Beat the Financial Market

Presented by
Dreamers Team.

Mentor: Matthias Neumüller

Team

Luis Bravo
Santa Monica College
CS major

Izeth Torres
Cal State LA
CS major

Rieun “Lin” Kim
UIUC
CS major

Gayvalin “Tammy” Sujaritchai
Pasadena City College
CS major

Agenda

- **Motivation & Approach**
  - Data Preparation / Training
- **ML Model Comparison**
  - LSTM vs Bidirectional LSTM
  - GRU
  - RNN
- **Conclusion & Future Direction**
  - Learning Outcome

Pain point

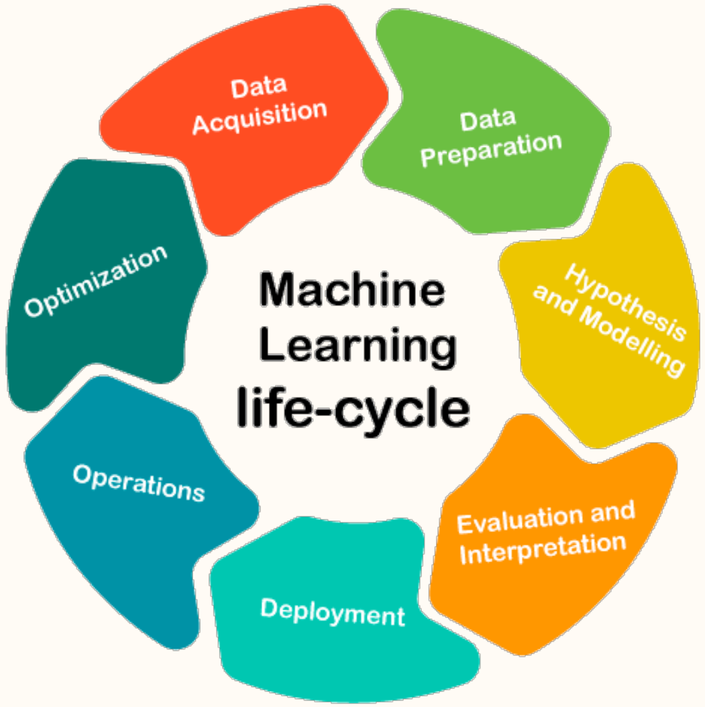
The financial market is constantly changing, how do we automate the process of predicting future changes based on past events?

Target user: financial advisor, investor, general public

Approach

Machine Learning Models (ML)

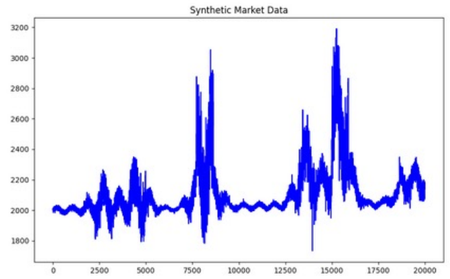
- Models
  - Long Short-Term Memory (LSTM)
  - Bidirectional LSTM
  - Gated recurrent unit (GRU)
  - Recurrent Neural Network (RNN)
- Binary Classification



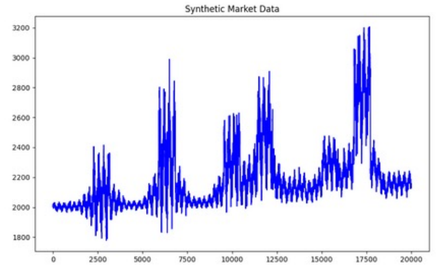
Data

3x data sets

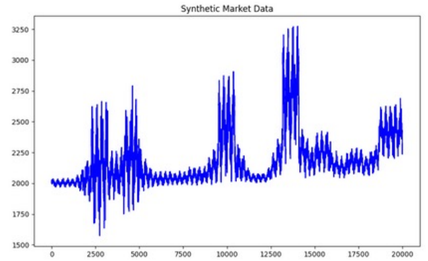
- **Contains**:
  - Synthetic time series data
- **Insights**:
  - Reflect financial market data & trends
  - Single column (univariate)



dataset 1



dataset 2



dataset 3

Concepts

- **Simple Moving Average (SMA)**
  - Taking average of time series over specified window
- **Lagging features**
  - Shifting time series values backwards in time/ providing historical context
- **Rolling standard deviation**
  - Measuring volatility by taking standard deviation over a window
- **Rate of Change**
  - Binary indicator to predict if time series increases or decreases

