Chapter 1

INTRODUCTION

Background of report

The stock market is a complex and dynamic system influenced by various factors, such as economic indicators, market sentiment, geopolitical events, and company-specific information. Predicting the behavior of the stock market accurately has always been a challenging task for investors and financial analysts. However, with the advancements in technology and the availability of vast amounts of historical data, machine learning algorithms have emerged as powerful tools for predicting stock market trends.

Machine learning techniques leverage the computational power of computers to analyze historical data, identify patterns, and make predictions based on statistical models. These algorithms have the potential to uncover hidden relationships and capture subtle market signals that may not be easily discernible by human analysts. This has opened up new avenues for improving the accuracy of stock market predictions and enhancing investment decision-making.

Problem Statement

In today's world, the stock market has become a source of enthusiasm and interest among both youngsters and adults. Many individuals view it as a potential source of side income and a means to hustle and build wealth. However, due to the lack of knowledge and expertise in stock market analysis, individuals often face challenges and end up losing money in their investment endeavors.

The volatility and complexity of the stock market make it difficult for individuals to make accurate predictions and informed investment decisions. Traditional methods of

analysis may not provide the necessary accuracy and reliability needed to navigate the dynamic market environment effectively. As a result, there is a need to explore alternative approaches that can enhance the accuracy level of stock market predictions and empower individuals to make more informed investment choices.

By applying machine learning models to stock market prediction, we can leverage the power of data-driven algorithms to analyze historical market patterns, identify relevant features, and make predictions with a higher degree of accuracy. Machine learning models have the potential to uncover hidden relationships and patterns in vast amounts of data that may not be easily discernible through traditional analysis methods. Therefore, employing machine learning techniques can provide individuals with valuable insights and assist them in making better-informed investment decisions.

The problem statement revolves around the need to address the lack of knowledge and accuracy in stock market predictions among individuals. By applying machine learning models to this problem, we aim to add an additional level of accuracy to their predictions, thereby empowering individuals to make more informed investment choices and potentially mitigate the risk of financial losses.

Goal & Vision

The goal of this project is to explore the application of machine learning techniques, specifically the random forest algorithm, in predicting stock market trends. By leveraging historical stock market data and employing the random forest model, we aim to develop a predictive framework that can generate valuable insights for investors and traders. The vision is to enhance the accuracy of stock market predictions and provide decision support tools that can assist market participants in making informed investment choices.

Chapter 2

LITERATURE REVIEW

1. Studies Using Artificial Neural Networks to Predict Stock Market Values

The first set of articles includes studies that primarily focus on stock market prediction using artificial neural networks (ANNs). ANNs are computational models based on biological neural networks. In the network, sets of nodes are grouped into layers starting with an input layer and ending with an output layer. Signals are transmitted (propagated) through the connected nodes as they learn based on examples and attempt to reduce the level of prediction error. As the system is working to improve its performance, weights are adjusted for the signals between connected nodes. The following provides a brief description of each ANN-related study's unique research focus and findings.

Jasic and Wood (2004) developed an artificial neural network to predict daily stock market index returns using data from several global stock markets. The focus is on trying to support profitable trading. A method is introduced based on univariate neural networks using untransformed data inputs to provide short-term stock market index return predictions. The study uses the daily closing values of the Standard and Poor's 500 Index (S&P 500), the German DAX Index, the Japanese TOPIX index, and London's Financial Times Stock Exchange Index (FTSE All Share). The samples for the S&P 500, DAX and FTSE Index are from January 1, 1965 to November 11, 1999. The sample for TOPIX covers the period from January 1, 1969 to November 11, 1999 since data from earlier years was not available. The prediction performance for the neural network is evaluated against a benchmark linear autoregressive model and prediction improvement is confirmed when applied to the S&P 500 and DAX indices.

Enke and Thawornwong (2005) use a machine learning information gain technique to evaluate the predictive relationships for numerous financial and economic variables. By computing the information gain for each model variable, a ranking of the variables is obtained. A threshold is determined to select only the strongest relevant variables to be retained in the forecasting models. Neural network models for level

estimation and classification are examined for their ability to provide an effective forecast of future values. A cross-validation technique is also employed to improve the generalizability of several models. The models are compared using S&P data from a 24-year period from March 1976 to December 1999. The results show that the trading strategies guided by the classification models generate higher risk-adjusted profits than the buy-and-hold strategy, the other neural network models, and the linear regression models.

