INNOVATION

Summary of stock price prediction:

The financial markets are a complex and dynamic environment where investors seek to maximize their returns while managing risks. One of the most challenging aspects of investing is predicting the future performance of stocks accurately. The ability to forecast stock prices with precision is highly sought after, as it can enable investors to make well-informed decisions and optimize their investment strategies. In this project, we aim to develop a predictive model that leverages historical market data to forecast stock prices. This model will serve as a valuable tool for investors looking to enhance their decision-making processes in the stock market.

Dataset and its details:

The dataset is taken from Kaggle ,it contains life time stocks data from 3/13/1986 to 12/10/2019 and it contains 7 columns including dates ,opening ,high ,low ,closing, adj-close ,volume.

The dataset can be used to train machine learning models to predict future stock prices. Kaggle provides several notebooks that use this dataset to predict stock prices using advanced deep learning techniques like CNN-LSTM or attention mechanisms for improved accuracy in predicting stock prices. These datasets can be used as a starting point for building your own stock price prediction model.

The dataset used in our project is from https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

Libraries used in our projects:

Numpy :

*In the terminal, use the command pip install numpy to install numpy package.

*Once the package is installed successfully, it is ready to import in our python file. Notice the version is supported too.

*To import the numpy in our python file use the command ,import numpy as np.

Pandas :

*In the terminal, use the command pip install pandas to install pandas package.

* Once the package is installed successfully, it is ready to import in our pandas file. Notice the version is supported too.

*To import the pandas in our python file use the command ,import pandas in pd.

Matplotlib:

*In the terminal, use the command pip install pandas to install matplotlib package.

* Once the package is installed successfully, it is ready to import in our matplotlib file. Notice the version is supported too.

*To import the matplotlib in our python file use the command , import matplotlib.pyplot as plt.

Sklearn:

*To install the train_test_split function from the sklearn.model_selection module,first install the scikit-learn package, which is also known as sklearn.

* Use the pip command to install scikit-learn `pip install -U numpy scipy scikit-learn`.

*After installing scikit-learn, verify your installation by running the following commands in your terminal:

`python -m pip show scikit-learn # to see which version and where scikit-learn is installed`

`python -m pip freeze # to see all packages installed in the active virtual environment`

`python -c "import sklearn.show_versions()" # to see the versions of scikit-learn and its dependencies`

*Once you have successfully installed scikit-learn, you can import the train_test_split function from the sklearn.model_selection module using the following command:

`from sklearn.model_selection import train_test_split`

`from sklearn.linear_model import LogisticRegression`

`from sklearn.metrics import accuracy_score, classification_report, confusion_matrix`

*First command allows you to split your data into random train and test subsets, which are useful for training and testing your machine learning models.

*Second command, from sklearn.linear_model import LogisticRegression is used for performing logistic regressionuses of the commands from sklearn.linear_model import LogisticRegression.

*Third command is used to classify the dataset and used to find the accuracy.

Training and Testing dataset:

The training data set is used to train the model, while the test data set is used to evaluate the prediction performance. The training data set is a subset of the original dataset that is used to fit the machine learning model. The test data set is another subset of the original data that is independent of the training dataset and is used to evaluate the model's performance and ensure that it can generalize well with new or unseen datasets.

The quality of the training and testing datasets is critical for guiding algorithms effectively and evaluating their performance accurately . The better the quality of the training data, the better will be the performance of the model .

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Generate some sample data
np.random.seed(0)
data = {
   'Exam1': np.random.rand(100) * 100,
   'Exam2': np.random.rand(100) * 100,
   'Admitted': np.random.randint(2, size=100)
}
df = pd.DataFrame(data)
print(df)
#
# # Split the data into features (X) and target (y)
X = df[['Exam1', 'Exam2']]
y = df['Admitted']
print(X)
print(y)
#
# # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
#
```

```
# # Create a logistic regression model
model = LogisticRegression()
#
# # Fit the model to the training data
model.fit(X_train, y_train)
#
# # Make predictions on the test data
y_pred = model.predict(X_test)
print("------------------")
print(y_pred)
```

Accuracy of the dataset:
CNN-LSTM is a machine learning architecture that combines Convolutional Neural Network (CNN) layers with Long Short-Term Memory (LSTM) layers to support sequence prediction problems with spatial inputs. The CNN layers are used for feature extraction on input data, while the LSTM layers are used for sequence prediction.

The CNN-LSTM architecture is particularly useful for visual time series prediction problems. The CNN-LSTM model can be used in a hybrid model with an LSTM backend where the CNN is used to interpret subsequences of input that together are provided as a sequence to an LSTM model to interpret.

In our project,stock price prediction CNN-LSTM mechanism is used to find accuracy.

Code:

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import LSTM, Dense, Dropout, Flatten, Conv1D,
MaxPooling1D


# Load your stock price dataset using pandas

# Replace 'your_dataset.csv' with your actual dataset file path

data = pd.read_csv('your_dataset.csv')

data = data[['Date', 'Close']]  # Assuming you have 'Date' and 'Close'
columns


# Data preprocessing

data['Date'] = pd.to_datetime(data['Date'])

data.set_index('Date', inplace=True)

data = data.dropna()


# Normalize the data

scaler = MinMaxScaler()

data['Close'] = scaler.fit_transform(data['Close'].values.reshape(-1, 1))


# Split the data into training and testing sets

train_size = int(len(data) * 0.8)

train_data = data[:train_size]

test_data = data[train_size:]
```

```
# Function to create sequences for the CNN-LSTM model
def create_sequences(data, sequence_length):
    sequences = []
    target = []
    for i in range(len(data) - sequence_length):
        seq = data[i:i+sequence_length]
        target_val = data.iloc[i+sequence_length]
        sequences.append(seq)
        target.append(target_val)
    return np.array(sequences), np.array(target)

sequence_length = 10  # You can adjust this as needed
X_train, y_train = create_sequences(train_data, sequence_length)
X_test, y_test = create_sequences(test_data, sequence_length)

# Build the CNN-LSTM model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(sequence_length, 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(50, return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
```

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32)

# Evaluate the model
train_loss = model.evaluate(X_train, y_train, verbose=0)
test_loss = model.evaluate(X_test, y_test, verbose=0)

print(f"Train Loss: {train_loss}")
print(f"Test Loss: {test_loss}")

# Make predictions
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# Inverse transform the predictions to get actual prices
train_predictions = scaler.inverse_transform(train_predictions)
test_predictions = scaler.inverse_transform(test_predictions)

# Plot the predictions
plt.figure(figsize=(12, 6))
plt.plot(data.index[:len(train_predictions)], train_predictions,
label='Train Predictions', color='blue')

plt.plot(data.index[len(train_predictions) + sequence_length - 1:],
test_predictions, label='Test Predictions', color='red')

plt.plot(data.index, data['Close'], label='Actual Prices', color='green')

plt.legend()

plt.xlabel('Date')

plt.ylabel('Stock Price')

plt.title('Stock Price Prediction Using CNN-LSTM')

plt.show()
```

Therefore, this above mechanism is used for predicting the stock price in the project.