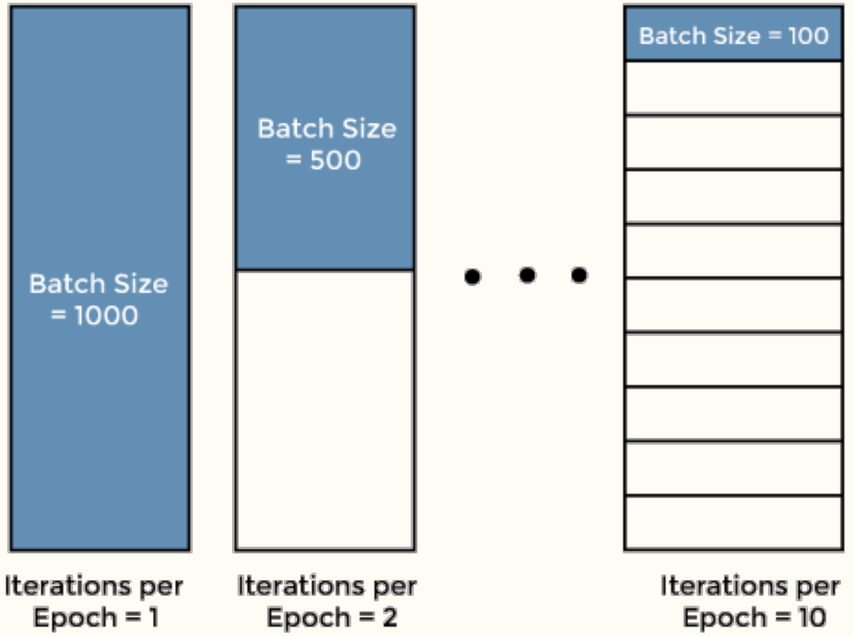
Data Preparation

- Same for all models
- **Data Preprocessing:**
  - Generate lagged features and rolling averages as trend indicators
  - Standardize features
  - 80/20 - train/test split
  - Binary target for price increase

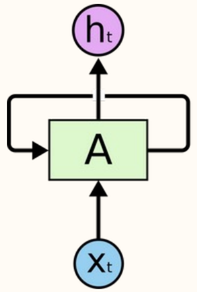
value	SMA_5	SMA_20	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Rolling_STD_5	Rolling_STD_20	ROC
991617	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
481947	NaN	NaN	2006.991617	NaN	NaN	NaN	NaN	NaN	NaN	0
457167	NaN	NaN	2000.481947	2006.991617	NaN	NaN	NaN	NaN	NaN	1
985979	NaN	NaN	2000.457167	2000.481947	2006.991617	NaN	NaN	NaN	NaN	1
836582	2003.350658	NaN	2001.985979	2000.457167	2000.481947	2006.991617	NaN	3.311812	NaN	1
404177	2003.433170	NaN	2006.836582	2001.985979	2000.457167	2000.481947	2006.991617	3.428293	NaN	0
018989	2004.340579	NaN	2007.404177	2006.836582	2001.985979	2000.457167	2000.481947	3.029065	NaN	1
518403	2005.752826	NaN	2005.018989	2007.404177	2006.836582	2001.985979	2000.457167	2.331652	NaN	1
649261	2007.485482	NaN	2007.518403	2005.018989	2007.404177	2006.836582	2001.985979	2.032354	NaN	1
216525	2008.561471	NaN	2010.649261	2007.518403	2005.018989	2007.404177	2006.836582	2.858968	NaN	0
402896	2007.961215	NaN	2012.216525	2010.649261	2007.518403	2005.018989	2007.404177	3.422268	NaN	1
170830	2008.791583	NaN	2004.402896	2012.216525	2010.649261	2007.518403	2005.018989	3.008595	NaN	0
829678	2007.853838	NaN	2009.170830	2004.402896	2012.216525	2010.649261	2007.518403	4.053797	NaN	1
873009	2007.298587	NaN	2002.829678	2009.170830	2004.402896	2012.216525	2010.649261	3.754249	NaN	0
122570	2005.479796	NaN	2007.873009	2002.829678	2009.170830	2004.402896	2012.216525	2.876220	NaN	0
305303	2004.460278	NaN	2003.122570	2007.873009	2002.829678	2009.170830	2004.402896	4.026727	NaN	0
922076	2000.810527	NaN	1999.305303	2003.122570	2007.873009	2002.829678	2009.170830	6.311662	NaN	0
957963	1997.236184	NaN	1990.922076	1999.305303	2003.122570	2007.873009	2002.829678	9.256019	NaN	1
152849	1992.692152	NaN	1984.957963	1990.922076	1999.305303	2003.122570	2007.873009	8.251050	NaN	1
632909	1989.394220	2001.696536	1985.152849	1984.957963	1990.922076	1999.305303	2003.122570	6.038592	8.376451	1
121306	1987.157420	2000.753021	1986.632909	1985.152849	1984.957963	1990.922076	1999.305303	2.461338	8.800652	1

Data Training

- Same for all models
- **Model Training:**
  - Input shaped as (samples, timesteps, features)
  - Train for 50 epochs with early stopping patience 10
  - Batch size 32
  - Evaluate on test set using accuracy and classification report
  - Repeat for each of the 3 datasets



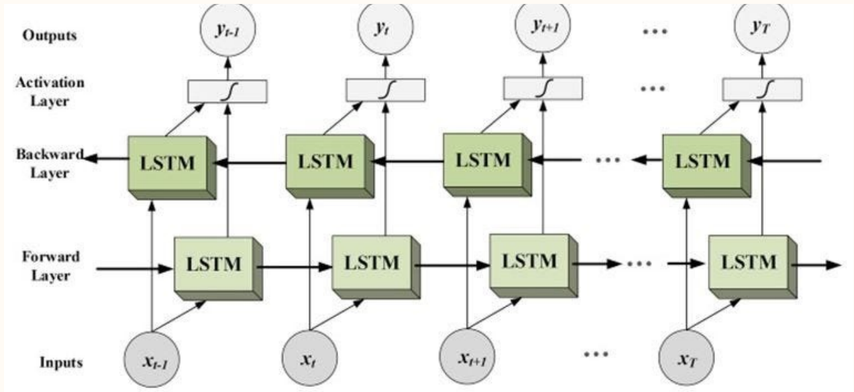
LSTM



- Processes sequence data in 1 direction, either past → future **OR** future → past.