Chapter 3

TOOLS USED

3.1 Python

3.2 Tensorflow

3.3 Numpy

3.4 Pandas

3.5 Scikit learn

3.6 Keras

Chapter 4

Modules Used

4.1 LSTM

Long Short-Term Memory (LSTM) is a specialized recurrent neural network architecture designed to overcome the vanishing gradient problem inherent in traditional RNNs. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs are adept at capturing long-term dependencies in sequential data, making them particularly effective in tasks such as natural language processing and time series prediction. The key innovation lies in the use of cell states and gates, allowing LSTMs to selectively store, forget, and output information. With its ability to maintain context over extended sequences, LSTM has become a foundational element in deep learning, contributing significantly to advancements in various fields reliant on sequential data analysis.

4.2 Random Forest

Random Forest is a powerful ensemble learning method widely employed in machine learning for classification and regression tasks. By constructing multiple decision trees through bootstrap sampling and introducing randomness in feature selection, Random Forest mitigates overfitting and enhances the robustness of predictions. The final outcome is determined by a majority vote for classification or an averaging process for regression, resulting in a more stable and accurate model than individual trees. Renowned for its versatility, Random Forests are adept at handling diverse data types and complexities, providing valuable insights into feature importance for better interpretability. This approach has found applications across various domains, showcasing its effectiveness in practical machine learning scenarios.

4.3 Moving Average

In machine learning, moving averages are employed as a data preprocessing technique to smooth out fluctuations in time series data. By calculating the average of a specified window of recent data points, moving averages help reduce noise and highlight trends, making the data more amenable to analysis and modeling. This technique is

particularly useful in tasks such as forecasting and anomaly detection, where understanding the underlying patterns in sequential data is crucial. Additionally, moving averages are commonly applied in feature engineering to provide models with a more stable input, aiding in the extraction of meaningful patterns and contributing to improved predictive performance. Whether used as a standalone method or as part of a more complex machine learning pipeline, moving averages play a valuable role in enhancing the interpretability and accuracy of models dealing with time-dependent data.

4.4 XGBoost:

XGBoost is a robust machine-learning algorithm that can help you understand your data and make better decisions. XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models.

