Live Stock Price Prediction Using Deep Learning

Mahnoor Javaid

LIVE STOCK PRICE PREDICTION USING DEEP LEARNING

Introduction

The financial market is highly dynamic and predicting stock prices is a complex and challenging task. Accurate stock price predictions can provide significant advantages in making informed investment decisions. In this project, we employ deep learning techniques to predict the stock prices of GAIL (India) Limited using historical data obtained from Yahoo Finance. The project demonstrates the application of Long Short-Term Memory (LSTM) networks for time series forecasting.

2. Purpose

The primary purpose of this project is to utilize advanced machine learning techniques to forecast future stock prices based on historical data. By leveraging LSTM networks, we aim to capture the underlying patterns and trends in the stock market data to make accurate predictions. This can aid investors and traders in making better-informed decisions, thereby potentially increasing returns and reducing risks.

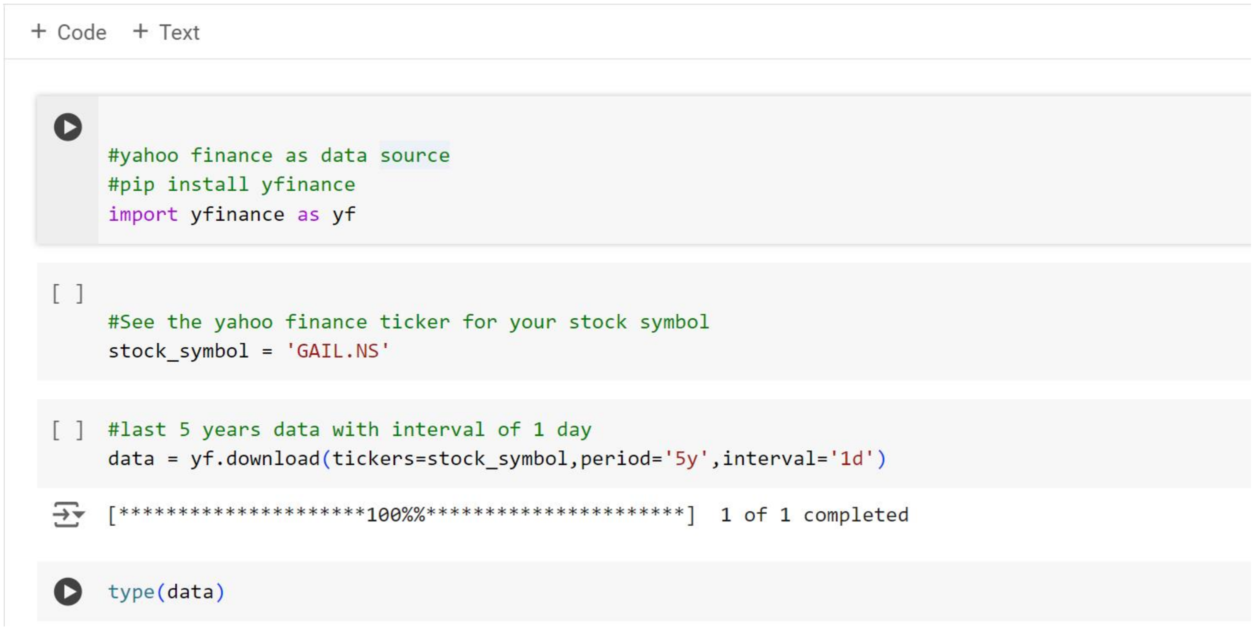
3. Project Goal

The goal of this project is to build a robust predictive model using LSTM networks to forecast the 'Open' prices of GAIL stock for the next 30 days. The project involves data preprocessing, model training, and evaluation to ensure the accuracy and reliability of the predictions. The specific objectives include:

- Downloading and preprocessing historical stock price data.
- Normalizing the data for better model performance.
- Splitting the data into training and testing sets.
- Building and training the LSTM model.
- Making future price predictions and visualizing the results.