V
S

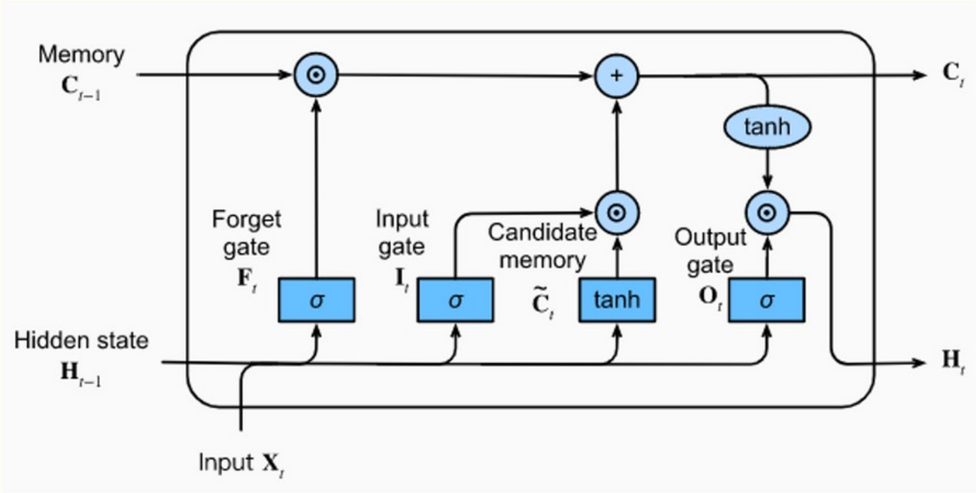
Bidirectional LSTM



- Processes sequence data in both directions, past → future **AND** future → past
- Better model sequence data by incorporating context from both directions.

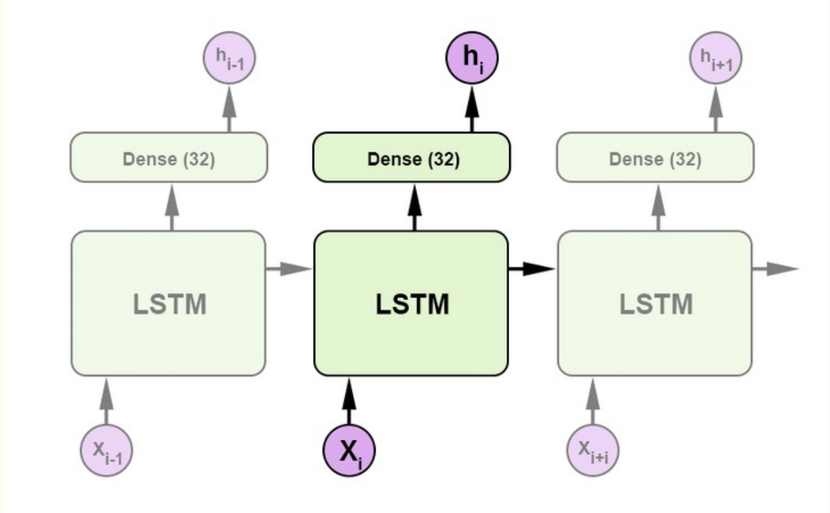
LSTM

`long short-term memory networks`



- Good for sequential data
- memory cells allow the model to:
  - Remember patterns across a long series of data points.
  - Make predictions based on long sequences
- The gates decide what info to store/forget
  - Allow trends and make forecasts using historical time series data.

Architecture



- **Overview**
  - 3 LSTM layers (64, 32, 16 units)
  - 1 Dense layer (8 units)
  - 1 Output layer
- **LSTM Layers**
  - First 2 layers: return_sequences=True
  - 3rd layer: No return_sequences
- **Dense Layer**
  - 1st Dense: 8 units, ReLU activation
- **Output Layer**
  - 1 unit, Sigmoid activation
- **Key Points**
  - Sigmoid for binary classification
  - Hierarchical LSTM structure

Results

Dataset 1

Accuracy : 0.50

F-1 score : 0.34

	Class 0	Class 1
Precision	0.50	1.00
Recall	1.00	0.00

Dataset 2

Accuracy : 0.49

F-1 score : 0.33

	Class 0	Class 1
Precision	1.00	0.49
Recall	0.00	1.00

Dataset 3

Accuracy : 0.50

F-1 score : 0.34

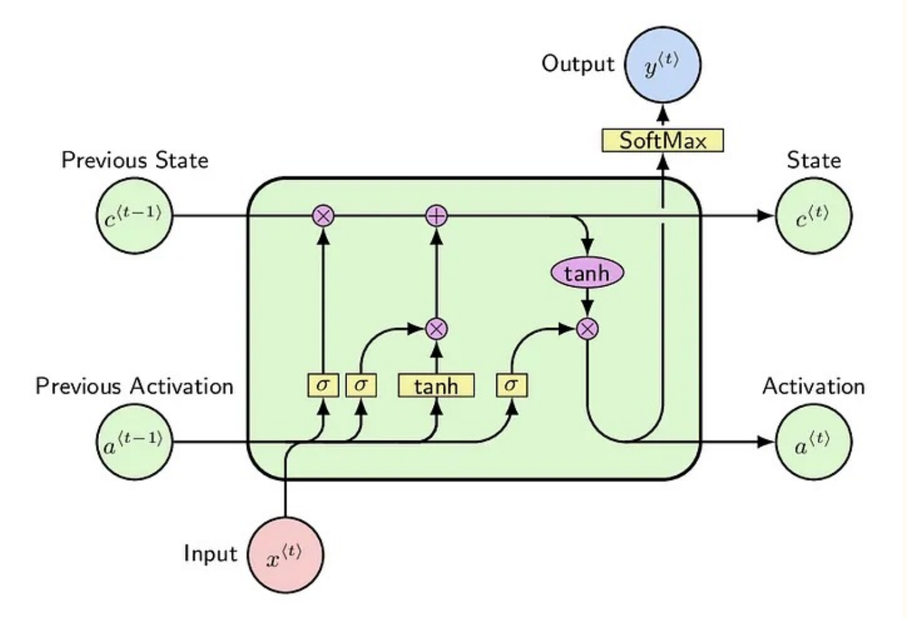
	Class 0	Class 1
Precision	1.00	0.50
Recall	0.00	1.00

