Anthony
Benny
Venus

DEEP LEARNING PRESENTATION

Stock Price Prediction Project

Objective

Use basic criteria and additional technical indicators (in 5-mins-interval) to predict AAPL's stock price after 30 minutes.

Data Input

01

- DATE
- OPEN
- CLOSE
- HIGH
- LOW
- VOLUME

02

TECHNICAL
INDICATORS

03

CRITERIAL
&
BASIC REQUIREMENT

04

X:THOSE CRITERIAL
y: Closed price after 30 minutes

05

TRAIN THE MODEL

Data Collection

Almost all python-supporting API charge

Due to the huge amount of data when it comes to the data freq. in 5-mins-interval, web-scrapping would not be an efficient option to obtain the massive dataset.

Turn to a Google Sheet based API

Leveraging the API service from the data provider "marketdata.app" through Google Sheet but under a limitation of getting the data in max. range of 2 months each time.

Use code to connect with Google Sheet and automate the data obtaining process

Use a simple for-loop to update the formula in Google Sheet to obtain historical data since 2017.

PERIOD of DATA

2017/01/03 0830 - 2022/06/24 1555
total : 106680 entries
train size : 78235 (2017/01/03 - 2020/12/31)
test size : 23445 (2021/01/04 - 2022/06/24)

Original Data set

	Date	Open	High	Low	Close	Volume
0	03/01/2017 08:30	116.100	116.2000	116.100	116.1700	8854
1	03/01/2017 08:35	116.170	116.2300	116.170	116.2200	3758
2	03/01/2017 08:40	116.230	116.2400	116.200	116.2000	11902
3	03/01/2017 08:45	116.210	116.2200	116.210	116.2200	268
4	03/01/2017 08:50	116.220	116.2500	116.070	116.1900	17702
...	...	...	...	...	...	...
106675	24/06/2022 15:35	140.205	140.4172	140.180	140.4144	745150
106676	24/06/2022 15:40	140.420	140.6000	140.410	140.5800	1058976
106677	24/06/2022 15:45	140.575	140.6900	140.420	140.4750	1065038
106678	24/06/2022 15:50	140.470	141.2100	140.465	141.1450	2939247
106679	24/06/2022 15:55	141.150	141.9100	141.090	141.8100	5400234

106680 rows × 6 columns

Data checking

Basics

Date	106680 non-null
Open	106680 non-null
High	106680 non-null
Low	106680 non-null
Close	106680 non-null
Volume	106680 non-null
year	106680 non-null
month	106680 non-null
day	106680 non-null
hour	106680 non-null
minute	106680 non-null

Technical indicators

sma10	106671 non-null
sma20	106661 non-null
sma50	106631 non-null
sma120	106561 non-null
SMA_30	106651 non-null
SMA_50	106631 non-null
SMA_150	106531 non-null
SMA_200	106481 non-null
SMA_slope_200	106462 non-null
SMA_slope_30	106632 non-null

Criteria

Criteria11	106680
Criteria12	106680
Criteria13	106680
Criteria14	106680
Criteria15	106680
Criteria18	106680
Criteria19	106680
Criteria110	106680
Criteria111	106680
Criteria112	106680
Criteria113	106680
Criteria114	106680

Dealing with null data

```
data = data.iloc[219:,:]
```

Separating date in to diff. col.

```
data["Date"] = pd.to_datetime(data['Date'],format="%d/%m/%Y %H:%M")
data['year'] = data["Date"].dt.year
```

Model creation

Splitng the data

```
# number of training record
no_train_records = 78235

# X_train
df_train = data.iloc[:no_train_records,1:33]

# y_train
y_tr = data.iloc[:no_train_records,33:]

# X_test
df_test = data.iloc[no_train_records:,1:33]

# y_test
y_te = data.iloc[no_train_records:,33:]
```

