Exploratory Data Analysis and Sentiment Analysis Step

Introduction:

This document will guide you through the process of analyzing stock market data and sentiment analysis of news headlines using Python. The analysis involves using various Python libraries to download stock data, perform statistical analysis, visualize trends, and analyze sentiment from news headlines.

Prerequisites:

1. Python Installation: Ensure Python is installed on your computer. You can download it from [Python's official website](https://www.python.org/downloads/).

2. Installing Required Libraries: Open a command prompt (or terminal) and run the following command to install necessary Python libraries:

```
pip install yfinance pandas matplotlib seaborn nltk wordcloud
```

If any library isn't installed, you can install it by uncommenting `#!pip install yfinance` and running the script.

3. Download the Script: Download the Python script containing the code from [GitHub](https://github.com/RansomJunior/gcp-algorithmic-triad-nustzw-project2023-N02212644N.git).

Steps to Execute:

Step 1: Downloading and Analyzing Stock Market Data

1. Open the Python Script:

- Open the Python script (`analysis_script.py`) in a text editor or an integrated development environment (IDE) like VS Code, PyCharm, or Jupyter Notebook.

2. Understand the Code:

- Review the script to understand its purpose and structure. It includes sections for importing libraries, defining stock tickers, setting date ranges, downloading data, performing exploratory data analysis (EDA), plotting graphs, and analyzing returns.

3. Execute the Script:

Tafadzwa RJ Mheuka

N02212644N

- Run the script by executing it in your Python environment. You can do this by:

- Command Line: Navigate to the directory containing the script and run `python analysis_script.py`.

- IDE: Open the script in your preferred IDE and execute it using the "Run" or "Execute" command.

4. View Results:

- After execution, the script will generate several outputs:

- Summary statistics and graphs showing closing prices and daily returns of stocks.

- A heatmap showing correlations between macroeconomic indicators (if provided).

- Excel files (`summary_statistics_prices.xlsx` and `returns_summary.xlsx`) containing detailed statistical summaries.

Step 2: Analyzing Sentiment from News Headlines

1. Navigate to Sentiment Analysis Section:

- Scroll down or search for the section titled "Sentiment Analysis Model" in the Python script.

2. Understand Sentiment Analysis Model:

- This section uses the `nltk` library's Vader sentiment analyzer to evaluate sentiment (positive, negative, neutral) of news headlines related to selected stocks.

3. Execute Sentiment Analysis:

- Follow similar steps as in Step 1 to execute this section of the script. Ensure your Python environment has internet access to fetch real-time news headlines.

4. View Sentiment Analysis Results:

- After execution, the script will output:

- A DataFrame (`sentiment_df`) summarizing sentiment (positive, negative, neutral) and corresponding news headlines for each stock.

- Bar charts showing the distribution of sentiment categories.

- A word cloud visualizing common words from news headlines.

Risk Assessment Model Manual

This manual provides a step-by-step guide to using the Risk Assessment Model for analyzing stock data and predicting risk levels using machine learning techniques.

Tafadzwa RJ Mheuka

N02212644N

Step 1: Installing Required Packages

Before running the code, ensure you have installed the necessary Python packages. You can install them using pip, a package manager for Python:

```
pip install yfinance pandas ta xgboost matplotlib seaborn scikit-learn nltk wordcloud
```

Step 2: Importing Necessary Libraries

Open a Python script or Jupyter notebook and start by importing the required libraries. These libraries provide functions for data handling, technical analysis, machine learning model training, and visualization.

```
import yfinance as yf

import pandas as pd import

ta

from sklearn.model_selection import train_test_split, RandomizedSearchCV from

sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import xgboost as xgb import matplotlib.pyplot as plt

from sklearn.metrics import roc_curve, precision_recall_curve, auc from

sklearn.calibration import calibration_curve
```

Step 3: Defining the RiskAssessmentModel Class

The `RiskAssessmentModel` class is defined to encapsulate methods for downloading stock data, adding technical indicators, training a machine learning model, evaluating the model, and plotting evaluation metrics.

```
class RiskAssessmentModel:     def __init__(self,

tickers, start_date, end_date):
```

```
        self.tickers = tickers
    self.start_date = start_date
    self.end_date = end_date


    def download_data(self):
        # Method to download historical stock data from Yahoo Finance


    def add_technical_indicators(self, data):
        # Method to add technical indicators to the stock data


    def merge_data(self, stock_data, macro_data):
        # Method to merge stock data with macroeconomic data


    def prepare_data(self, merged_data):
        # Method to prepare data for training by defining risk levels


    def train_model(self, X, Y):
        # Method to train an XGBoost classifier with hyperparameter tuning


    def evaluate_model(self, model, X_train, Y_train, X_val, Y_val, X_test, Y_test):
        # Method to evaluate the trained model on training, validation, and testing sets


    def plot_metrics(self, model, X_train, Y_train, X_val, Y_val, X_test, Y_test,
    train_predictions, val_predictions, test_predictions):
        # Method to plot evaluation metrics: accuracy, feature importance, ROC curve, calibration curve,
    and precision-recall curve


    def risk_assessment_pipeline(self):
```