Summary

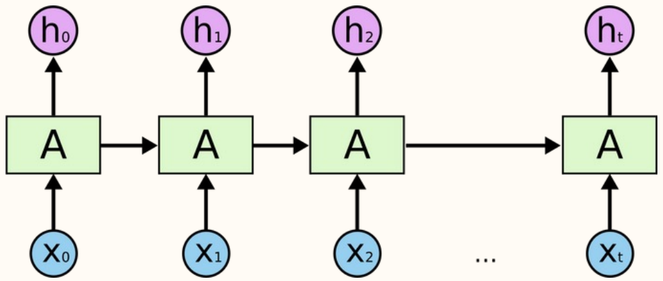
- All three datasets show imbalanced precision and recall values for the classes, with one class having a precision and recall of 1.00 and the other class having a precision and recall of 0.00.
- The accuracy is around 0.50 for all datasets, indicating that the models are not performing better than random chance.
- These results suggest potential issues with the models, such as problems with the training process. Further investigation and model refinement may be necessary to improve performance.

Bidirectional LSTM

A bidirectional LSTM model consists of two LSTM layers stacked together – one that processes the input sequence forwards and one that processes it backwards. This allows the model to have both past and future context when making predictions.



Model Architecture



- **Overview**
  - Input Layer: Takes feature vectors (SMA_5, SMA_20, etc.).
  - LSTM Layers: Bidirectional wrapper with two 32-unit LSTM layers.
  - Dense Layer: 1 node, sigmoid activation for binary classification.
- **Details**
  - LSTM: Hierarchical structure with forward and backward processing.
  - Dense Output: Sigmoid activation for binary classification.
- **Compilation**
  - Loss: Binary Crossentropy
  - Optimizer: Adam
  - Metric: Accuracy

Bidirectional LSTM

Results

Dataset 1

Accuracy : 0.58

F-1 score : 0.58

	Class 0	Class 1
Precision	0.59	0.58
Recall	0.59	0.58

Dataset 2

Accuracy : 0.52

F-1 score : 0.49

	Class 0	Class 1
Precision	0.55	0.51
Recall	0.28	0.77

Dataset 3

Accuracy : 0.57

F-1 Score : 0.56

	Class 0	Class 1
Precision	0.55	0.60
Recall	0.71	0.43

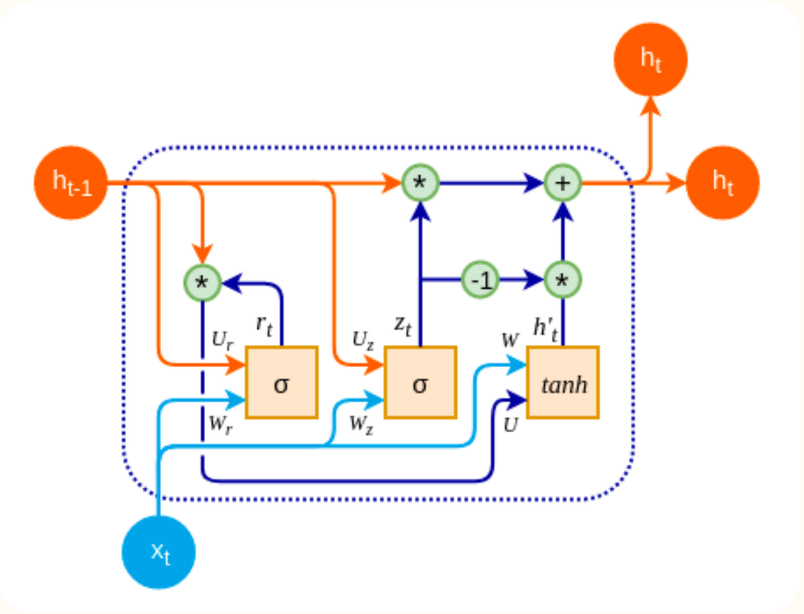
Summary

- Dataset 1 has balanced precision and recall for both classes, with an accuracy of 0.58
- Dataset 2 has imbalanced precision and recall, with lower values for Class 0 and higher values for Class 1, resulting in an accuracy of 0.52
- Dataset 3 shows better performance for Class 0 (higher recall) but lower performance for Class 1 (lower recall), resulting in an accuracy of 0.57

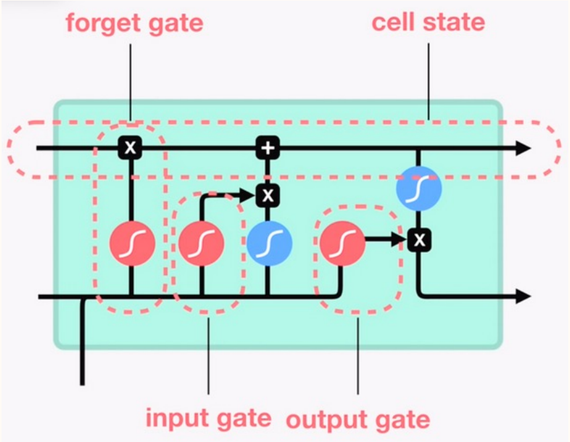
GRU

Gated Recurrent Unit

- A variant of LSTM focused on modeling sequential data
- More streamlined than LSTM, with fewer trainable parameters
- Has reset gate, update gate, and hidden state as main components