Feature scaling

```
from sklearn.preprocessing import StandardScaler

sc1, sc2 = StandardScaler(), StandardScaler()

df_train_scaled = sc1.fit_transform(df_train)
print(df_train_scaled.shape)

df_train_scaled_y = sc2.fit_transform(y_tr['price after 30mins'])
print(df_train_scaled_y.shape)
```

```
(78235, 32)
(78235, 1)
```

Shifting the closing price

```
data['price after 30mins'] = data['Close'].shift(periods = -6)
```

Creating a data structure

```
# Creating a data structure with 14 timesteps and 1 output
steps = 14

X_train = []
y_train = []

for i in range(steps,no_train_records):
    X_train.append(df_train_scaled[i-steps:i])
    y_train.append(df_train_scaled_y[i][0])

X_train, y_train = np.array(X_train), np.array(y_train)
print("X_train", X_train.shape)
print("y_train", y_train.shape)
```

```
X_train (78221, 14, 32)
y_train (78221,)
```

Model creation

```
# Initialising the RNN
regressor = Sequential()
# units : 50, 100, 200, 256, 512, 1024
# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train_shape.shape[1], 32)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'Adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train_shape, y_train, epochs = 50, batch_size = 64)
```

GET RESULTS

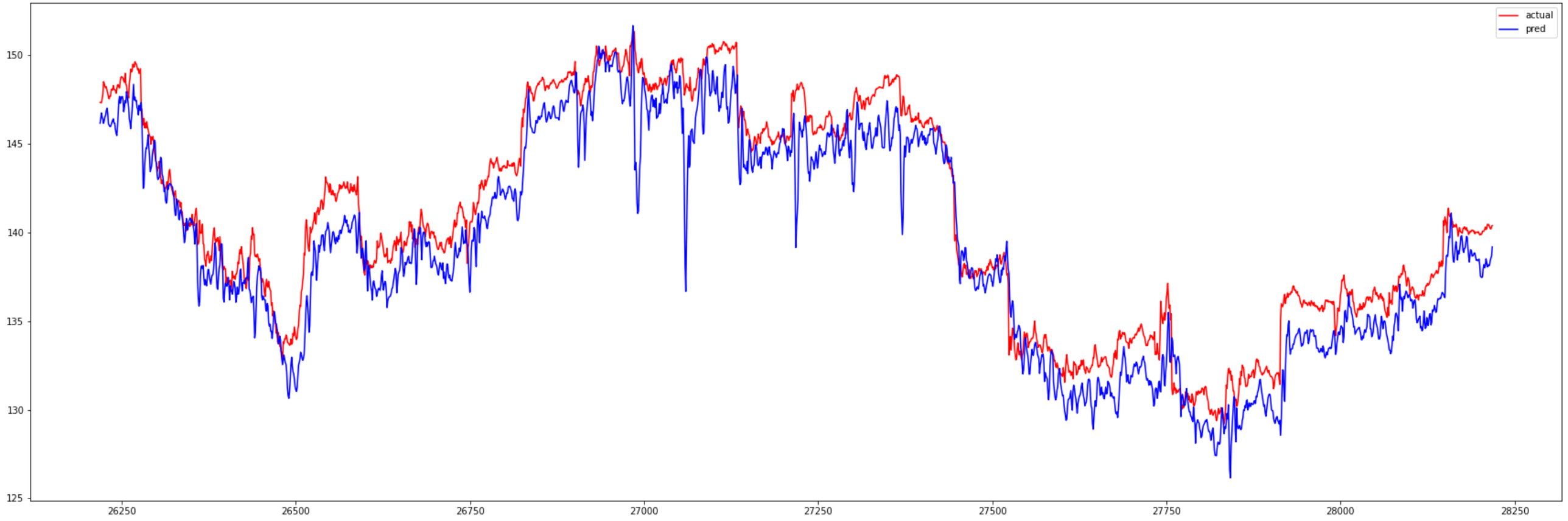
Generate the line chart to visualization the result

```
plt.figure(figsize=(30,10))

plt.plot(full_final["Close"][-2000:],label="actual",  color="red")
plt.plot(full_final["final_pred"][-2000:],label="pred",  color="blue")

plt.legend()
```

```
plt.figure(figsize=(30,10))

plt.plot(full_final["diff"][-2000:],label="diff",  color="green")
plt.legend()
```

