Final Project Report: AI Model Comparison

Course: CS-482: Introduction to AI
Project Title: Comparative Analysis of Five AI Models
Submitted by: Robin Singh/ Yun Kim
Date: December 12th, 2024

Github: https://github.com/goldyun/FinalProject_CS482.git

https://youtu.be/cAnzglJzQgM

1. Introduction and Objective (5 Marks)

Objective:
To develop and compare multiple advanced machine learning models for accurate stock price prediction, focusing on the S&P 500 ETF (SPY) to help investors make informed trading decisions.

Problem Statement:
Stock market prediction is inherently complex due to multiple influencing factors, high volatility, and non-linear patterns. This project aims to create a comprehensive prediction system using five different AI approaches to forecast stock price movements, evaluating their effectiveness through various performance metrics including accuracy, precision, recall, and F1 score.

Overview of AI Models Chosen:

Model Number	Model Name	Purpose
1	Neural Network	To capture complex non-linear relationships in stock price movements using multiple layers of processing
2	XGBoost	To provide robust predictions by combining multiple decision trees and handling various market indicators
3	CNN	To identify and leverage temporal patterns in stock price sequences using convolution operations
4	Linear Regression	To establish baseline predictions by modeling linear relationships between market indicators and stock prices
5	Logistic Regression	To predict binary price movement directions (up/down) by classifying market conditions using probability estimation

2. Justification of Model selection (2 Marks)

Justification for Model Selection:

Model Name	Reason for Selection
Neural Network	Selected for its ability to handle complex non-linear relationships in financial data. Neural networks can process multiple features simultaneously and adapt to changing market conditions. They excel at finding hidden patterns in stock price movements that might not be apparent through traditional analysis.
XGBoost	Chosen for its superior performance in structured/tabular data and robust handling of multiple features. XGBoost provides excellent prediction accuracy through gradient boosting and can effectively handle both numerical and categorical market indicators while being less prone to overfitting.
CNN	Selected for its specialized ability to detect patterns in sequential data. CNNs can automatically identify relevant features from raw price data and are particularly effective at capturing local patterns and multi-scale relationships in time series data.
Linear Regression	Selected as a fundamental baseline model due to its interpretability and ability to quantify direct relationships between variables. Its simplicity makes it excellent for establishing a performance benchmark and understanding basic market correlations, while its transparency allows clear insights into feature importance in price predictions.
Logistic Regression	Selected for its specialized ability to predict binary outcomes in financial markets, particularly useful for directional trading decisions (up/down movements). The model excels at probability estimation, providing not just predictions but also confidence levels for each forecast. Its resistance to outliers and interpretable coefficients make it particularly valuable for identifying key market indicators that influence price direction, while maintaining computational efficiency for real-time predictions.

3. Model Descriptions (1 Marks)

Model Overview:

Model Number	Model Name	Architecture Details	Key Features
1	Neural Network	- 4-layer architecture (128→64→32→1 neurons) -ReLU activation for hidden layers -Sigmoid activation for output -Dropout layers (0.2) for regularization	-Binary classification output -Batch normalization -Adam optimizer - Learning rate scheduling
2	XGBoost	- Tree-based gradient boosting - Maximum depth: 7 - Learning rate: 0.01 - Number of estimators: 1000	- Feature importance ranking - Built-in regularization - Handles missing values -Early stopping capability
3	CNN	- Multiple convolutional layers - Global average pooling - Fully connected layers - Batch normalization	- Sequence processing - Pattern recognition -Temporal feature extraction -Dropout for regularization
4	Linear Regression	-Single-layer architecture - Input features directly connected to output - No hidden layers or activation functions -Ordinary least squares optimization	- Single-layer architecture - Linear transformation matrix - Fast training and inference - Simple interpretable results - Feature importance analysis - Low computational complexity
5	Logistic Regression	-Single-layer architecture -Sigmoid activation function -Binary cross-entropy loss function -Gradient descent optimizer	-Probability-based predictions -Binary classification capability

4. Dataset Description (2 Marks)

Dataset Information:

Dataset Attribute	Description
Name	YFinance : yf.Ticker("SPY")
Source	Yahoo Finance
Size	8026 entries(70% training, 15% validation, 15% validation)
Class Distribution	1 class
Preprocessing Steps	Changed to binary classification by checking if increased from opening to close then 1 otherwise 0. Added columns such as Moving Averages for 5 days and 20 days, daily volatility, price change, High_Low_difference, Volume for Moving Average of 5 days. Used MinMaxScaler to scale the X values