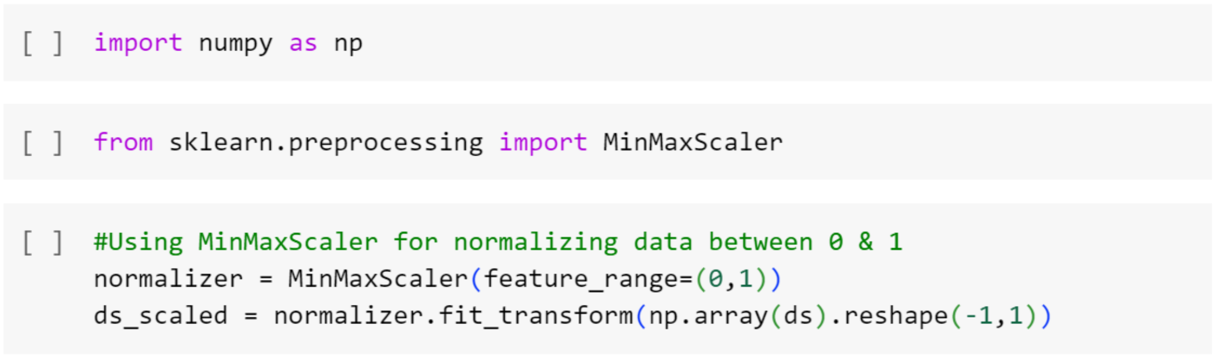
4. Models Used and Techniques

4.1 Data Collection and Preprocessing


```
+ Code   + Text

#yahoo finance as data source
#pip install yfinance
import yfinance as yf

[ ]
#See the yahoo finance ticker for your stock symbol
stock_symbol = 'GAIL.NS'

[ ] #last 5 years data with interval of 1 day
data = yf.download(tickers=stock_symbol,period='5y',interval='1d')

[*********************100%%**********************]  1 of 1 completed

type(data)
```

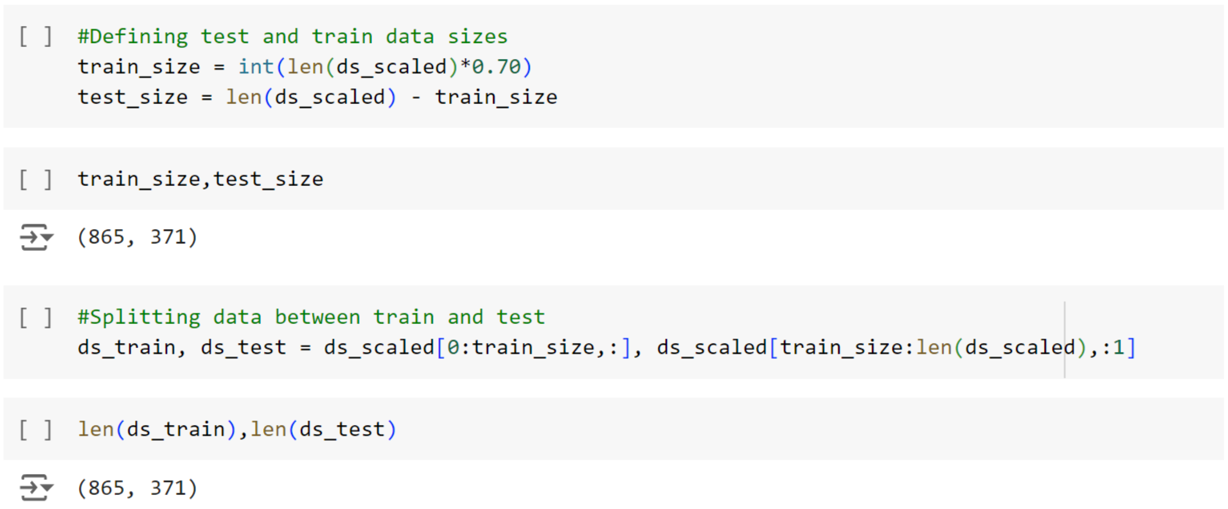
- **Data Collection:** We use the `yfinance` library to fetch historical stock price data for GAIL (India) Limited.
- **Preprocessing:** We focus on the 'Open' price column and plot the data to visualize the historical trends.

