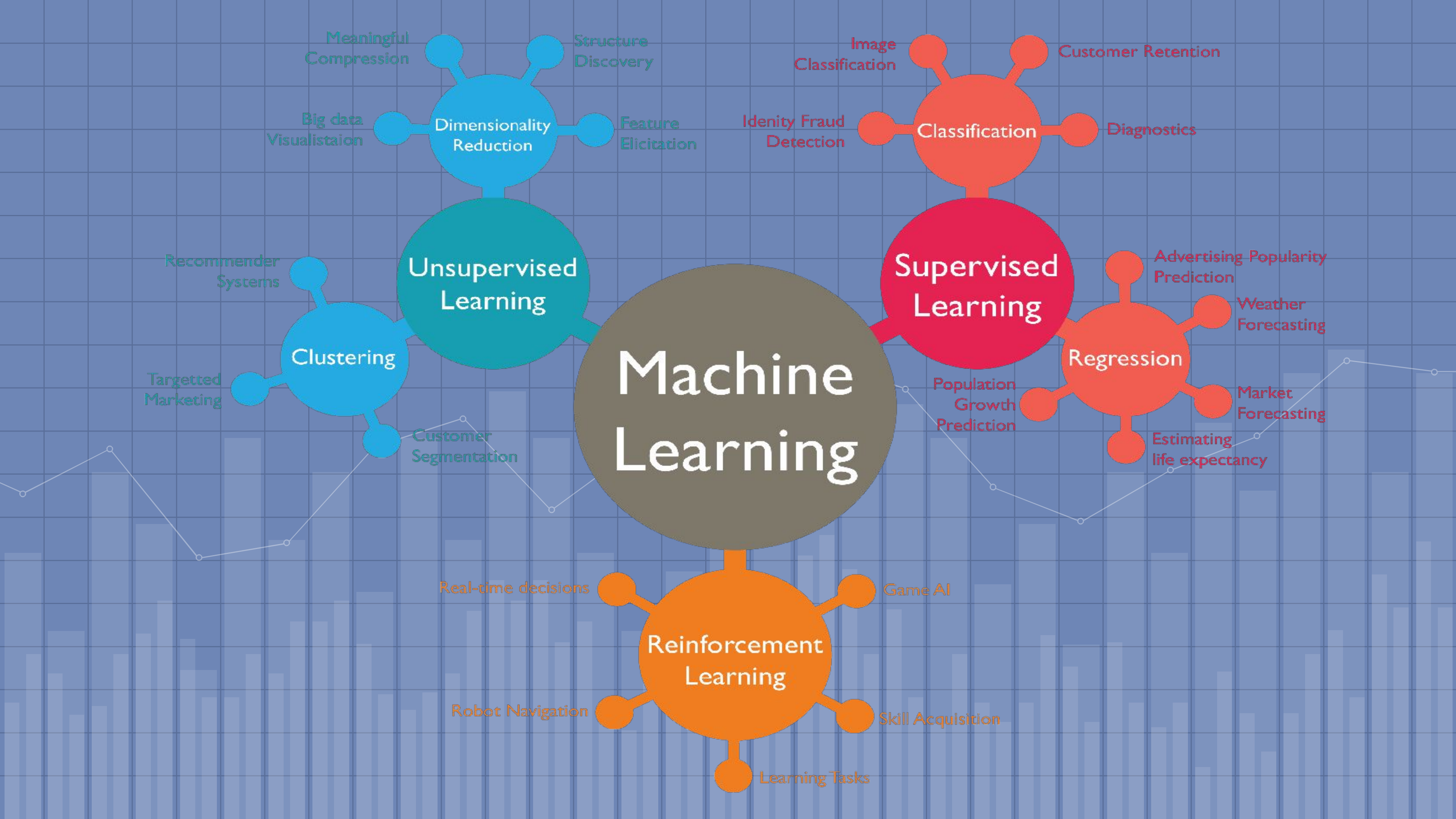
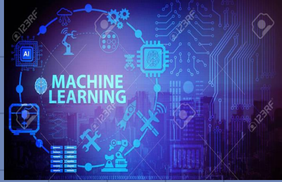

Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

An attempt to predict the future

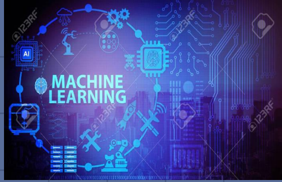


Coding machine learning and AI models to predict pharmaceutical company stocks.

Contributors:
Fernanda
Diana
Vijaya
Victor
Yan

Machine Learning Project -
FINTECH BOOTCAMP - University of Toronto

OBJECTIVES:



CODE A MODEL WHICH WILL BE AS CLOSE AS POSSIBLE TO PREDICTING CLOSING PRICES IN THE FUTURE.

MODELS STUDIED:

ARMA, ARIMA, XGBREGRESSOR, LSTM

DATA:

YAHOO FINANCE (MODERNA, ASTRAZENECA, PFIZER AND JNJ)

TIMEFRAME:

2010 : PRESENT (OR WHEN IPO PRESENTED)

ARMA & ARIMA

ARMA(p,q) and ARIMA (p ,d, q) , have p-values higher than 0.05 and are very similar, the difference is that ARMA can only be used for stationary data.