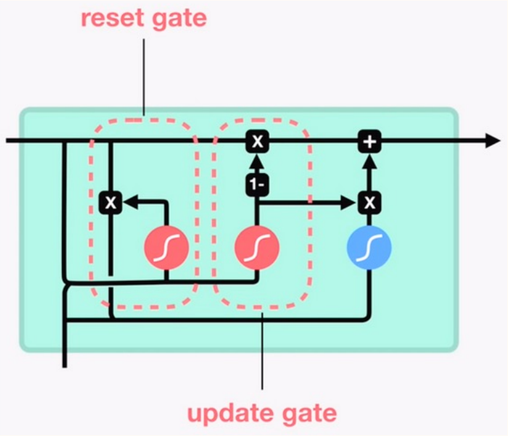


LSTM



V

S

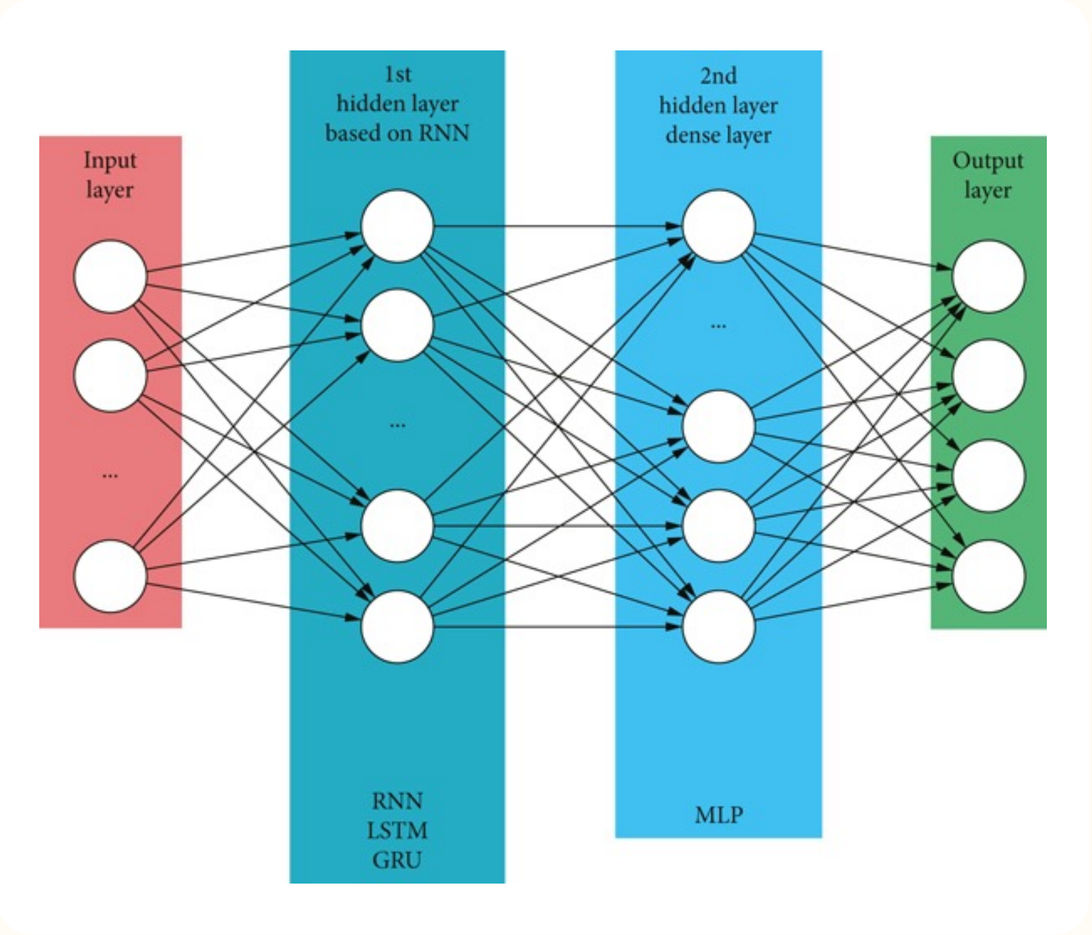


GR

U

Model Architecture

- A sequential model with 1 GRU layer and 1 dense layer
- **GRU layer**
  - 32 units, Relu activation
- **Dense layer**
  - 1 node, Sigmoid activation
- **Optimizer** : Adam