Method to execute the entire risk assessment pipeline: download data, add indicators, merge data, prepare data, train model, evaluate model, and plot metrics

```

Step 4: Instantiating the Model and Running the Pipeline

Instantiate the `RiskAssessmentModel` class with a list of stock tickers and define the start and end dates for historical data retrieval. Then, execute the `risk_assessment_pipeline()` method to perform the entire analysis pipeline.

```python

# List of stock tickers and date range

tickers = [...] start_date = '2017-01-

01' end_date = '2024-01-01'

# Instantiate the RiskAssessmentModel

risk_assessment_model = RiskAssessmentModel(tickers, start_date, end_date)

# Execute the risk assessment pipeline risk_assessment_model.risk_assessment_pipeline()

```

Step 5: Interpreting the Results

After executing the pipeline, the model will:

- Download historical stock data and macroeconomic data.
- Add technical indicators to the stock data.
- Merge the stock data with macroeconomic data.
- Prepare the data by defining risk levels based on historical returns.
- Train an XGBoost model to predict risk levels.
- Evaluate the model's performance on training, validation, and testing datasets.
- Plot various evaluation metrics including accuracy, feature importance, ROC curve, calibration curve, and precision-recall curve.

Step 6: Viewing and Saving Outputs
```

Throughout the process, the model will save intermediate and final outputs to Excel files (`stock_data.xlsx`, `merged_data.xlsx`, `prepared_data.xlsx`, `stock_data_with_scores.xlsx`). These files contain processed data, risk assessments, and model predictions.

Portfolio Optimization Before Sentiment Adjustment:

Manual Document

This manual provides a step-by-step guide on using Python code to perform portfolio optimization based on historical stock data and risk-return metrics. The process involves loading data, defining a reinforcement learning (RL) environment, training with a random policy, and devising a buy and sell strategy for selected stocks.

Step 1: Setting Up Environment

1. Install Required Packages: Ensure Python is installed along with necessary libraries. You can install them using pip:

```
pip install numpy pandas matplotlib
```

2. Import Libraries: Open a Python script or Jupyter notebook and start by importing necessary libraries:

```
import numpy as np    import
pandas as pd    import
matplotlib.pyplot as plt
```

Step 2: Loading and Preparing Data

1. Load Data: Load the preprocessed data containing stock tickers, daily returns, and risk scores from an Excel file (`stock_data_with_scores.xlsx`):

```python
df = pd.read_excel('stock_data_with_scores.xlsx')
df = df.groupby('Ticker').agg({'Daily Return': 'mean', 'Risk_Scores': 'mean'}).reset_index()
```

2. Normalize Features: Normalize the 'Daily Return' and 'Risk_Scores' columns for standardization:

```python
df['Daily Return'] = (df['Daily Return'] - df['Daily Return'].mean()) / df['Daily Return'].std()
df['Risk_Scores'] = (df['Risk_Scores'] - df['Risk_Scores'].mean()) / df['Risk_Scores'].std()
```

Step 3: Defining the PortfolioEnvironment Class

1. Define RL Environment: Create a class `PortfolioEnvironment` that defines methods for managing the portfolio state, computing portfolio metrics, and executing portfolio actions based on selected stocks:

```python
class PortfolioEnvironment:
    def __init__(self, data):
        # Initialization and state setup

    def reset(self):
        # Reset environment to initial state

    def step(self, action):
        # Perform portfolio action and calculate rewards

    def get_portfolio_metrics(self):
        # Calculate and return portfolio metrics
```

Step 4: Training and Strategy Development

1. Training Loop: Implement a training loop where the RL agent (for demonstration purposes, a random policy is used) interacts with the environment over multiple episodes:

```python
env = PortfolioEnvironment(df)
state = env.reset()
for _ in range(1000):  # Train for 1000 episodes
    action = np.random.rand(len(df))  # Random policy
    next_state, reward, done, _ = env.step(action)
    if done:
        break
```

2. Select Top Stocks: Identify the top 5 stocks in the portfolio based on the action values:

```python
top_5_indices = np.argsort(action)[-5:]
top_5_stocks = df.loc[top_5_indices]['Ticker'].values
```