RESULTS

prediction vs actual stock price (last ~26 days))

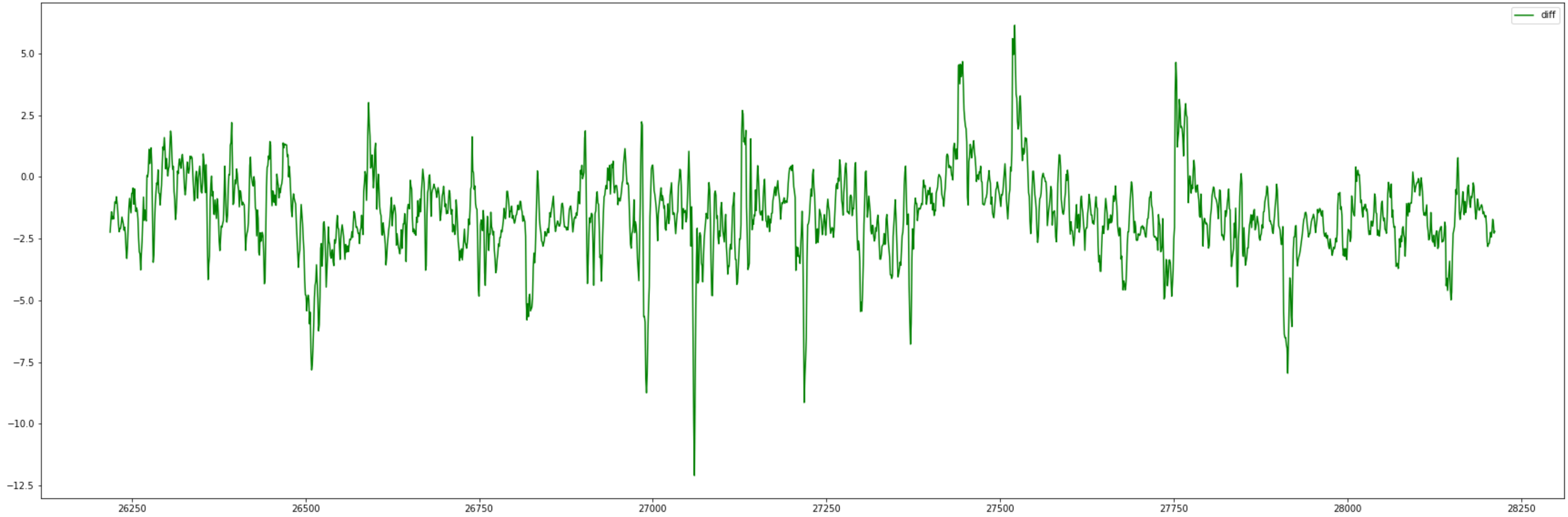


RESULTS

prediction vs actual stock price (last 2 days)

RESULTS

differences between prediction and actual (last ~26 days))



RESULTS

differences between prediction and actual (last 2 days))

Computation Resource

Running the LSTM model by using google cloud GPU is significantly faster than CPU (almost 4 times)