4.2 Normalization


```
[ ] import numpy as np

[ ] from sklearn.preprocessing import MinMaxScaler

[ ] #Using MinMaxScaler for normalizing data between 0 & 1
normalizer = MinMaxScaler(feature_range=(0,1))
ds_scaled = normalizer.fit_transform(np.array(ds).reshape(-1,1))
```

- **Normalization:** The 'Open' prices are normalized using `MinMaxScaler` to scale the data between 0 and 1. This step is crucial for improving the performance and convergence of the LSTM model.

4.3 Data Splitting



```
#Defining test and train data sizes
train_size = int(len(ds_scaled)*0.70)
test_size = len(ds_scaled) - train_size
```

```
train_size,test_size
```

```
(865, 371)
```

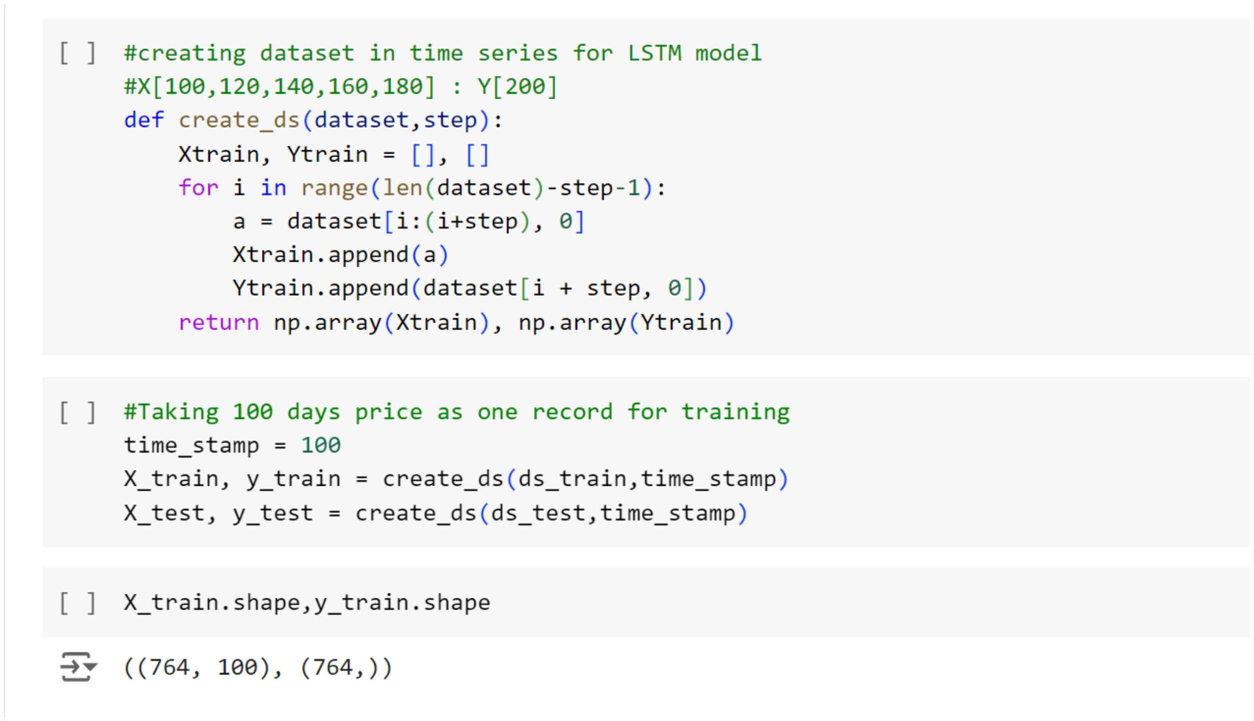
```
#Splitting data between train and test
ds_train, ds_test = ds_scaled[0:train_size,:], ds_scaled[train_size:len(ds_scaled),:1]
```

```
len(ds_train),len(ds_test)
```

```
(865, 371)
```

- **Data Splitting:** The normalized data is split into training (70%) and testing (30%) sets to train the model and evaluate its performance.