3. Buy and Sell Strategy: Develop a buy and sell strategy based on volatility and Sharpe ratio thresholds for the selected 5 stocks:

```python
volatility_threshold = np.percentile(env.portfolio_volatility, 25)
sharpe_ratio_threshold = np.percentile(env.portfolio_shape_ratio, 75)

# Determine stocks to buy and sell based on thresholds
buy_indices = np.where((selected_stocks_volatility < volatility_threshold) & (selected_stocks_sharpe_ratio > sharpe_ratio_threshold))[0]
sell_indices = np.where((selected_stocks_volatility > volatility_threshold) & (selected_stocks_sharpe_ratio < sharpe_ratio_threshold))[0]
```

Tafadzwa RJ Mheuka

N02212644N

Sentiment Analysis Model: Manual Document

This manual provides a step-by-step guide on using Python code to perform sentiment analysis on financial news headlines and subsequently adjust stock data based on the sentiment scores obtained.

Step 1: Setting Up Environment

1. Install Required Packages: Ensure Python is installed along with necessary libraries. You can install them using pip:

```
pip install pandas yfinance nltk keras matplotlib scikit-learn
```

2. Import Libraries: Open a Python script or Jupyter notebook and start by importing necessary libraries: ```python import pandas as pd import yfinance as yf from nltk.sentiment.vader import SentimentIntensityAnalyzer from sklearn.model_selection import train_test_split from sklearn.preprocessing import LabelEncoder from keras.preprocessing.text import Tokenizer from keras.preprocessing.sequence import pad_sequences from keras.models import Sequential from keras.layers import LSTM, Dense, Embedding from keras.callbacks import EarlyStopping import matplotlib.pyplot as plt from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

```

Step 2: Initializing the SentimentAnalysisModel Class

1. Define the Class: Create a class `SentimentAnalysisModel` that will handle downloading news, analyzing sentiment, training a sentiment analysis model, predicting sentiment scores, adjusting stock data, and saving results to Excel.

```python class

SentimentAnalysisModel:

def __init__(self, tickers):

self.tickers = tickers

```

Tafadzwa RJ Mheuka

N02212644N
```

Step 3: Downloading News and Analyzing Sentiment

1. Download News: Implement a method `download_news_and_analyze_sentiment()` to fetch news headlines for given stock tickers using Yahoo Finance API (yfinance) and analyze sentiment using NLTK's Vader sentiment analyzer.

```python
def download_news_and_analyze_sentiment(self):
    sid = SentimentIntensityAnalyzer()
    ticker_list = []
    sentiment_list = []
    headlines_list = []

    for ticker in self.tickers:
        ticker_data = yf.Ticker(ticker)
        news = ticker_data.news
        if news:
            headlines = [headline['title'] for headline in news]
            sentiments = [sid.polarity_scores(headline)['compound'] for headline in headlines]
            avg_sentiment = sum(sentiments) / len(sentiments)
            sentiment = "Positive" if avg_sentiment > 0 else "Negative" if avg_sentiment < 0 else "Neutral"
            ticker_list.append(ticker)
            sentiment_list.append(sentiment)
            headlines_list.append(headlines)
        else:
            ticker_list.append(ticker)
            sentiment_list.append("Not available")
            headlines_list.append([])

    df = pd.DataFrame({
        'Ticker': ticker_list,
```

```
            'Sentiment': sentiment_list,
            'Headlines': headlines_list
        })
        return
df
```

Step 4: Training the Sentiment Analysis Model

1. Train the Model: Implement a method `train_sentiment_model()` to preprocess text data, tokenize headlines, train an LSTM-based sentiment analysis model using Keras, and evaluate its performance.

Step 5: Saving Results to Excel

1. Save to Excel: Implement a method `save_to_excel()` to save the adjusted stock data to an Excel file.

```
    def save_to_excel(self, df, filename):
df.to_excel(filename, index=False)
```

Portfolio Optimization After Sentiment Adjustment

Execution Instructions:

Prerequisites: Ensure you have Python installed along with necessary libraries such as numpy, pandas, matplotlib, and others used in the code.

Data Preparation: Prepare your stock data in an Excel file (stock_data_with_adjustments.xlsx) with adjusted daily returns and risk scores.

Execution: Copy and paste the provided Python code into a script file (e.g., portfolio_optimization.py) and execute it using Python.

Output: The script will display the top 5 stocks recommended for your portfolio, allocation strategies (buy and sell), portfolio metrics, and visualizations (portfolio allocation and efficient frontier).

Tafadzwa RJ Mheuka

N02212644N