GRU

Results

Dataset 1

Accuracy : 0.58

F-1 score : 0.58

	Class 0	Class 1
Precision	0.59	0.57
Recall	0.54	0.62

Dataset 2

Accuracy : 0.46

F-1 score : 0.43

	Class 0	Class 1
Precision	0.47	0.41
Recall	0.68	0.23

Dataset 3

Accuracy : 0.55

F-1 score : 0.51

	Class 0	Class 1
Precision	0.53	0.62
Recall	0.83	0.27

Summary

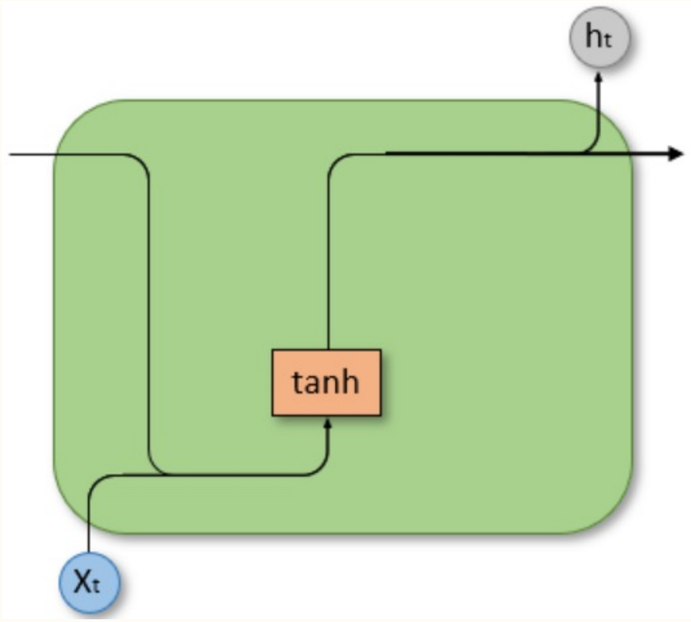
- Dataset 1 shows relatively balanced precision and recall values for both classes, resulting in an accuracy of 0.58
- Dataset 2 has imbalanced precision and recall values, with higher values for Class 0 and lower values for Class 1, resulting in an accuracy of 0.46
- Dataset 3 demonstrates better performance for Class 0 (higher recall) but lower performance for Class 1 (lower recall), resulting in an accuracy of 0.55

RNN

Recurrent Neural Network

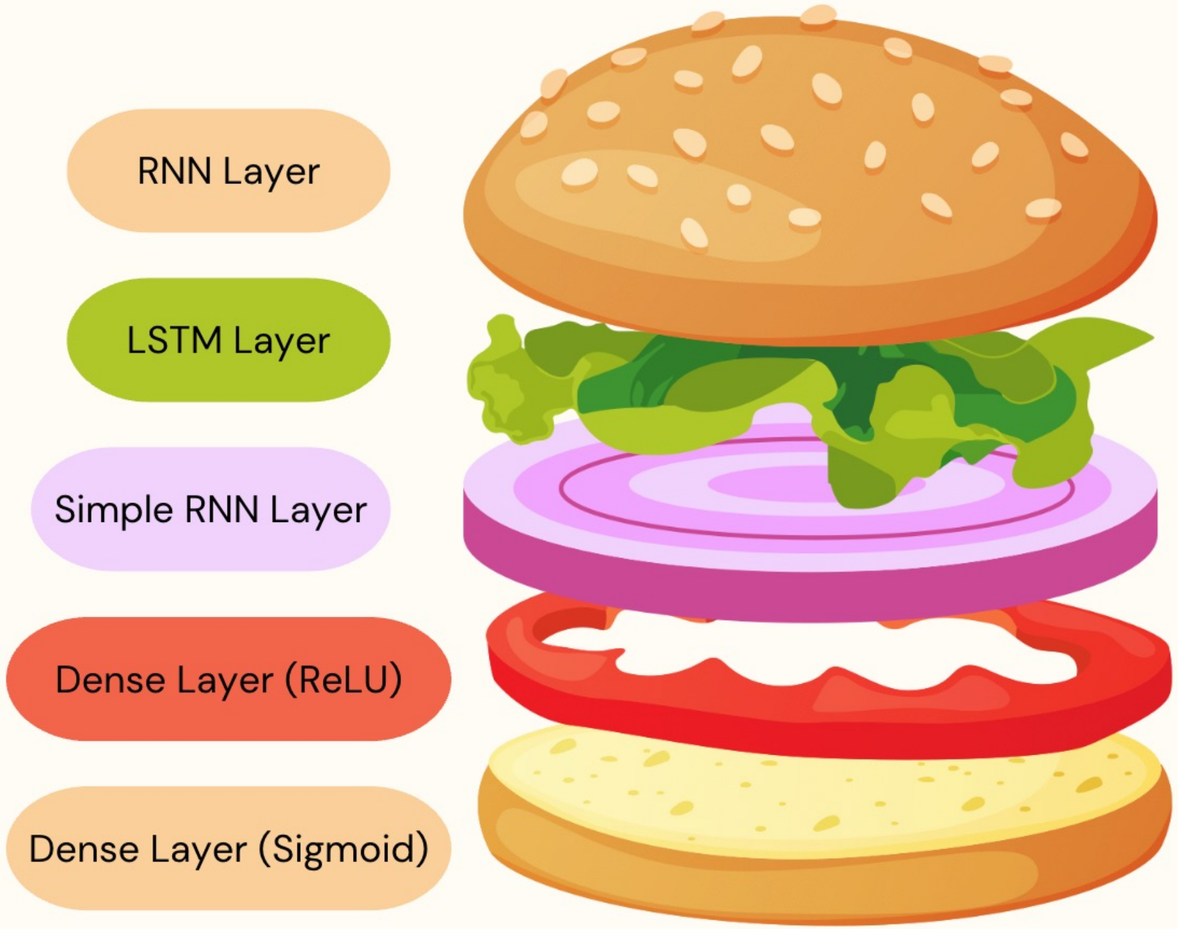
- Artificial neural networks that specialize in processing sequence data
- Have looping or recurrent connections that allow information to persist across time steps of input sequences
    - This gives RNNs memory to learn patterns, context, and temporal dynamics within sequences for tasks like speech, text, time series and more