Both of those time series models can be used to forecast. And they were a great start point for this analysis, but we believe that more key indicators should be added to get a more accurate prediction. As a result, they were not considered good models for the purpose of this project.

Please see some results in the next slides.

ARMA -

Autoregressive (AR) Moving Average (MA)

```
result_azn.summary()
```

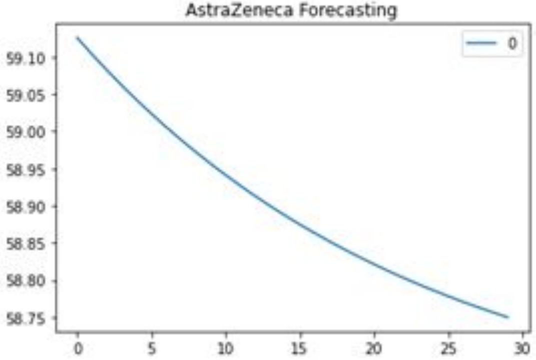
ARMA Model Results

Dep. Variable:	y	**No. Observations:**	2939
Model:	ARMA(2, 2)	**Log Likelihood**	-2412.593
Method:	css-mle	**S.D. of innovations**	0.549
Date:	Mon, 06 Sep 2021	**AIC**	4837.187
Time:	22:50:10	**BIC**	4873.102
Sample:	0	**HQIC**	4850.119

	coef	std err	z	P>|z|	[0.025	0.975]
const	69.5103	nan	nan	nan	nan	nan
ar.L1.y	1.9589	1.33e-05	1.48e+05	0.000	1.959	1.959
ar.L2.y	-0.9589	6.5e-06	-1.48e+05	0.000	-0.959	-0.959
ma.L1.y	-1.0186	0.019	-54.324	0.000	-1.055	-0.982
ma.L2.y	0.0444	0.019	2.357	0.018	0.007	0.081

```
# Forecasting the future AstraZeneca (AZN).
pd.DataFrame(result_azn.forecast(steps=30)[0]).plot(title="AstraZeneca Forecasting")
```

<AxesSubplot:title={'center':'AstraZeneca Forecasting'}>



ARIMA -

Autoregressive (AR) Integrated Moving Average (MA)

```python
arima_results.summary()
```

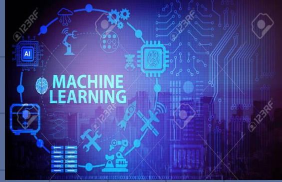
ARIMA Model Results

Dep. Variable:	D.Close	**No. Observations:**	2938
Model:	ARIMA(5, 1, 1)	**Log Likelihood**	-2413.684
Method:	css-mle	**S.D. of innovations**	0.550
Date:	Mon, 06 Sep 2021	**AIC**	4843.368
Time:	15:38:38	**BIC**	4891.252
Sample:	1	**HQIC**	4860.610

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0120	0.009	1.287	0.198	-0.006	0.030
ar.L1.D.Close	-0.9076	0.069	-13.125	0.000	-1.043	-0.772
ar.L2.D.Close	-0.0457	0.025	-1.795	0.073	-0.096	0.004
ar.L3.D.Close	0.0073	0.025	0.294	0.769	-0.042	0.056
ar.L4.D.Close	-0.0560	0.025	-2.244	0.025	-0.105	-0.007
ar.L5.D.Close	0.0018	0.022	0.083	0.933	-0.041	0.044
ma.L1.D.Close	0.8345	0.067	12.515	0.000	0.704	0.965

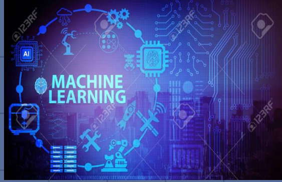
```python
pd.DataFrame(arima_results.forecast(steps=5)[0]).plot(title="5 Day Futures Price Forecast")
plt.margins(x=0)
```

Processing Data - Key Indicators



- **SMA - Simple Moving Average**: Average of a range in determined periods of time
- **EMA - Exponential Moving Average**: Weighted moving average that provides a preponderance to most recent data
- **RSI - Relative Strength Index:** Measures the magnitude of recent price changes to assess overbought or oversold conditions of certain stock. It’s a momentum indicator and usually values higher than 70 are considered indication of the stock being overbought.
- **MACD - Moving Average Convergence Divergence:** Shows the relationship between two moving averages and it’s calculated by subtracting the 26-period exponential moving average (EMA) from 12-period EMA.