4.4 Creating Datasets for LSTM



```
#creating dataset in time series for LSTM model
#X[100,120,140,160,180] : Y[200]
def create_ds(dataset,step):
    Xtrain, Ytrain = [], []
    for i in range(len(dataset)-step-1):
        a = dataset[i:(i+step), 0]
        Xtrain.append(a)
        Ytrain.append(dataset[i + step, 0])
    return np.array(Xtrain), np.array(Ytrain)
```

```
#Taking 100 days price as one record for training
time_stamp = 100
X_train, y_train = create_ds(ds_train,time_stamp)
X_test, y_test = create_ds(ds_test,time_stamp)
```

```
X_train.shape,y_train.shape
```

```
((764, 100), (764,))
```

- **Creating Datasets:** The data is transformed into time series format suitable for LSTM. Each record consists of 100 days of prices used to predict the next day's price.

4.5 Building and Training the LSTM Model

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, LSTM
```

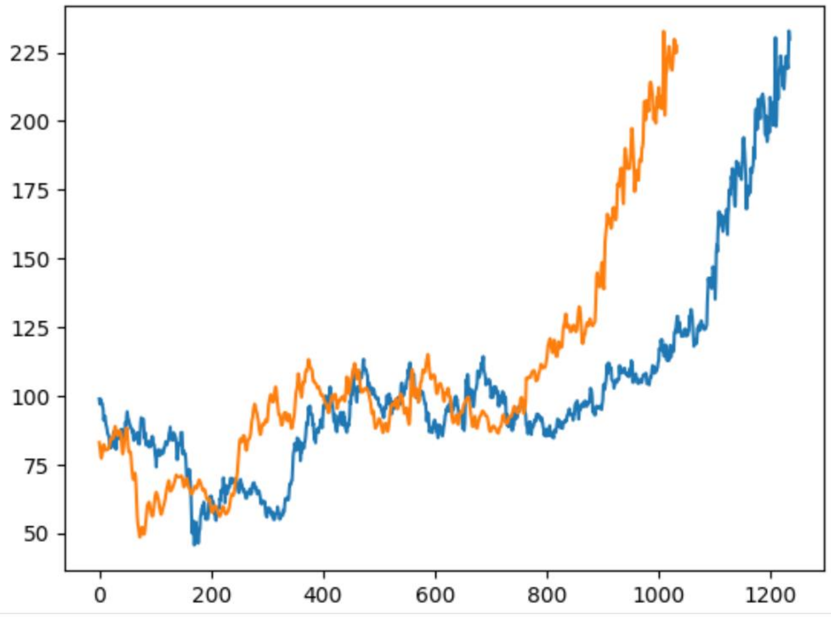
```
[ ]  #Creating LSTM model using keras
     model = Sequential()
     model.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1],1)))
     model.add(LSTM(units=50,return_sequences=True))
     model.add(LSTM(units=50))
     model.add(Dense(units=1,activation='linear'))
     model.summary()
```

- **Model Building:** We build an LSTM model using Keras with three LSTM layers and one dense layer for output. The model is compiled with the Adam optimizer and mean squared error loss function.
- **Model Training:** The model is trained on the training dataset for 100 epochs with a batch size of 64, and the training loss is monitored.

4.6 Model Evaluation and Prediction

```
[ ]  #Combining the predited data to create uniform data visualization
     plt.plot(normalizer.inverse_transform(ds_scaled))
     plt.plot(test)
```

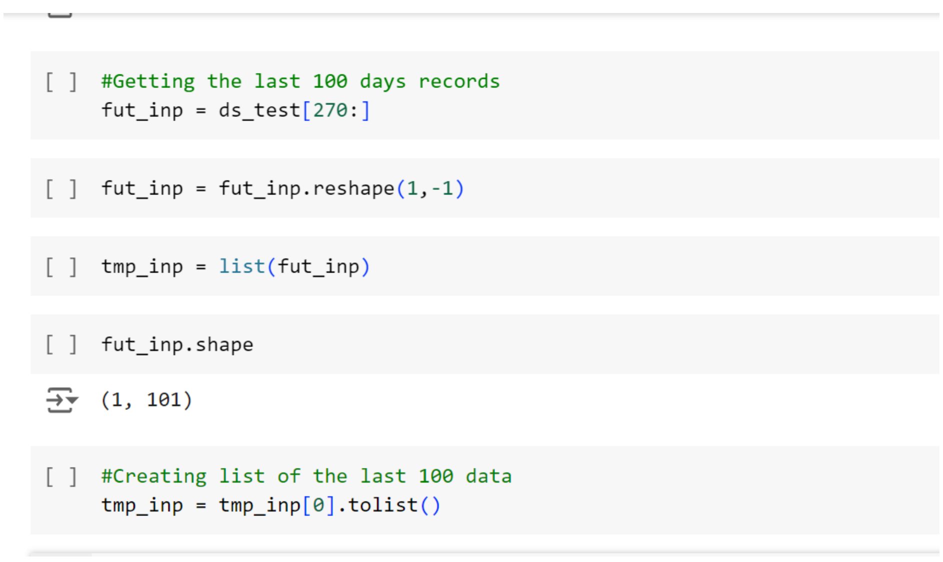
```
[<matplotlib.lines.Line2D at 0x7dbd824636d0>]
```



- **Loss Plotting:** The training loss is plotted to visualize the model's convergence.
- **Predictions:** The model makes predictions on both the training and testing datasets.
- **Inverse Transformation:** The predictions are transformed back to the original scale using `normalizer.inverse_transform` for better interpretability.

- **Comparison:** The actual and predicted prices are plotted to compare the model's performance visually.

4.7 Future Price Prediction