- Our RNN model is a hybrid model that combines the properties of both SimpleRNNs and LSTMs in order to process the temporal/sequence financial data



Architecture

- Total of five layers
  - **SimpleRNN** Layer (with return_sequences=True)
    - 32 units, ReLU activation.
  - **LSTM** Layer (with return_sequences=True)
    - 32 units, ReLU activation.
  - **SimpleRNN** Layer
    - 32 units, ReLU activation
  - **Dense** Layer
    - 32 units, ReLU activation
  - **Dense** Layer
    - 1 unit, sigmoid activation



Results

Dataset 1

Accuracy : 0.50

F-1 score : 0.50

	Class 0	Class 1
Precision	0.50	0.49
Recall	0.99	0.01

Dataset 2

Accuracy: 0.56

F-1 score : 0.33

	Class 0	Class 1
Precision	0.58	0.55
Recall	0.52	0.61

Dataset 3

Accuracy : 0.50

F-1 score : 0.41

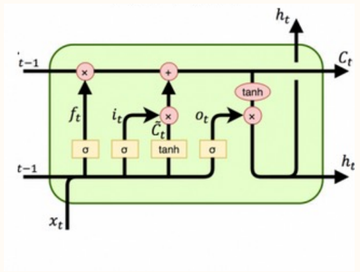
	Class 0	Class 1
Precision	0	0.50
Recall	0	1.00

Summary

- Dataset 1 has a high recall for Class 0 but very low recall for Class 1, resulting in an overall low accuracy of 0.50
- Dataset 2 shows relatively balanced precision and recall for both classes, with an accuracy of 0.56
- Dataset 3 has a precision of 0.50 for Class 1, but the model predicts only Class 1, resulting in a recall of 1.00 for Class 1 and an accuracy of 0.50
    - The model does not predict any instances of Class 0, leading to a confusion matrix with all zeros for Class 0

LSTM

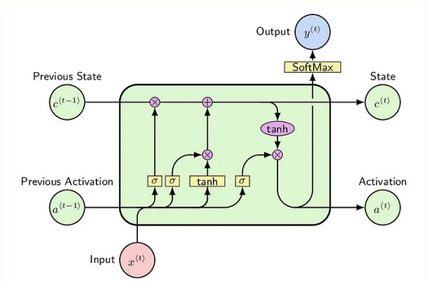
Long Short-Term Memory units



- Addresses vanishing gradient problem in RNNs
- Has memory cell, input, forget, and output gates

Bi-LSTM

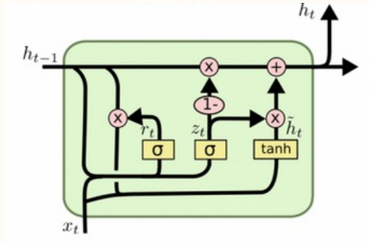
Bidirectional LSTM



- adds one more LSTM layer to LSTM, which reverses the direction of information flow
- combines the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or concatenation

GRU

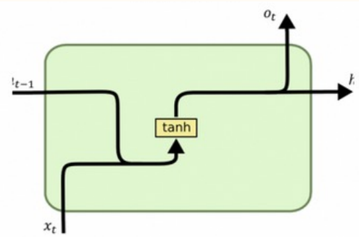
Gated Recurrent Units



- Simpler than LSTM with fewer parameters
- Merges memory cell and hidden state

RNN

Recurrent neural network



- Basic sequential model with feedback loop
- Prone to vanishing gradients
- Limited ability to capture long-range dependencies

💡 Better at handling long-range dependencies

💡 Good with real-world problems, especially in NLP

💡 Balance performance and simplicity

💡 designed for sequential data processing

Confusion Matrix – Dataset 2

LSTM	Predicted Positive	Predicted Negative
Actual Positive		2023
Actual Negative		1974

Bidirectional LSTM	Predicted Positive	Predicted Negative
Actual Positive	1068	955
Actual Negative	693	1281

GRU	Predicted Positive	Predicted Negative
Actual Positive	890	1133
Actual Negative	581	1393

RNN	Predicted Positive	Predicted Negative
Actual Positive	543	1480
Actual Negative	420	1554

Overall Results

LSTM

	Test accuracy	F-1 Score
Dataset 1:	0.50	0.34
Dataset 2:	0.49	0.33
Dataset 3:	0.50	0.34

Bidirectional LSTM

	Test accuracy	F-1 Score
Dataset 1:	0.58	0.58
Dataset 2:	0.52	0.49
Dataset 3:	0.57	0.56

GRU

	Test accuracy	F-1 Score
Dataset 1:	0.58	0.58
Dataset 2:	0.46	0.43
Dataset 3:	0.55	0.51

RNN

	Test accuracy	F-1 Score
Dataset 1:	0.50	0.50
Dataset 2:	0.56	0.33
Dataset 3:	0.50	0.41

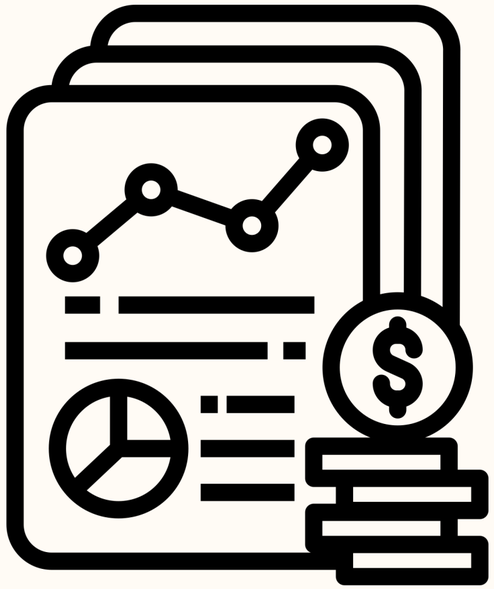
Conclusion

Expectation

- Reasons why we believed LSTM works well with time series
- Exploring other models
  - Bidirectional LSTM, GRU, and RNN