Chapter 5

WORKING

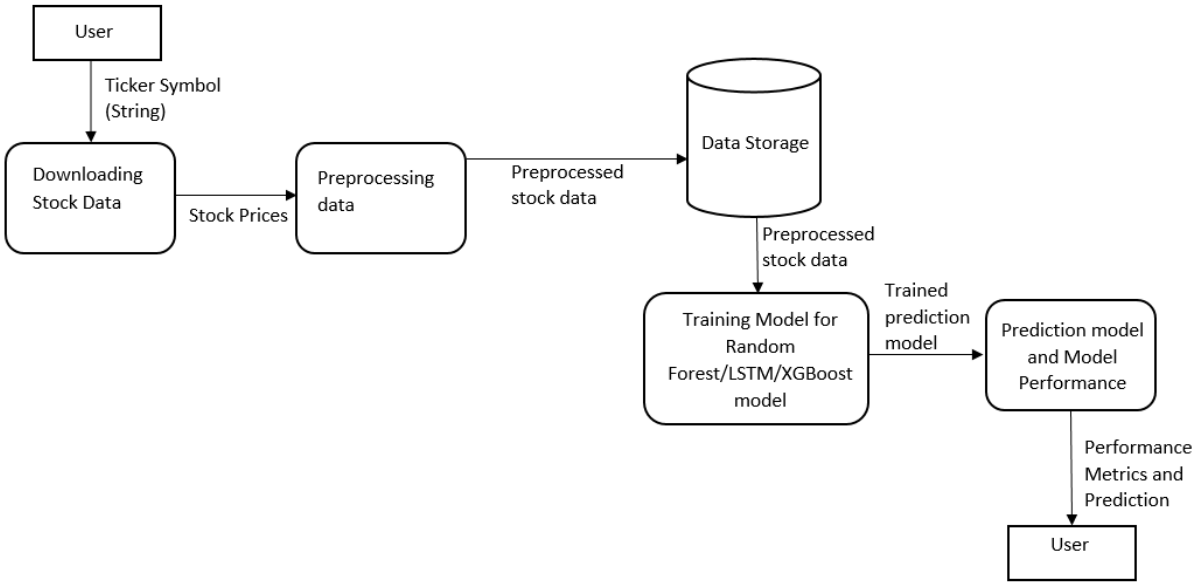

fig: data flow model

5.1 Working of the machine learning model:

The architecture of the stock market prediction system involves several components working together to provide accurate predictions and insights. Here is an overview of the working of the architecture:

5.1. 1 Data collection:

The first part encompasses data collection, where historical stock data is gathered to serve as the foundation for training and testing the model. This step involves utilizing the 'yfinance' library to retrieve the desired stock data based on the user-provided ticker symbol, which is then stored in a CSV file for easy access and manipulation.

5.1.2 Model Building and training:

Random Forest Model:

Imports necessary libraries for modeling and evaluation Defines a Random Forest classifier with specific hyperparameters.Splits the data into training (50 days) and testing (50 days) sets.Chooses relevant features like Close, Volume, Open, High, and Low.Trains the model on the training data and predicts on the testing data.Combines actual and predicted values to compare and plot.Calculates and prints the precision score for the model's performance.

LSTM model:

Data preparation: Extract and reshape features from training data.

Model definition: Create a Sequential model with three layers:

Two LSTM layers with 50 units each.One Dense layer with a single neuron and sigmoid activation.

Model compilation: Set loss function to binary_crossentropy and optimizer to adam.

.Model training: Train the model for 20 epochs with a batch size of 32.

Prediction probabilities: Use the model to predict probabilities of price increases for the test set.

Convert probabilities to labels: Use a threshold of 0.5 to convert probabilities to binary predictions for price increase/decrease.

Calculate precision score: Measure the accuracy of predicted price increases.

Count predictions: Analyze the distribution of predicted price increases and decreases.

XGboost model :

An XGBoost model with specific hyperparameters is defined.objective="binary:logistic" specifies a binary classification task.n_estimators=200 sets the number of decision trees to 200.Other hyperparameters like min_child_weight, max_depth, and learning_rate are also tuned. random_state=1 ensures reproducibility.

5.1.3 Backtesting:

Defines a function predict to predict on new data based on a trained model.Defines a function backtest to perform backtesting with specific steps which are Iterates through the data with a defined step size. For each iteration, separates data into training and testing sets. Makes predictions on the testing set using the trained model. Combines all predictions and actual values for further analysis. Performs backtesting on the entire data and calculates precision score. Prints the precision score and counts of predicted values.

5.1.4 Additional predictors:

Defines a list of time horizons for calculating moving averages.Creates new features by calculating ratios and trends based on the moving averages.Drops rows with missing values and sets the index name for better display.Updates the model definition and uses the new features for prediction.Performs backtesting with the updated model and new features.Prints the precision score and counts of predictions with the improved model.