```
#Getting the last 100 days records
fut_inp = ds_test[270:]
```

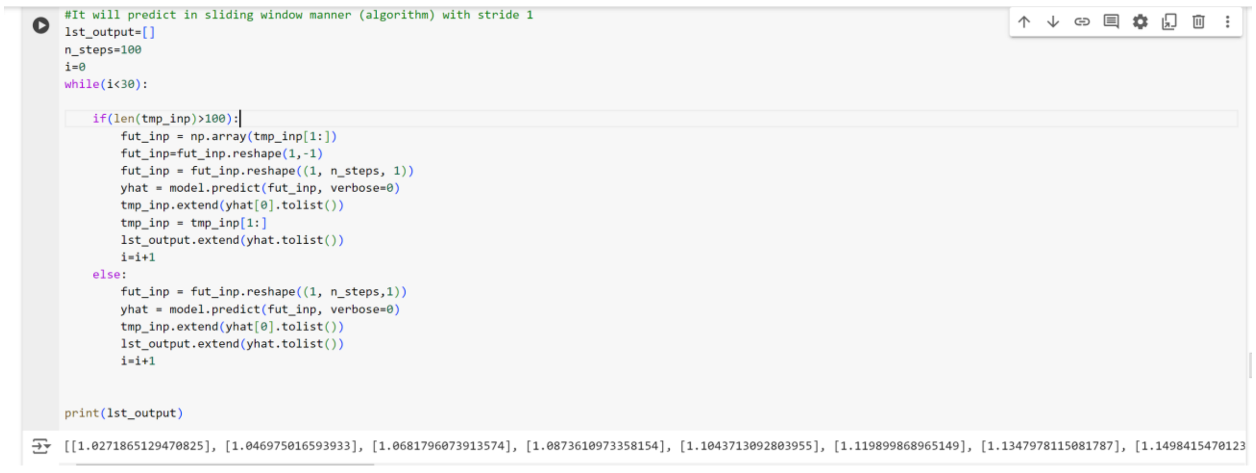
```
fut_inp = fut_inp.reshape(1,-1)
```