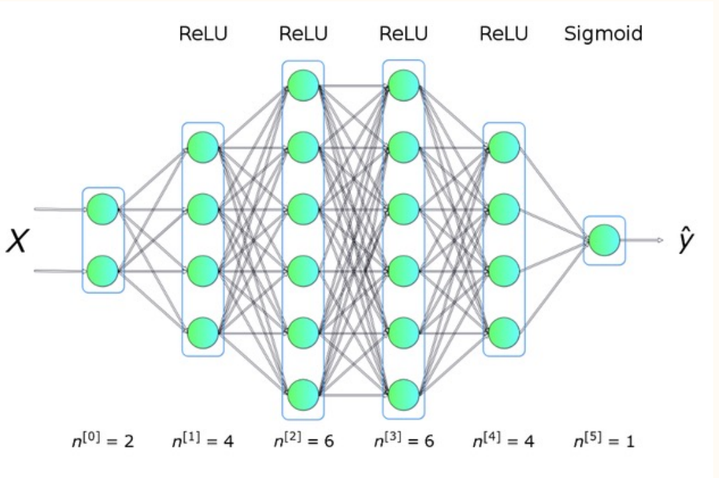
What we got

- Bidirectional LSTM: Highest performance
- GRU: Performed reasonably well
- LSTM: Mixed performance
- RNN: Lowest performance



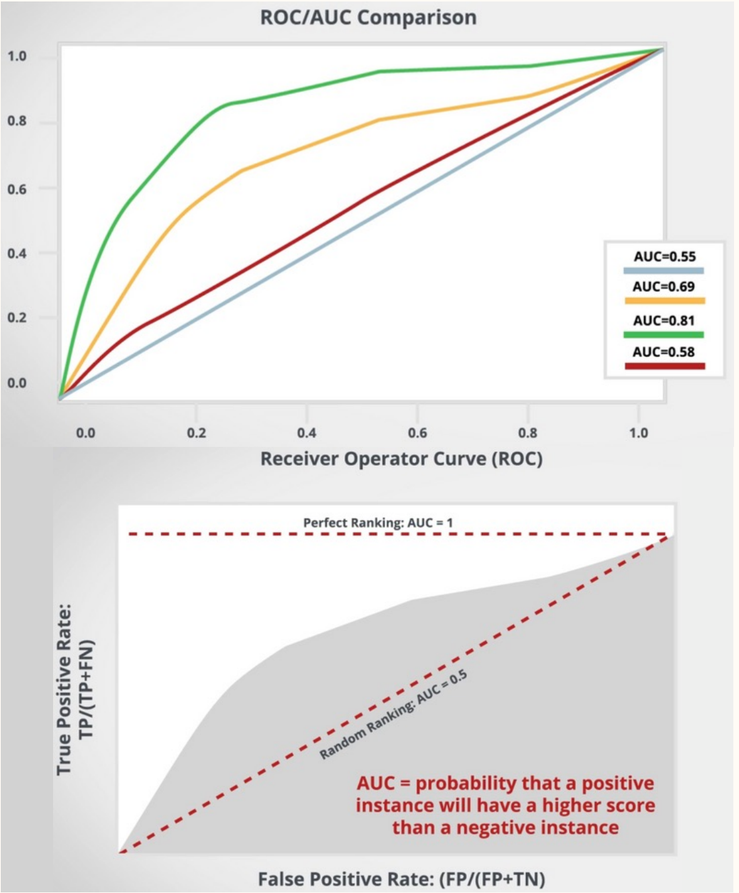
Ways to improve results

- **Hyperparameter Tuning:**
  - Add more hidden layers to neural networks
  - Tune hyperparameters like number of LSTM units, batch size, epochs etc
  - Adjust the learning rate.
- **Data Normalization:**
  - Min-max scaling
  - Try different activation functions like tanh
- **Explore More Models:**
  - Autoregressive Models (AR, ARIMA)
  - Decision trees
  - Temporal Convolutional Networks (TCNs)



Ways to improve results

- **Reduce Overfitting:**
  - Use regularization techniques like dropout
- **Use More Data**
  - Train on larger variety of datasets
- **Evaluation Metrics:**
  - Use AUC-ROC or PRC rather than just accuracy



Learning Experience

- Theory ≠ Reality
- Synthetic data vs. real world data
- Future direction
  - Improve current models
  - Exploring other models
- Came out more knowledgeable

Resources

Time-series Data:

- https://builtin.com/data-science/time-series-python
- https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python

Data transformation (normalization & standardization):

- https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html

Data leakage / look ahead bias:

- https://bowtiedraptor.substack.com/p/look-ahead-bias-and-how-to-prevent

Documentation

- Notion
  - Scheduling
  - Note-taking
  - Keeping track of links

- Google Collab
  - Project imported from Jupyter notebook

- Future Website
  - Documents, background, process, progress and future work

Any questions?

Thank you Matthias, Dominic and Break Through Tech AI Team!

Feedback

Colab Notebook