RESULTS

Output of Data Preprocessing Module

data 09-12-2023 07:40 AM Microsoft Excel Co... 1,103 KB

data

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2	1986-03-1	0.054893	0.062965	0.054893	0.060274	1.03E+09	0	0
3	1986-03-1	0.060274	0.063503	0.060274	0.062427	3.08E+08	0	0
4	1986-03-1	0.062427	0.064042	0.062427	0.063504	1.33E+08	0	0
5	1986-03-1	0.063504	0.064042	0.06135	0.061889	67766400	0	0
6	1986-03-1	0.061888	0.062427	0.060274	0.060812	47894400	0	0
7	1986-03-2	0.060812	0.060812	0.05866	0.059198	58435200	0	0
8	1986-03-2	0.059198	0.060274	0.056507	0.057583	59990400	0	0
9	1986-03-2	0.057583	0.057583	0.055431	0.055969	65289600	0	0
10	1986-03-2	0.055969	0.057045	0.055431	0.057045	32083200	0	0
11	1986-03-2	0.057045	0.059198	0.056507	0.05866	22752000	0	0
12	1986-03-2	0.05866	0.059736	0.05866	0.059736	16848000	0	0
13	1986-03-3	0.059736	0.059736	0.058122	0.059198	12873600	0	0
14	1986-04-0	0.059198	0.059198	0.05866	0.05866	11088000	0	0
15	1986-04-0	0.05866	0.060274	0.05866	0.059198	27014400	0	0
16	1986-04-0	0.059736	0.06135	0.059736	0.059736	23040000	0	0
17	1986-04-0	0.059736	0.060274	0.059736	0.059736	26582400	0	0
18	1986-04-0	0.059736	0.060274	0.057583	0.05866	16560000	0	0
19	1986-04-0	0.05866	0.060274	0.05866	0.059198	10252800	0	0
20	1986-04-0	0.059198	0.060812	0.059198	0.060274	12153600	0	0
21	1986-04-1	0.060274	0.06135	0.059198	0.060812	13881600	0	0
22	1986-04-1	0.06135	0.062965	0.06135	0.061889	17222400	0	0
23	1986-04-1	0.061888	0.062965	0.061888	0.062427	12153600	0	0
24	1986-04-1	0.062427	0.062427	0.060274	0.062427	9302400	0	0
25	1986-04-1	0.062427	0.065118	0.061889	0.06458	31910400	0	0

Output of the Trained Models (Random Forest/LSTM/XGboost)

```
1)
Precision Score: 0.7333
2)
Precision score after back testing:  0.5107097858042839
Predictions count: Predictions
0    3669
1    2381
Name: count, dtype: int64
3)
Precision score after considering the moving averages:  0.5387096774193548
0.5387096774193548
Predictions
0.0    4740
1.0     310
Name: count, dtype: int64
4)
Precision Score (LSTM): 0.64
Predictions
1.0: 50
5)
Precision Score (XGBoost): 0.8
Predictions
0: 30
1: 20
```

CONCLUSION

Future scope of our project:

The objective for this study is to identify directions for future machine learning stock market prediction research based upon a review of current literature. Given the ML-related systems, problem contexts, and findings described in each selected article, and the taxonomy categories presented earlier, several conclusions can be made about our current knowledge in this research area. Our current model's prediction accuracy is hindered by the limited amount of available datasets. However, there are several potential avenues for improvement:

1. **Enrichment :** Data Gathering a larger and more diverse dataset that encompasses a wider range of market conditions, economic indicators, news sentiment, and other relevant factors can enhance the model's ability to capture complex market dynamics.

2. **Feature Engineering** : Exploring and incorporating additional relevant features, such as technical indicators, market sentiment analysis, or macroeconomic data, can provide a more comprehensive representation of the stock market's behavior and improve prediction accuracy.

3. **Advanced Machine Learning Techniques :** Exploring advanced machine learning techniques, such as deep learning models (e.g., recurrent neural networks or transformers), ensemble methods, or reinforcement learning algorithms, can potentially capture more intricate patterns and relationships in the data, leading to improved prediction performance.

4. **Hyperparameter Optimization:** Fine-tuning the hyperparameters of the chosen machine learning model, such as the number of estimators, minimum samples split, or maximum depth, through systematic experimentation or using automated

techniques like grid search or Bayesian optimization, can help optimize the model's performance.

5. **Ensemble Models:** Combining multiple models, such as using an ensemble of different machine learning algorithms or incorporating expert opinions from financial analysts, can leverage the strengths of individual models and potentially improve prediction accuracy and robustness.

6. **Real-time Data Streaming:** Integrating real-time data streaming capabilities to continuously update the model with the latest market information can enhance its responsiveness to changing market conditions and improve the accuracy of predictions.

7. **Domain Expertise Integration:** Collaborating with domain experts, such as financial analysts or investment professionals, to incorporate their knowledge, insights, and heuristics into the model development process can augment the predictive power and accuracy of the system.