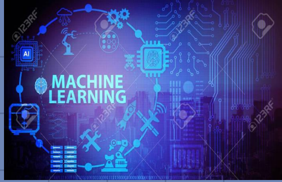
Moving Average Indicators

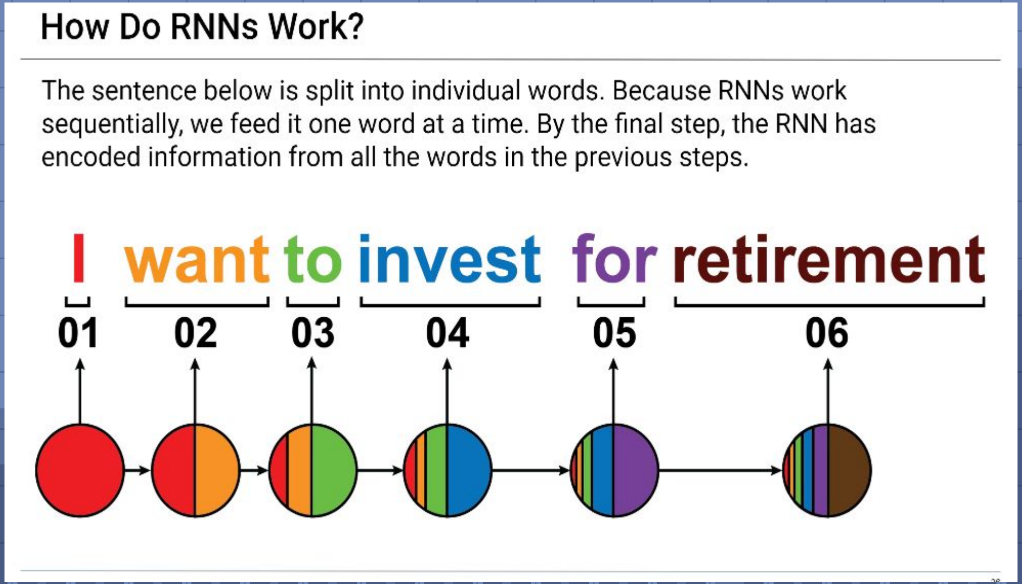


- SMA_5 (Simple Moving Average with a 5 day window)
- SMA_10 (Simple Moving Average with a 10 day window)
- SMA_15 (Simple Moving Average with a 15 day window)
- SMA_30 (Simple Moving Average with a 30 day window)
- EMA_9 (Exponential Moving Average with a 9 day window)

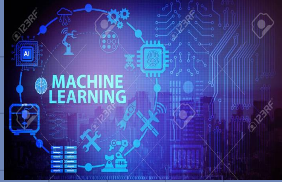
MODEL PROCESS: LSTM

Long Short-Term Memory





LSTM:



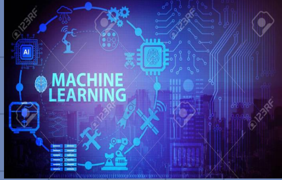
Introduction :

Time-series forecasting models are models that are capable of predicting future values based on previously observed values. (Time series is a sequence of observations taken sequentially in time)

Model :

- LSTM is an artificial recurrent neural network (RNN) architecture used in deep learning. It can process both single data points and also entire sequences of data points (ex. Speech / video etc)
- LSTMs are very powerful in sequence prediction problems because they are able to store past information. This is important in our case study because the previous price of a stock is crucial in predicting its future price.

LSTM:



Working the Model using Python:

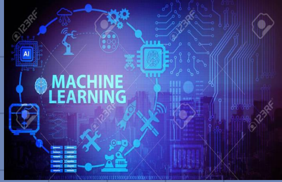
We built a multi-layer LSTM recurrent neural network to predict the last value in a sequence of values, in this case AstraZeneca stock price.

Getting the historical stock price Data :

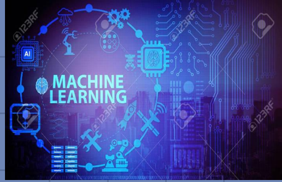
We used Yahoo Finance to get the historical data of AstraZeneca (ticker symbol : AZN)
https://ca.finance.yahoo.com/quote/AZN/

- Modules used :
  - pandas, numpy, yfinance, datetime
  - sklearn, statsmodel, ta, tensorflow
  - matplotlib, plotly

LSTM:

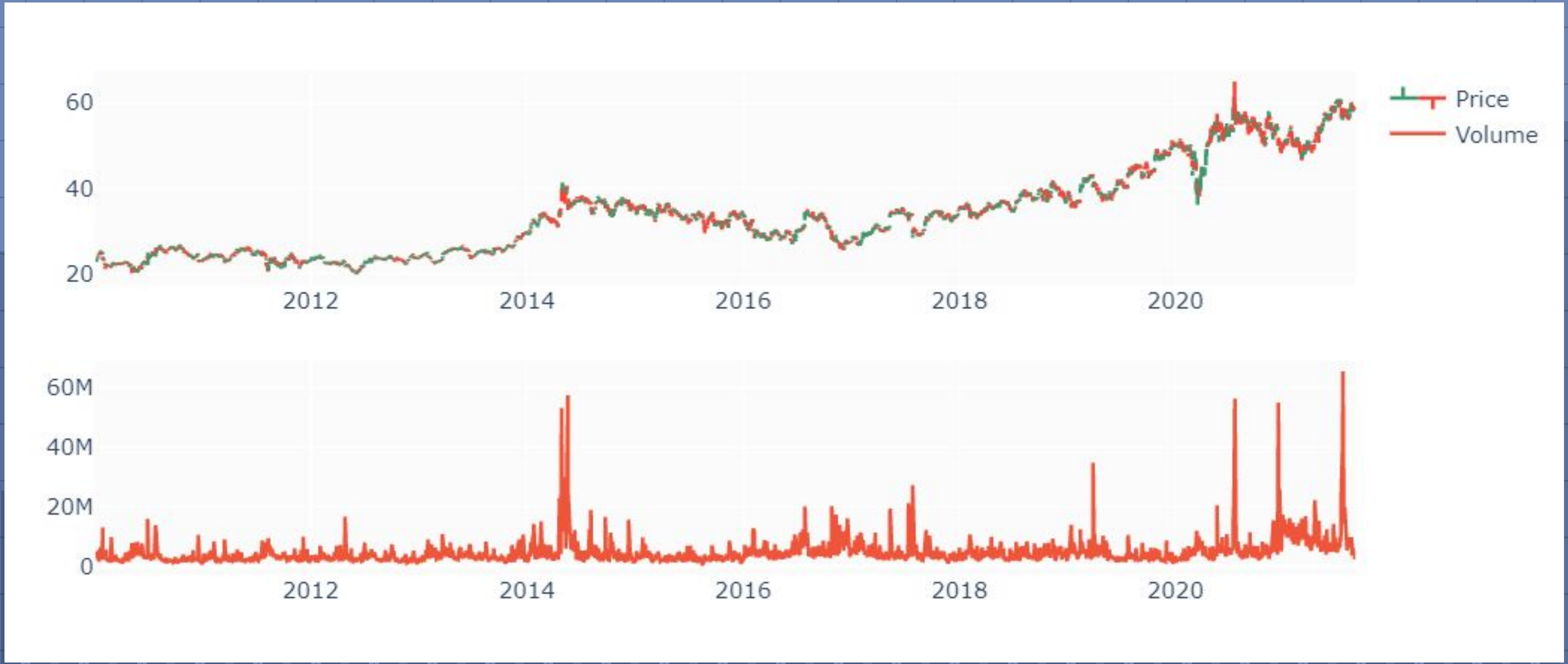


- <u>Functions created :</u>
  - plot_ohlc : To visualize the Open, High, Low, Close value of the stock
  - plot_decomposed_close_data : To visualize close data in decomposed form of Observed, Trend, Seasonal, Residual
  - plot_moving_averages : To visualize Moving Average
  - plot_RSI : To visualize Relative Strength Index
  - plot_MACD : To visualize Moving Average Convergence Divergence
  - plot_predictions : To visualize the predictions
  - get_moving_averages : To calculate the Moving Average
  - get_RSI : To calculate the Relative Strength Index
  - get_MACD : To calculate the Moving Average Convergence Divergence



LSTM:

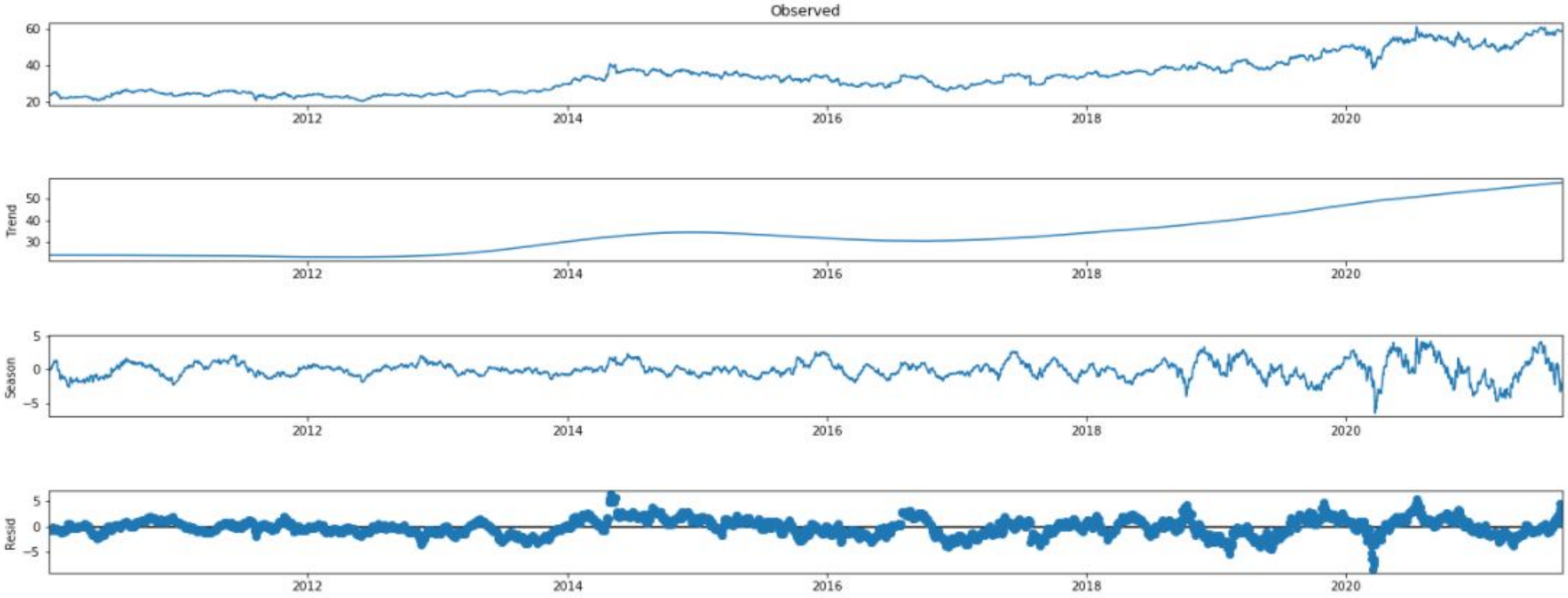
Loading and Inspecting Data : AZN (Price and Volume)