```
tmp_inp = list(fut_inp)
```

```
fut_inp.shape
```

```
(1, 101)
```

```
#Creating list of the last 100 data
tmp_inp = tmp_inp[0].tolist()
```



```
#It will predict in sliding window manner (algorithm) with stride 1
lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(tmp_inp)>100):
        fut_inp = np.array(tmp_inp[1:])
        fut_inp=fut_inp.reshape(1,-1)
        fut_inp = fut_inp.reshape((1, n_steps, 1))
        yhat = model.predict(fut_inp, verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        tmp_inp = tmp_inp[1:]
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        fut_inp = fut_inp.reshape((1, n_steps,1))
        yhat = model.predict(fut_inp, verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
[[1.0271865129470825], [1.046975016593933], [1.0681796073913574], [1.0873610973358154], [1.1043713092803955], [1.119899868965149], [1.1347978115081787], [1.1498415470123
```

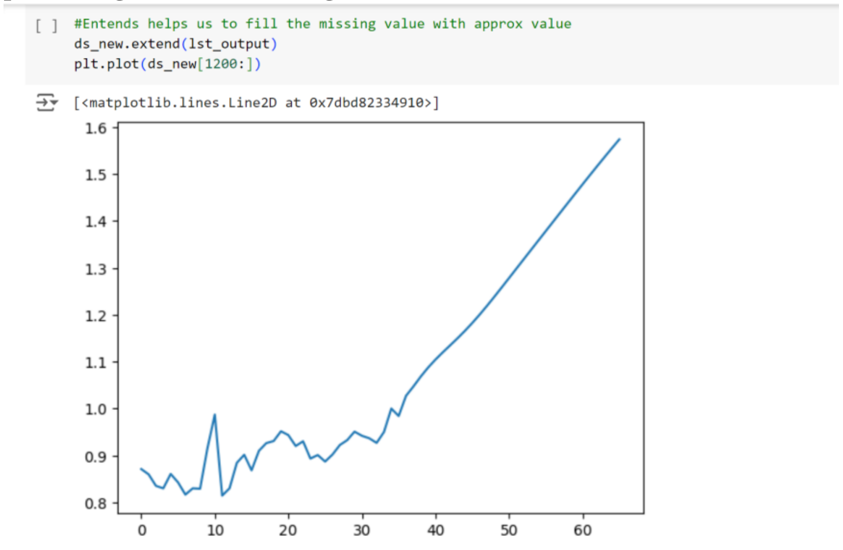
- **Future Prediction:** Using the last 100 days of test data, the model predicts the 'Open' prices for the next 30 days.
- **Visualization:** The final results, including the predicted future prices, are plotted for visualization.

5. Results

The LSTM model trained on historical stock prices of GAIL (India) Limited provides accurate predictions for both the training and testing datasets. The future price predictions for the next 30 days offer valuable insights into potential market movements. The project demonstrates the effectiveness of LSTM networks in capturing complex patterns in stock market data and making reliable predictions.

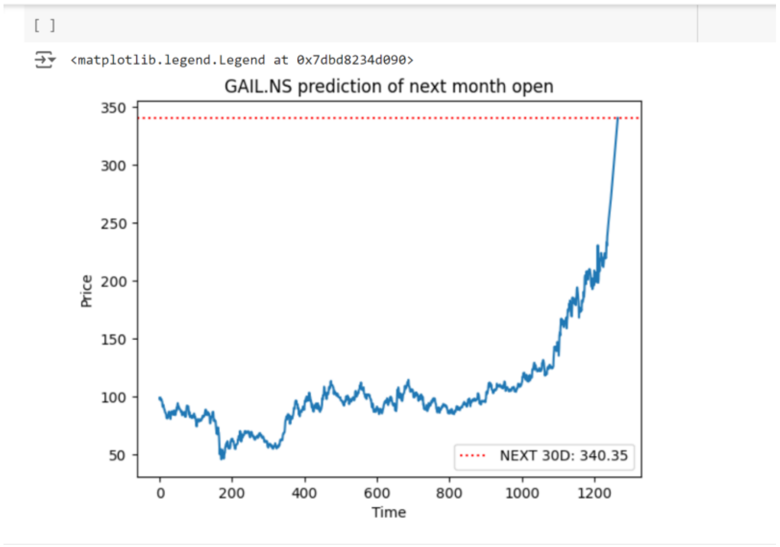
Key Results:

- **Training Loss:** The training loss decreases significantly, indicating effective learning by the model.
- **Prediction Accuracy:** The predicted prices closely follow the actual prices, showcasing the model's accuracy.
- **Future Prices:** The model successfully predicts the stock prices for the next 30 days, providing actionable insights.



```
[ ] #Entends helps us to fill the missing value with approx value
    ds_new.extend(lst_output)
    plt.plot(ds_new[1200:])
```

```
[<matplotlib.lines.Line2D at 0x7dbd82334910>]
```



```
[ ]
```

```
<matplotlib.legend.Legend at 0x7dbd8234d090>
```

Conclusion

This project successfully applies LSTM networks to predict stock prices based on historical data. The use of normalization, time series data creation, and LSTM model architecture contributes to the model's accuracy and reliability. The results highlight the potential of deep learning techniques in financial forecasting, aiding investors and traders in making informed decisions.