By pursuing these avenues, we can further enhance the accuracy and reliability of our stock market prediction model, enabling more informed decision-making and potentially improving investment outcomes.

APPENDIX

Packages and data preprocessing:

index5.py > ...

```python
import numpy as np
import pandas as pd
import yfinance as yf
import os
import matplotlib.pyplot as plt
import sys
import tensorflow as tf
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
import xgboost as xgb
# ticker = sys.argv[1]
ticker = "MSFT"
ticker_symbol = f"{ticker}"  # ticker symbol
filename = "data.csv"  # CSV file to save the data
file = "sp500"

if os.path.exists(filename):
    os.remove(filename)
    # stock = pd.read_csv(filename, index_col=0)


stock = yf.Ticker(ticker_symbol)
stock = stock.history(period="max")
stock.to_csv(filename)

stock.index = pd.to_datetime(stock.index)
del stock["Dividends"]
del stock["Stock Splits"]

# print(stock)
# Plot the data
# print("1)")
stock["Tomorrow"] = stock["Close"].shift(-1)

stock["Target"] = (stock["Tomorrow"] > stock["Close"]).astype(int)
stock = stock.loc["1990-01-03":].copy()
```

Training the model:

index5.py > ...

```
#TRAINING THE MODEL
print("1)")
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score

model = RandomForestClassifier(n_estimators=100, min_samples_split=100,random_state=1)

                    #timeseries data so can't use cross validation

train = stock.iloc[:-50]       # 50 days for testing
test = stock.iloc[-50:]        # 50 days for testing

predictors = ["Close", "Volume", "Open", "High", "Low"]
model.fit(train[predictors], train["Target"])



preds = model.predict(test[predictors])     #prediction score
preds = pd.Series(preds, index=test.index)

combined = pd.concat([test["Target"], preds], axis=1)
combined.plot()

precision = precision_score(test["Target"], preds)
precision = round(precision, 4)
print("Precision Score:", precision)
#print(precision)
```

Back testing:

index5.py > ...

```python
#BACK TESTING
print("2)")
def predict(train, test, predictors, model):
    model.fit(train[predictors], train["Target"])
    preds = model.predict(test[predictors])
    preds = pd.Series(preds, index=test.index, name="Predictions")
    combined = pd.concat([test["Target"], preds], axis=1)
    return combined

def backtest(data, model, predictors, start=2500, step=250):
    all_predictions = []

    for i in range(start, data.shape[0], step):
        train = data.iloc[0:i].copy()
        test = data.iloc[i:(i+step)].copy()
        predictions = predict(train, test, predictors, model)
        all_predictions.append(predictions)

    return pd.concat(all_predictions)

# ...

predictions = backtest(stock, model, predictors)
predictions_counts = predictions["Predictions"].value_counts()

precision = precision_score(predictions["Target"], predictions["Predictions"])
print("Precision score after back testing: ", precision)
print("Predictions count:", predictions_counts)

```

Adding additional predictors to improve the model(moving average):

index5.py > ...

```python
#ADDING ADDITIONAL PREDICTORS TO IMPROVE MODEL (MOVING AVERAGES)
print("3)")
horizons = [2,5,60,250,1000]
new_predictors = []

for horizon in horizons:
    rolling_averages = stock.rolling(horizon).mean()

    ratio_column = f"Close_Ratio_{horizon}"
    stock[ratio_column] = stock["Close"] / rolling_averages["Close"]

    trend_column = f"Trend_{horizon}"
    stock[trend_column] = stock.shift(1).rolling(horizon).sum()["Target"]

    new_predictors+= [ratio_column, trend_column]

stock = stock.dropna(subset=stock.columns[stock.columns != "Tomorrow"])
stock.index.name = ""
pd.set_option('display.max_columns', None)