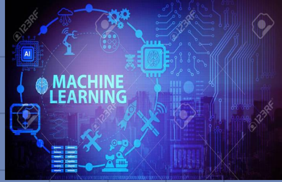


LSTM:

Decomposed data

```
plot_decomposed_close_data(azn_df)
```





LSTM:

Calculating Indicators :

Moving Average :

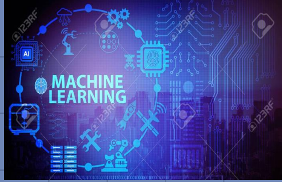
```
df_sma = SMAIndicator(close=df['Close'], window=5)
df['SMA_5'] = df_sma.sma_indicator()
```

RSI :

```
df_rsi = RSIIndicator(close=df['Close'], window=14)
df['RSI'] = df_rsi.rsi()
```

MACD:

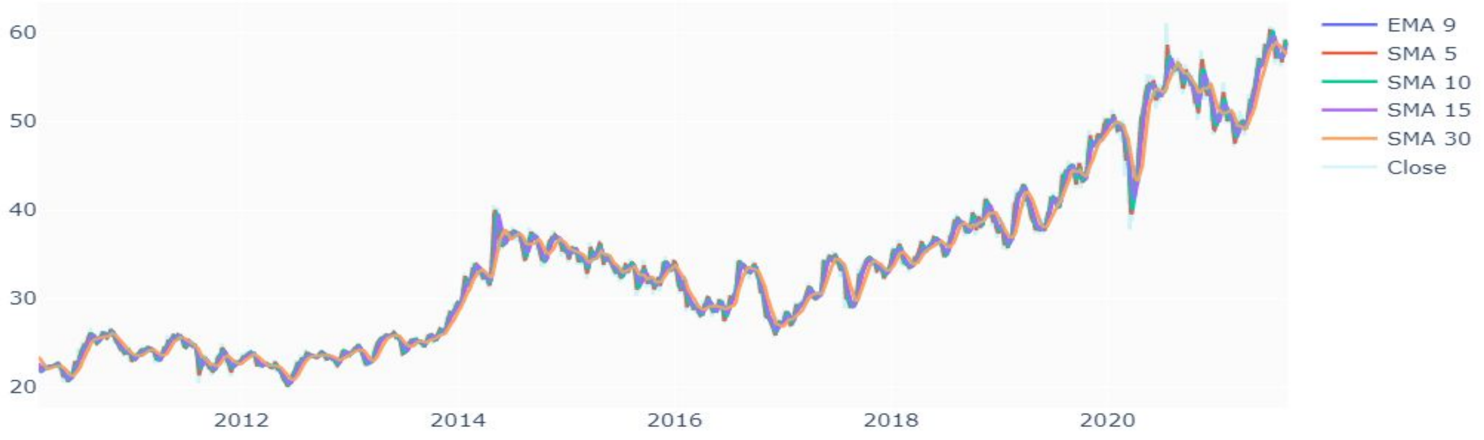
```
macd = MACD(close=df['Close'])
df['MACD'] = macd.macd()
df['MACD Signal'] = macd.macd_signal()
```



LSTM:

Moving Average :

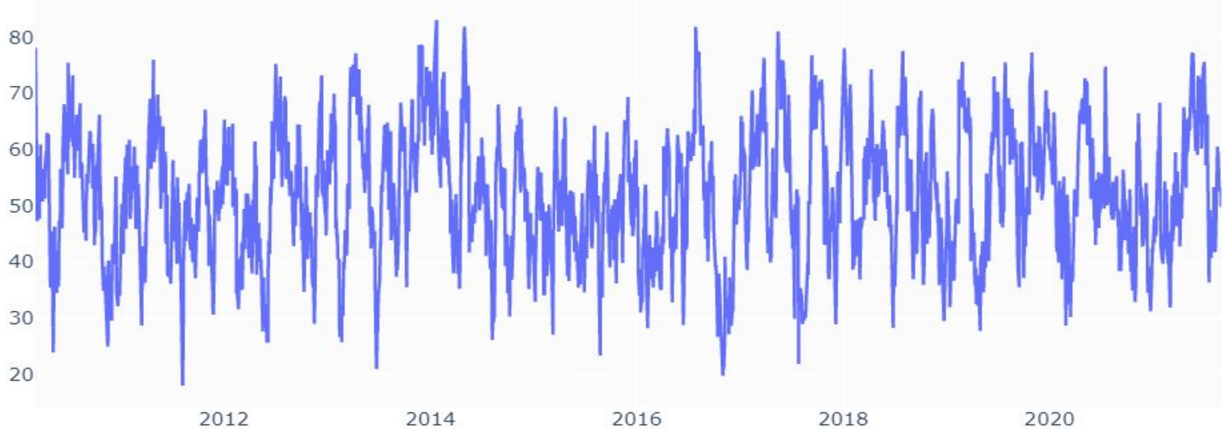
```
plot_moving_averages(azn_df)
```

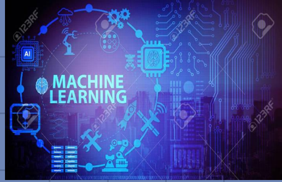


LSTM:

Relative Strength Index :

```
plot_RSI(azn_df)
```

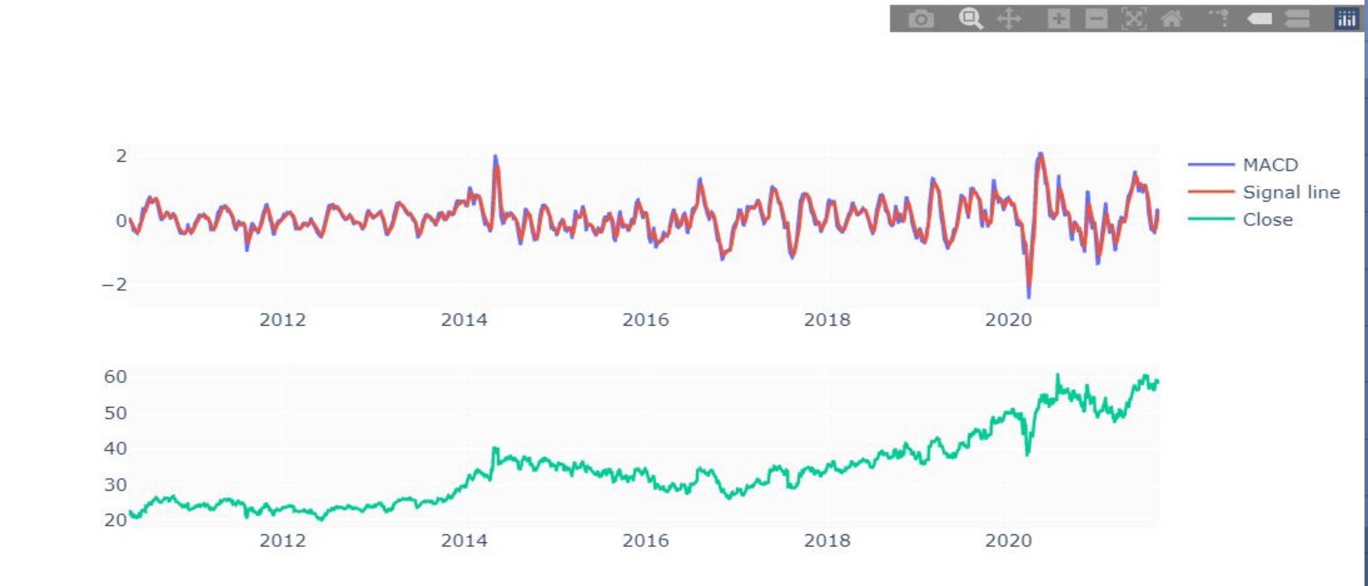


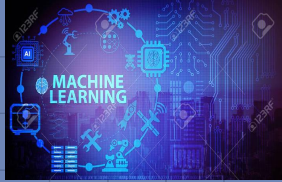


LSTM:

MACD / MACD signal :

```
plot_MACD(azn_df)
```



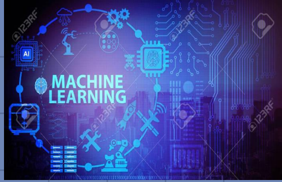


LSTM:

Sample data from the dataframe :

```
azn_df.head()
```

Date	Open	High	Low	Close	Adj Close	Volume	SMA_5	SMA_10	SMA_15	SMA_30	EMA_9	RSI	MACD	MACD Signal
2010-04-22	22.475000	22.485001	22.309999	22.565001	13.714867	3814000	22.657000	22.6570	22.570334	22.427000	22.616643	48.914873	0.058974	0.052132
2010-04-23	22.285000	22.580000	22.250000	22.455000	13.791260	1654600	22.637000	22.6570	22.576334	22.446000	22.606315	52.335118	0.051968	0.052099
2010-04-26	22.545000	22.605000	22.434999	21.790001	13.724032	1496000	22.605000	22.6380	22.577667	22.455334	22.576052	49.212574	0.037111	0.049102
2010-04-27	21.959999	22.075001	21.745001	21.840000	13.317597	4086200	22.390001	22.5530	22.542667	22.443167	22.418842	35.444464	-0.028000	0.033681
2010-04-28	21.740000	21.915001	21.500000	22.170000	13.348156	4230400	22.218000	22.4725	22.503000	22.431334	22.303073	36.874446	-0.074705	0.012004



LSTM:

Data Split :

Splitting the data frame into Training data and Test data using MinMaxScaler(). We split the DataFrame into 90 : 10 ratio for Training and Testing during the initial trial.

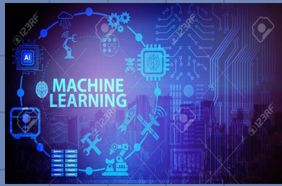
```
df_train, df_test = train_test_split(azn_df, train_size=0.9, shuffle=False)
```

Once the model was in working condition, we split the DataFrame with 2 years of training data and the next 15 months as testing data (from 2010 to 2021).

