Phase 3: Submission Document

Project Title: *Stock Price Prediction*

Phase 3: Development part 1

Topic: Stock Price Prediction by loading and preprocessing the dataset

Stock Price Prediction

Introduction:

Stock price prediction is a critical and challenging area within the field of financial analysis and investing. In an era of rapidly changing markets and an abundance of available data, the ability to forecast the future movements of stocks has become a valuable skill for traders, investors, and financial professionals.

Predicting stock prices involves the use of various techniques and methodologies, including fundamental analysis, technical analysis, and increasingly, machine learning and artificial intelligence.

Data Retrieval:

- ☐ Obtain historical stock price data from a reliable source. Common sources include Yahoo Finance, Google Finance, or dedicated financial APIs.

- ☐ You can also use libraries like `pandas` in Python to retrieve data from online sources. For example, you can use `pandas_datareader` to fetch financial data.

Data Cleaning:

- ☐ Remove or handle missing data: Check for missing values in your dataset and decide how to handle them (e.g., fill with the previous value, remove rows, or interpolate)

- ☐ Check for and handle outliers: Outliers can greatly affect predictive models. You might choose to remove them or transform the data to be less sensitive to outliers

Feature Engineering:

- ☐ Create relevant features: You may want to engineer features such as moving averages, volatility measures, technical indicators (e.g., RSI, MACD), or sentiment scores from news articles and social media

- ☐ Lag features: Create lag features by shifting the price data for various time intervals. These can capture trends and autocorrelation in the data.

Normalization/Scaling:

Scale or normalize the data: Feature scaling is often necessary, especially if you're using algorithms sensitive to the scale of input data, like neural networks.

Train-Test Split:

Split your dataset into a training set and a testing set. Typically, you will use a larger portion for training and a smaller portion for testing, e.g., 80% for training and 20% for testing. You should ensure that the data is chronologically split to simulate real-world scenarios.

Time Series Specific Steps(Optional):

If you are working with time series data, you might need to account for seasonality, trends, and stationarity. You can use techniques like differencing or decomposition to make the data more stationary.

Save Preprocessed Data:

Save the preprocessed data to a format that's easy to work with for modeling (e.g., CSV, HDF5, or a database).

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load data
data = pd.read_csv('stock_price_data.csv')

# Data cleaning
data.dropna(inplace=True)
data['Date'] = pd.to_datetime(data['Date'])

# Feature engineering
data['SMA_50'] = data['Close'].rolling(window=50).mean()

# Train-test split
train_size = int(0.8 * len(data))
train_data, test_data = data[:train_size], data[train_size:]

# Save preprocessed data
```

```
train_data.to_csv('train_data.csv', index=False)
test_data.to_csv('test_data.csv', index=False)
```

Conclusion:

Once you have your data preprocessed and split, you can use it to train and evaluate various stock price prediction models, such as linear regression, time series models (e.g., ARIMA, LSTM), or machine learning algorithms. The choice of model depends on the nature of your data and the specific problem you're trying to solve.