```

Back testing with new predictors:

index5.py > predict

```python
#BACKTESTING WITH NEW PREDICTORS
model = RandomForestClassifier(n_estimators=100, min_samples_split=100, random_state=1)

def predict(train, test, predictors, model):
    model.fit(train[predictors], train["Target"])
    preds = model.predict_proba(test[predictors])[:,1]
    preds[preds >=.6] = 1
    preds[preds <.6] = 0
    preds = pd.Series(preds, index=test.index, name="Predictions")
    combined = pd.concat([test["Target"], preds], axis=1)
    return combined
predictions = backtest(stock, model, new_predictors)
predictions_counts = predictions["Predictions"].value_counts()
precision = precision_score(predictions["Target"], predictions["Predictions"])

print("Precision score after considering the moving averages: ", precision)
print(precision)
print(predictions_counts)

```

Define and train LSTM model:

index5.py > ...

```python
# Define and train the LSTM model
print("4)")
train_x = train[predictors].values
model = Sequential()
model.add(
    LSTM(units=50, return_sequences=True, input_shape=(train_x.shape[1], 1))
)
model.add(LSTM(units=50))
model.add(Dense(1, activation="sigmoid"))
model.compile(loss="binary_crossentropy", optimizer="adam")

model.fit(train[predictors].values, train["Target"].values, epochs=20, batch_size=32)

# Evaluate LSTM model on test set
test_y_pred_probs = model.predict(test[predictors].values)

# Convert probabilities to binary predictions using a threshold (e.g., 0.5)
threshold = 0.5
test_y_pred = (test_y_pred_probs > threshold).astype(int)

# Precision score
precision = precision_score(test["Target"].values, test_y_pred)
print("Precision Score (LSTM):", precision)

# Prediction counts
predictions_counts = pd.Series(test_y_pred.flatten()).value_counts()

# Print prediction counts in the desired format
print("Predictions")
for label, count in predictions_counts.items():
    print(f"{float(label)}: {count}")
```

Training and testing data:

index5.py > ...

```
175  # Define training and testing data
176  train = stock.iloc[:-50]
177  test = stock.iloc[-50:]
178  predictors = ["Close", "Volume", "Open", "High", "Low"]
179
180  # Create XGBoost model
181  model = xgb.XGBClassifier(
182      objective="binary:logistic",
183      n_estimators=200,
184      min_child_weight=1,
185      max_depth=5,
186      learning_rate=0.1,
187      random_state=1,
188  )
189
190  # Train the model
191  model.fit(train[predictors], train["Target"])
192
193  # Make predictions
194  preds_probs = model.predict_proba(test[predictors])[:, 1]
195  preds = (preds_probs > 0.5).astype(int)
196  preds = pd.Series(preds, index=test.index, name="Predictions")
197
198  # Calculate precision score and prediction counts
199  combined = pd.concat([test["Target"], preds], axis=1)
200  precision = precision_score(test["Target"], preds)
201  predictions_counts = preds.value_counts()
202
203  # Print prediction counts in the desired format
204  print("Precision Score (XGBoost):", precision)
205  print("Predictions")
206  for label, count in predictions_counts.items():
207      print(f"{int(label)}: {count}")
208
209
```

Reference

Cabitza, F., Locoro, A., & Banfi, G. (2018). Machine learning in orthopedics: a literature review. Frontiers in Bioengineering and Biotechnology, 6, 75.

Chavan, P. S., & Patil, S. T. (2013). Parameters for stock market prediction. International Journal of Computer Technology and Applications, 4(2), 337.

Chiu, D. Y., & Chen, P. J. (2009). Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm. Expert Systems with Applications, 36(2), 1240-1248. Chong, E., Han, C., & Park, F. C. (2017).

Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications, 83, 187-205.

Dai, W., Wu, J. Y., & Lu, C. J. (2012). Combining nonlinear independent component analysis and neural network for the prediction of Asian stock market indexes. Expert systems with applications, 39(4), 4444-4452.

Das, S. P., & Padhy, S. (2012). Support vector machines for prediction of futures prices in Indian stock market. International Journal of Computer Applications, 41(3).

Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. The Journal of Finance and Data Science, 2(1), 42-57.

Enke, D., & Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. Expert Systems with applications, 29(4), 927- 940.

https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1435&context=jitim

https://legacy.reactjs.org/docs/getting-started.html

https://www.geeksforgeeks.org/machine-learning/