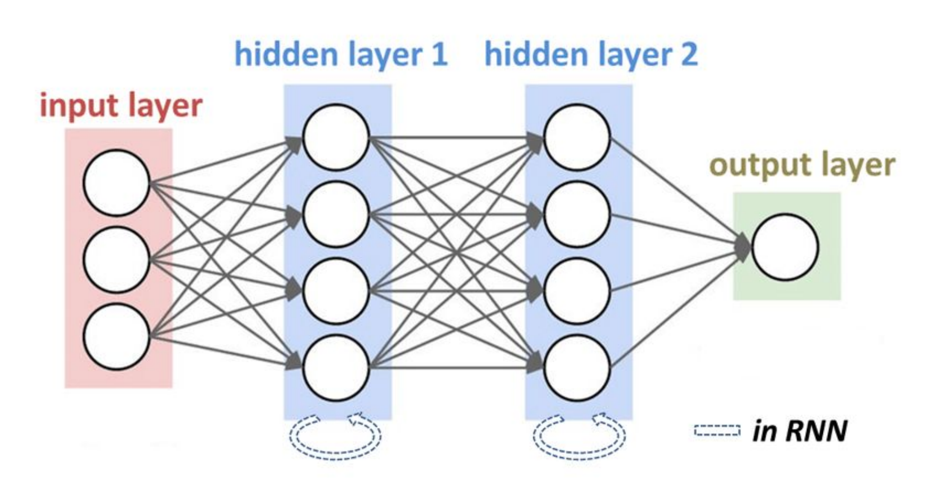
Preprocessing :

```
scaler = MinMaxScaler()

scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

scaler.fit(y_train)
y_train = scaler.transform(y_train)
y_test = scaler.transform(y_test)
```

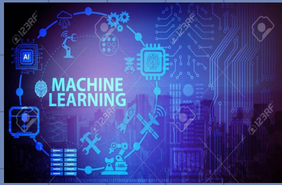
LSTM: Model with 50 neurons, 3 layers and, 10% dropout fraction.




```
model_1 = Sequential()
number_units = 50
dropout_fraction = 0.1

# Layer 1
model_1.add(LSTM(
   units = number_units,
   return_sequences = True,
   input_shape = (x_train.shape[1],1))
      )
model_1.add(Dropout(dropout_fraction))

# Layer 2
model_1.add(LSTM(
   units = number_units,
   return_sequences = True,
      ))
model_1.add(Dropout(dropout_fraction))

# Layer 3
model_1.add(LSTM(
   units = number_units,
   return_sequences = False,
      ))
model_1.add(Dropout(dropout_fraction))

model_1.add(Dense(1))
```



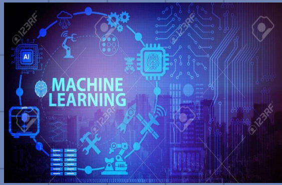
LSTM: Model



Compile : Optimizer as “adam” and loss as “mean_squared_error”

```
model_1.compile(optimizer="adam", loss="mean_squared_error")
model_1.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 13, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 13, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 13, 50)            20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

LSTM: Model



Evaluation of the model : Mean Squared Error

```
model_1.evaluate(x_test, y_test)
```

```
12/12 [==============================] - 1s 4ms/step - loss: 0.0024
0.0024029279593378305
```

```
y_pred = model_1.predict(x_test)
print(f'mean_squared_error = {mean_squared_error(y_test, y_pred)}')
```

```
mean_squared_error = 0.0024029282649802655
```

Formula

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

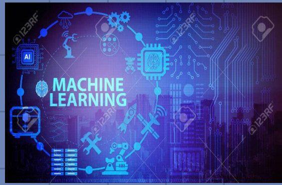
MSE = mean squared error

n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

LSTM: Model



Predicted values :

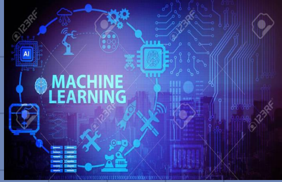
```
df_test_2014.head()
```

Date	Open	High	Low	Close	Adj Close	Volume	SMA_5	SMA_10	SMA_15	SMA_30	EMA_9	RSI	MACD	MACD Signal	Predicted Close
2012-04-23	22.820000	22.825001	22.575001	22.910000	15.563806	3056600	22.780	22.5070	22.437000	22.503667	22.633376	54.203175	0.041000	-0.050273	22.765718
2012-04-24	22.815001	22.995001	22.780001	22.955000	15.659497	2126600	22.835	22.6005	22.455000	22.513500	22.688701	56.478258	0.064046	-0.027409	22.863758
2012-04-25	22.730000	22.975000	22.705000	21.680000	15.690253	3224800	22.888	22.6750	22.483000	22.519500	22.741961	57.214017	0.084962	-0.004935	22.887241
2012-04-26	21.650000	21.785000	21.500000	21.775000	14.818763	16581600	22.685	22.5915	22.449333	22.493167	22.529569	37.744234	-0.001328	-0.004213	22.230415
2012-04-27	21.709999	21.844999	21.605000	21.950001	14.883699	4449400	22.418	22.5580	22.427667	22.473333	22.378655	39.398989	-0.061341	-0.015639	21.946381

LSTM: Model

Visualization of Predicted values :

LSTM: Model

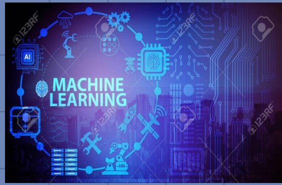


Next day’s Predicted value :

```
inp = np.array(inp)
inp = inp.reshape((inp.shape[0], inp.shape[1], 1))
tomorrow = model_1.predict(inp)
scaler.inverse_transform(tomorrow)
```