Time

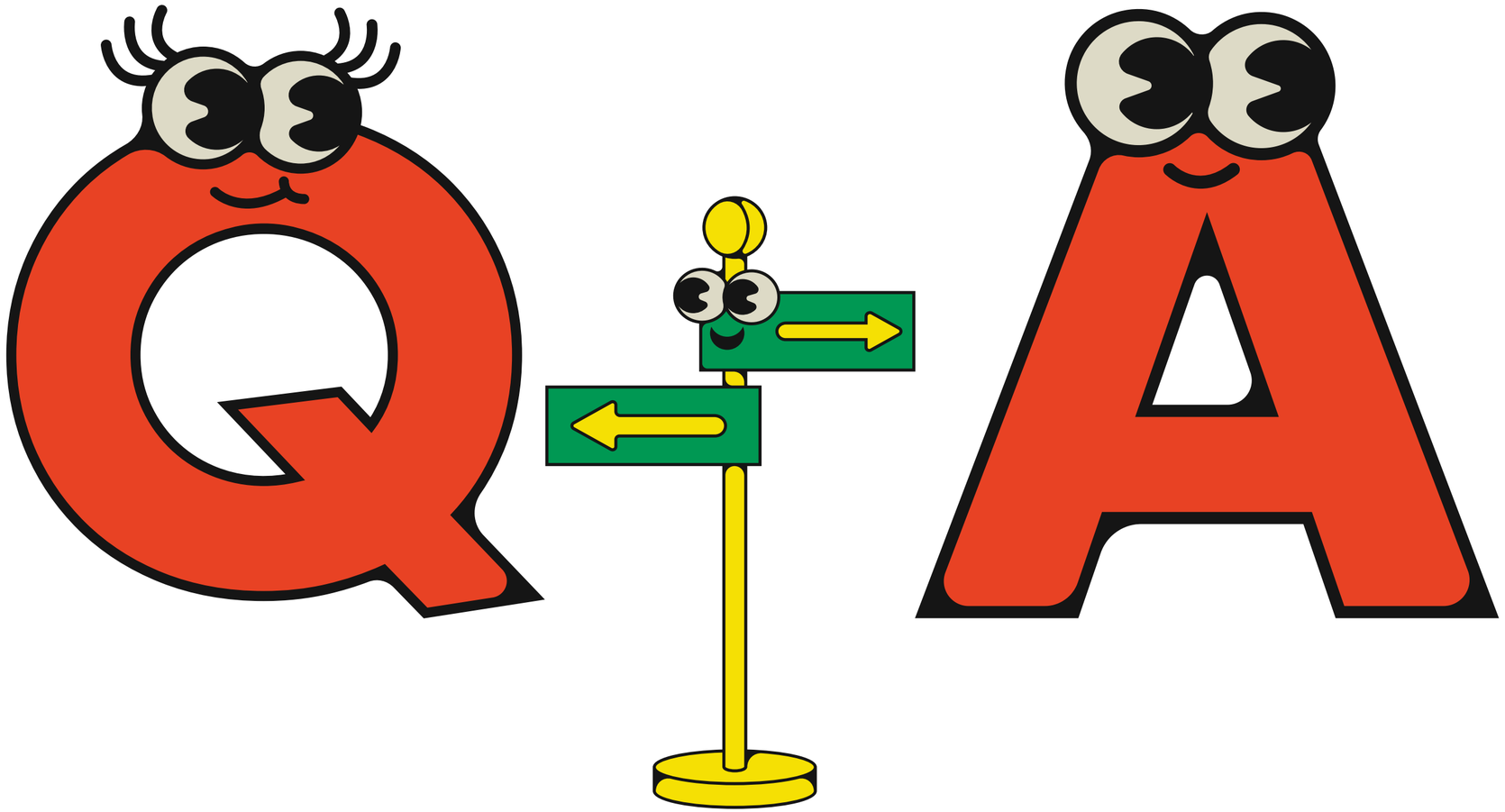
Using CPU to fit the data into the model is time consuming and we cannot leave the page due to the colab policy. (Long idle time may shut colab down)

Data volume & Tech. Indicators Tuning

Choosing the technical indicators required domain knowledge.

Model Layers & Parameter Tuning

The no. of layers, LSTM units in different layers, the percentage value of Dropout , etc.


Q
A