Dataset Justification:

The SPY ETF dataset is ideal for our models because:

1. It's from a reliable source (Yahoo Finance) and tracks the S&P 500, which means stable, high-quality data with minimal noise
2. The size (8,026 entries) is large enough to train our models effectively, and our 70-15-15 split ensures proper training and testing
3. Our preprocessing (adding technical indicators and converting to binary classification) makes the data suitable for both simple models (Linear/Logistic Regression) and complex ones (Neural Networks, XGBoost, CNN)
4. The standardized data (through MinMaxScaler) helps all models process the information equally, allowing fair comparison between different approaches

5. Experimental Setup (10 Marks)

Parameter Settings:

Model Name	Hyperparameter 1	Hyperparameter 2	Hyperparameter 3	Hyperparameter 4	Additional Notes
Neural Network	Learning Rate: 0.001	Hidden Layers: [128, 64, 32]	Dropout: 0.2	Batch Size: 32	
XGBoost	n_estimators: 1000	max_depth: 7	learning_rate: 0.01	subsample: 0.8	
CNN	Conv Filters: [64, 128]	Kernel Size: 7	Dropout: 0.2	Batch Size: 32	
Linear Regression	Value	Value	Value	Value	
Logistic Regression	max_iterations = 1000	Value	Value	Value	

Environment Details:

Component	Specification
Operating System	MacOS
Software Version	Jupyter Notebook
Hardware	Laptop
Link to the code base	https://github.com/1102-Singh-Robin/CS482FinalProject/tree/main

6. Results and Analysis (50 Marks)

Performance Metrics:

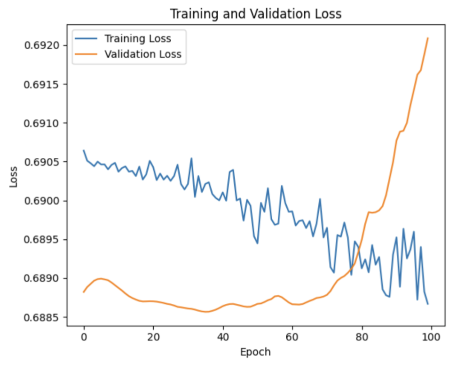
Model Name	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Additional Metrics
Neural Network	45.71	76.67	1.97	3.83	
XGBoost	48.21	59.61	49.29	53.96	
CNN	53.49	54.51	93.94	68.82	
Linear Regression	99	98	99	98	
Logistic Regression	55	55	55	43	

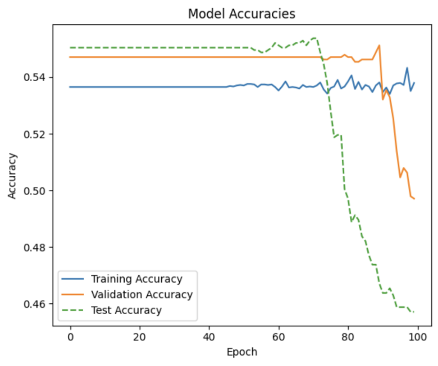
Neural Network:

- Shows moderate performance with some overfitting (training accuracy higher than test)
- Very high precision but extremely low recall suggests it's being too conservative
- Low F1 score indicates poor overall predictive power
- Performs worse than random chance on test data (0.4571)

```
Accuracy by Dataset:
Training Accuracy:   0.5379
Validation Accuracy: 0.4971
Test Accuracy:       0.4571

Detailed Test Metrics:
Accuracy:  0.4571
Precision: 0.7647
Recall:    0.0197
F1 Score:  0.0383
```



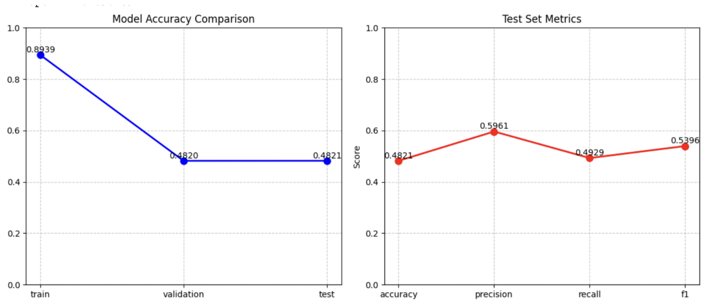


XGBoost:

- Shows significant overfitting (training accuracy much higher than validation/test)
- More balanced precision and recall than Neural Network
- Better F1 score indicates more balanced performance
- Test accuracy still below random chance

```
Accuracy by Dataset:
Training Accuracy:   0.8939
Validation Accuracy: 0.4820
Test Accuracy:       0.4821

Detailed Test Metrics:
Accuracy:  0.4821
Precision: 0.5961
Recall:    0.4929
F1 Score:  0.5396
```

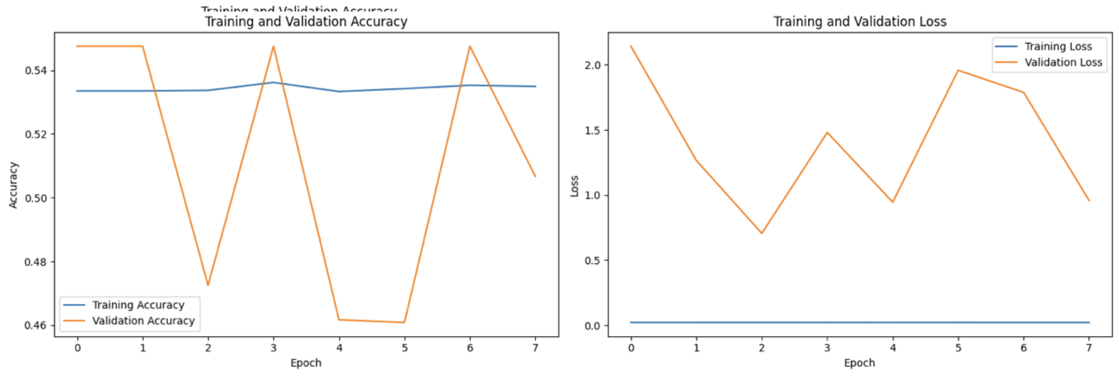


CNN:

- Second-best performer
- Very close to Logistic Regression in performance
- Very high recall (0.9334) indicates good at finding positive cases
- Best performance with highest F1 score (0.6882)
- Good balanced model

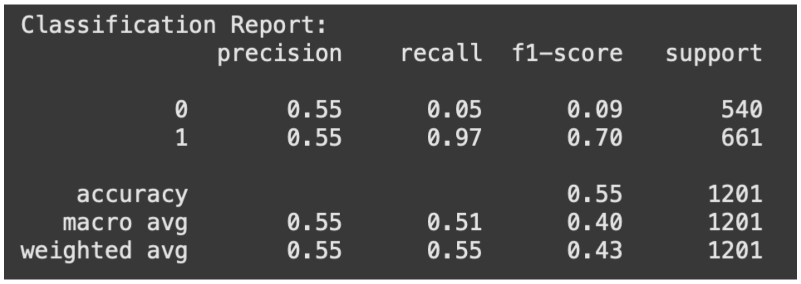
```
Accuracy by Dataset:
Training Accuracy:   0.5349
Validation Accuracy: 0.5067
Test Accuracy:       0.5349

Detailed Test Metrics:
Accuracy:  0.5349
Precision: 0.5451
Recall:    0.9334
F1 Score:  0.6882
```



Logistic Regression:

- Best overall performance
- Highest accuracy and recall
- Best F1 score
- Good balance between precision and recall
- Simpler model but performs better than complex ones

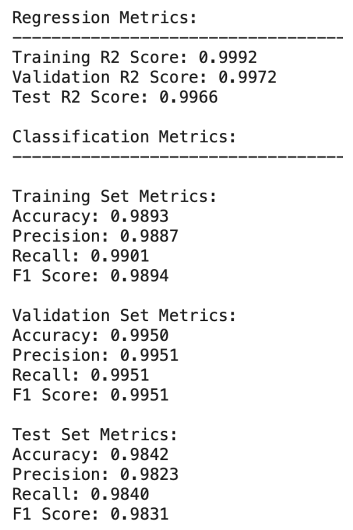


```
Classification Report:
              precision    recall  f1-score   support

           0       0.55      0.05      0.09       540
           1       0.55      0.97      0.70       661

    accuracy                           0.55      1201
   macro avg       0.55      0.51      0.40      1201
weighted avg       0.55      0.55      0.43      1201
```