```
array([[56.93612]], dtype=float32)
```

XGBooster Model




```
# Preprocessing

scaler = MinMaxScaler()

scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

scaler.fit(y_train)
y_train = scaler.transform(y_train)
y_test = scaler.transform(y_test)
```

```
model_3_xgb = xgb.XGBRegressor(n_estimators=100, objective='reg:squarederror')
model_3_xgb.fit(x_train, y_train)
```

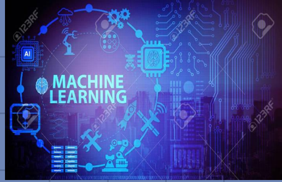
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=16, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

```
y_pred_xgb_3 = model_3_xgb.predict(x_test)
print(f'mean_squared_error = {mean_squared_error(y_test, y_pred_xgb_3)}')
```

```
mean_squared_error = 0.12183479125312457
```

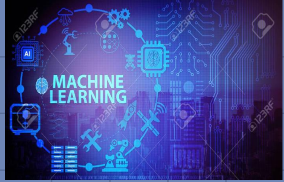
```
df_test_2020['Predicted Close XGB'] = scaler.inverse_transform(y_pred_xgb_3.reshape(-1,1))
```

Observations



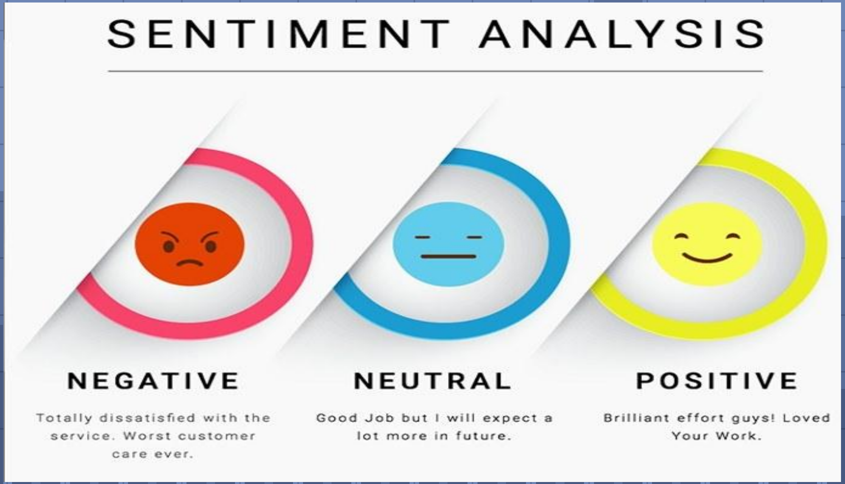
- We have also observed that in high volatile stocks the model gives a less accurate result, to overcome this we had to trial and error the model using different time frames and prediction windows.

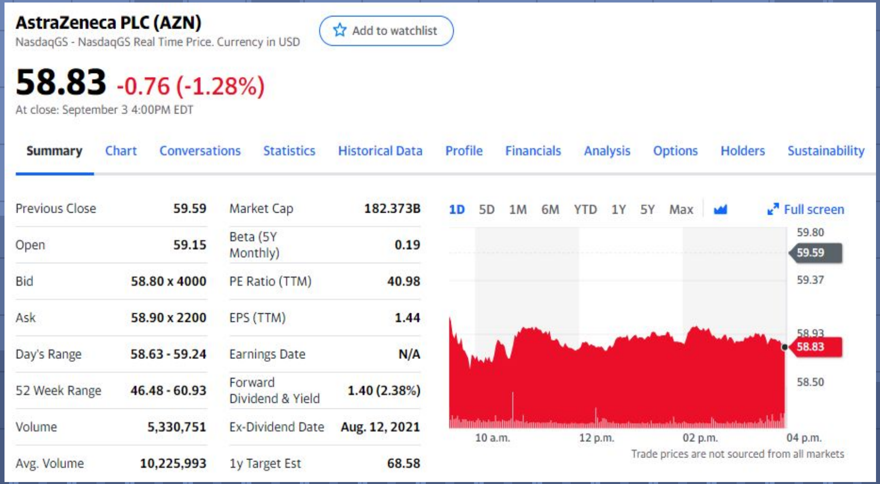
- We also noticed that using technical indicators from the TA library improved the accuracy of results by approximately 30%.



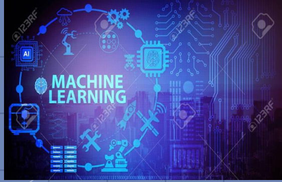
FURTHER IMPROVEMENTS

- This is a base stock prediction model to be built upon, we can add new tools to it in order to perfect it, such as Sentiment analysis and Fundamental analysis to achieve even better results.





Expectations



- keep it realistic, stock prediction models are not perfect, it needs to be re train and re tested as new information and events occurs, past performance does not guarantee future returns.
- Even the biggest Quant firms in the world have to constantly adjust their models to keep good returns.

Bloomberg

ıtes on the news affecting the global economy. **Enable Notifications.**

Markets

Renaissance Suffers $11 Billion Exodus With Meager Quant Returns

By Miles Weiss and Hema Parmar
June 18, 2021, 2:02 PM EDT

THANKS!

Any questions?

Let's start the Q & A.