TCS Stock Data Analysis and Prediction

Introduction

Tata Consultancy Services (TCS) is a globally recognized IT company with a market capitalization exceeding $200 billion. This project analyzes the historical data of TCS stock to provide insights into its behavior, identify trends, and forecast future stock prices using machine learning techniques.

Objectives

- Analyze TCS stock data to gain meaningful insights.
- Identify patterns and trends in stock behavior.
- Build predictive models to forecast stock prices.

Dataset Overview

The dataset consists of historical TCS stock trading data with the following attributes:

1. **Date**: Trading date.
2. **Open**: Opening stock price.
3. **High**: Highest stock price during the day.
4. **Low**: Lowest stock price during the day.
5. **Close**: Closing stock price.
6. **Volume**: Number of shares traded.
7. **Dividends**: Dividends paid on the stock.
8. **Stock Splits**: Number of stock splits.

Tools and Technologies

- **Languages**: Python, SQL, Excel
- **Tools**: VS Code, Jupyter Notebook
- **Libraries**: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, TensorFlow/Keras

---

Workflow

1. Data Preprocessing

- Handled missing values to ensure data integrity.
- Converted date column to datetime format for time-series analysis.
- Scaled numerical features using MinMaxScaler for machine learning models.

2. Exploratory Data Analysis (EDA)

- Visualized stock price trends using line plots.
- Analyzed volatility using rolling statistics and candlestick charts.
- Correlation analysis revealed relationships between features such as volume and closing prices.

3. Feature Engineering

- Generated new features, including moving averages (5-day, 20-day).
- Added technical indicators like RSI (Relative Strength Index) and MACD (Moving Average Convergence Divergence).

4. Machine Learning Models

Models Implemented:

- **Linear Regression**: A simple baseline model.
- **Random Forest Regressor**: Improved prediction accuracy using ensemble techniques.
- **LSTM (Long Short-Term Memory)**: Used for time-series forecasting to capture sequential dependencies in the data.

Model Evaluation Metrics:

- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared (^{2})

5. Insights and Conclusions

- Identified seasonal trends and patterns in stock behavior.
- Machine learning models demonstrated strong predictive performance, with LSTM providing the most accurate forecasts for future stock prices.

Challenges and Solutions

Challenges:

- High volatility in stock prices affected prediction accuracy.
- Limited data for certain periods reduced model generalization.

Solutions:

- Used rolling averages and technical indicators to smooth volatility.
- Augmented dataset with synthetic data for model training.

Results and Deliverables

1. **Preprocessed Dataset**: Cleaned and enriched dataset ready for analysis.
2. **EDA Results**: Comprehensive visualizations and statistical insights.
3. **Machine Learning Models**: Trained models capable of predicting future stock prices.
4. **Project Report**: Detailed documentation summarizing the methodology and findings.

Code Snippets

```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Load Dataset
data = pd.read_csv('tcs_stock_data.csv')
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)

# Plot Closing Prices
plt.figure(figsize=(10, 6))
plt.plot(data['Close'], label='Closing Prices')
plt.title('TCS Stock Closing Prices')
plt.legend()
plt.show()

# Feature Engineering
data['5-Day MA'] = data['Close'].rolling(window=5).mean()
```

```
data['20-Day MA'] = data['Close'].rolling(window=20).mean()


# Train-Test Split

train_size = int(len(data) * 0.8)

train, test = data[:train_size], data[train_size:]


# Random Forest Regressor

rf = RandomForestRegressor()

rf.fit(train[['Open', 'High', 'Low', 'Volume']], train['Close'])

rf_predictions = rf.predict(test[['Open', 'High', 'Low', 'Volume']])

print(f"RMSE: {np.sqrt(mean_squared_error(test['Close'], rf_predictions))}")


# LSTM Model

model = Sequential([

   LSTM(50, return_sequences=True, input_shape=(train.shape[1], 1)),

   LSTM(50),

   Dense(1)

])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(train.values, train['Close'].values, epochs=10, batch_size=32)
```

Future Scope

- Extend analysis to other stocks in the same sector for comparative insights.
- Incorporate sentiment analysis of news and social media data to enhance predictions.

Acknowledgments

- TCS for being an industry leader and providing inspiration for this project.
- Open-source libraries and resources that supported the implementation.