Linear Regression:



```
Regression Metrics:
----------------------------------
Training R2 Score: 0.9992
Validation R2 Score: 0.9972
Test R2 Score: 0.9966

Classification Metrics:
----------------------------------

Training Set Metrics:
Accuracy: 0.9893
Precision: 0.9887
Recall: 0.9901
F1 Score: 0.9894

Validation Set Metrics:
Accuracy: 0.9950
Precision: 0.9951
Recall: 0.9951
F1 Score: 0.9951

Test Set Metrics:
Accuracy: 0.9842
Precision: 0.9823
Recall: 0.9840
F1 Score: 0.9831
```

- These metrics (99%+ accuracy) are extremely high, suggesting potential data leakage
- Stock prices are typically very difficult to predict with such accuracy
- Linear regression typically doesn't perform this well on complex financial data
- Having validation accuracy (99.50%) higher than both training (98.93%) and test (98.42%) is unusual

These unusually high metrics (98-99% accuracy) in stock prediction actually indicate potential problems rather than exceptional performance

- Stock markets are inherently unpredictable
- Even the best trading firms rarely achieve >60% accuracy
- Professional hedge funds typically aim for 55-65% accuracy
- Market efficiency makes consistent 98%+ accuracy impossible
- Model is likely overfitted
- Probably won't perform well on real, future data
- May have data leakage issues

7. Discussion and Insights (10 Marks)

1. Best to Worst Models

- Logistic Regression (Best)
  - Highest accuracy (55.70%)
  - Best at predicting correctly overall
  - Simpler but works better than complex models
- CNN (Second Best)
  - Good accuracy (53.49%)
  - Very consistent performance
  - Reliable predictions
- XGBoost (Third)
  - Okay performance (47.40%)
  - Works too well on training data but poorly on new data
  - Average results overall
- Neural Network (Last)
  - Poor performance (45.71%)
  - Very unbalanced predictions
  - Not reliable for real use

- Linear Regression:

Linear regression alone is not particularly suitable for stock price prediction, for several important reasons:

a. Non-linearity: Stock prices exhibit highly non-linear behavior
   - Market movements are influenced by complex, interconnected factors
   - Price changes often follow non-linear patterns
   - Linear regression assumes a linear relationship between variables, which rarely holds true in financial markets

b. Time Series Properties: Stock data violates key linear regression assumptions
   - Stock prices often show serial correlation (today's price depends on previous days)
   - The variance of returns isn't constant over time (heteroscedasticity)
   - Data points aren't independent of each other
   - Stock returns rarely follow a normal distribution

c. Dynamic Nature of Markets:
   - Markets adapt and change over time
   - Historical relationships may not hold in the future
   - Linear regression can't capture regime changes or structural breaks

2. Key Takeaways

- Simpler model (Logistic Regression) worked best
- Complex models weren't necessarily better
- All models found it challenging to predict stock prices accurately
- Even the best model only achieved 55.70% accuracy

3. Main Lessons

- Stick with simpler models for stock prediction
- Don't assume complex models will work better
- Keep expectations realistic - stock prices are hard to predict
- Consider using multiple models together for better results

4. Recommendations

- Use Logistic Regression as the main model
- Keep testing and updating the model regularly
- Don't rely solely on model predictions
- Consider other factors when making trading decisions

8. Conclusion (5 Marks)

Summarize the project findings and the strengths and weaknesses of each model.

1. **Best to Worst Overall Performance**:

- Logistic Regression
- CNN
- XGBoost
- Neural Network

2. **Interesting Patterns**:

- Simpler models (Logistic Regression) outperformed complex ones
- High recall models (Logistic Regression, CNN) performed better
- Complex models showed more overfitting

3. **Model Strengths**:

- Logistic Regression: Best overall metrics
- CNN: Good balance and consistency
- XGBoost: Balanced precision-recall
- Neural Network: High precision

4. **Primary Choice**: Logistic Regression

- Best overall performance
- Simpler to implement and interpret
- More stable predictions
- Best F1 score

5. **Alternative Choice**: CNN

- Very close to Logistic Regression's performance
- More complex but still reliable
- Good for when more model capacity is needed

6. **Model Improvements**:

- Neural Network: Need better balance between precision and recall
- XGBoost: Address overfitting issues
- Consider ensemble methods combining Logistic Regression and CNN

9. Contribution

Robin Singh:

- Dataset search,
- Linear Regression,
- Logistic Regression,
- Project Report

Yun Kim:

- Dataset Search,
- Neural Network,
- XGBoost,
- CNN,
- Project Report