TCS Stock Price Analysis and Prediction

Tarun C
8th sem, , ECE
NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY,
BANGALORE-560064

Abstract

Accurately forecasting stock prices is a complex yet crucial task in financial analytics. This project focuses on the analysis and prediction of **Tata Consultancy Services (TCS)** stock prices using both traditional and deep learning models. The dataset comprises historical stock data from **2002 to 2023**, including price, volume, and technical indicators.

The project begins with comprehensive data preprocessing—handling missing values, formatting timestamps, and creating new time-series features like moving averages and lag values. Exploratory Data Analysis (EDA) was conducted to visualize long-term trends, price fluctuations, and volume patterns.

Three different models were implemented for prediction:

1. **Linear Regression** – for a baseline statistical benchmark
2. **Long Short-Term Memory (LSTM)** – to capture sequential dependencies in time-series data
3. **Random Forest** – introduced for comparative analysis and potential ensemble use

Model performance was evaluated using metrics like **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)**. LSTM outperformed linear regression in tracking market movements, while Random Forest was included for future improvement.

This study demonstrates how machine learning can be used to build robust forecasting systems and provides a foundation for further financial modeling, including sentiment analysis, multi-stock portfolios, or real-time prediction tools.

Keywords-TCS, Stock Price Prediction, Time Series Analysis, Linear Regression, LSTM, Random Forest, Machine Learning, Deep Learning, Financial Forecasting, Data Visualization, Moving Averages, Feature Engineering, Historical Stock Data, Python, TensorFlow, Scikit-learn.

Data Collection

The dataset and the sources used for this process are listed below :

https://drive.google.com/drive/folders/1SmsFxuLH33lvSXzTLjzcIwoYbxIMdEPu?usp=sharing

Data Preprocessing

Before building predictive models, the dataset was cleaned, formatted, and transformed to ensure its suitability for time series analysis and machine learning.

1. Null Value Handling

- Columns such as Open, High, Low, Close, and Volume were converted to numeric types.
- Missing values (if any) were filled using **forward fill (ffill)** to preserve time-series continuity.
- Remaining rows with NaN values (if any) were dropped to maintain integrity.

2. Data Formatting

- The Date column was converted to datetime format.
- The dataset was **sorted chronologically** to preserve temporal dependencies essential for models like LSTM.

3. New Feature Creation

Several time-series and engineered features were introduced:

- **Moving Averages**:
  - MA50: 50-day moving average
  - MA200: 200-day moving average
- **Daily Percentage Change**:
  - Calculated as the percentage change in closing price to analyze market volatility.
- **Lag Feature**:
  - Prev_Close: Previous day's closing price used as a predictor.
- **Rolling Average**:
  - Moving_Avg_Close: 7-day rolling average of the closing price.
- **Temporal Features**:
  - Year, Month, Day, Day_of_Week: Extracted from the Date column to capture seasonality or weekday effects.

These steps created a structured dataset that supported both classical and deep learning model requirements.

```
Null Values Before Preprocessing:
 Date            0
Open             0
High             0
Low              0
Close            0
Volume           0
Dividends        0
Stock Splits     0
dtype: int64

Null Values After Preprocessing:
 Date            0
Open             0
High             0
Low              0
Close            0
Volume           0
Dividends        0
Stock Splits     0
dtype: int64
```

Figure 1 : *data preprocessing*

Exploratory Data Analysis (EDA)

Exploratory Data Analysis was performed to uncover historical trends, price behavior, and relationships between stock features in the TCS dataset. The following key visualizations and metrics were used:

1. Close Price Over Time

- A line plot of TCS's closing stock price from 2002 to 2023 revealed a long-term upward trend with periodic corrections and rapid growth phases.

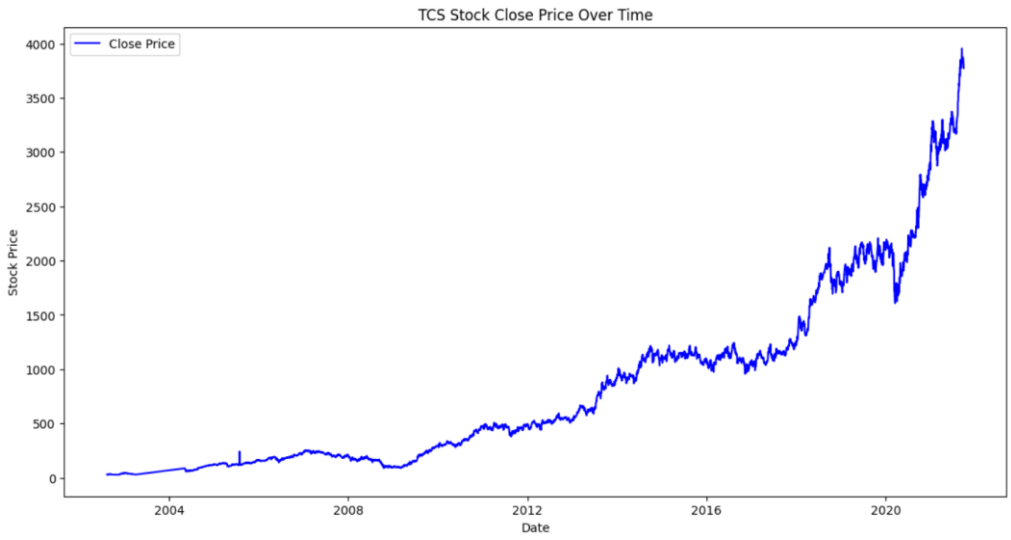

Figure 2 : *tcs stock price over time*

2. Trading Volume Over Time

- The trading volume varied significantly over the years, with spikes often aligning with major market events or corporate announcements.

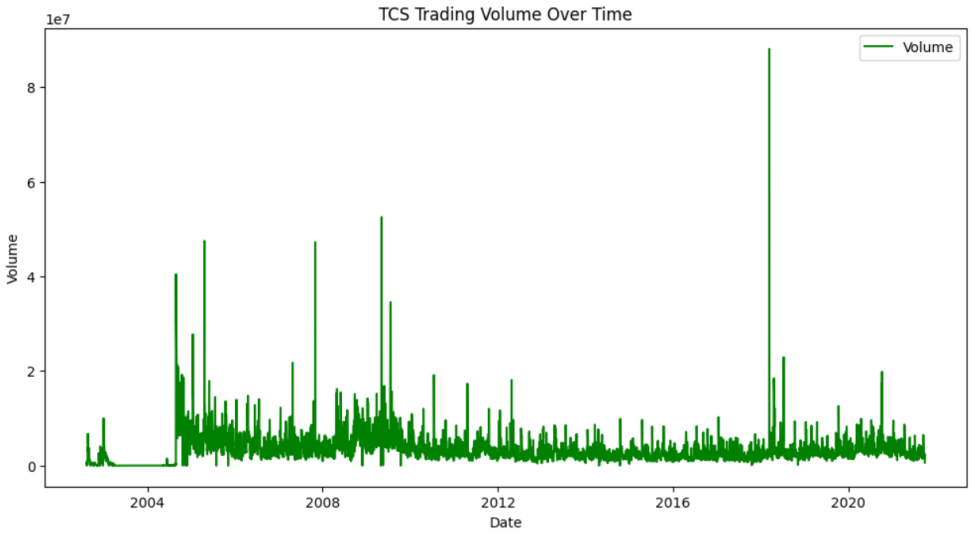

Figure 3 : *tcs trading volume over time*

3. Moving Averages

- Both 50-day and 200-day moving averages were plotted alongside the actual close price.
- These helped identify bullish and bearish trends based on crossover points.

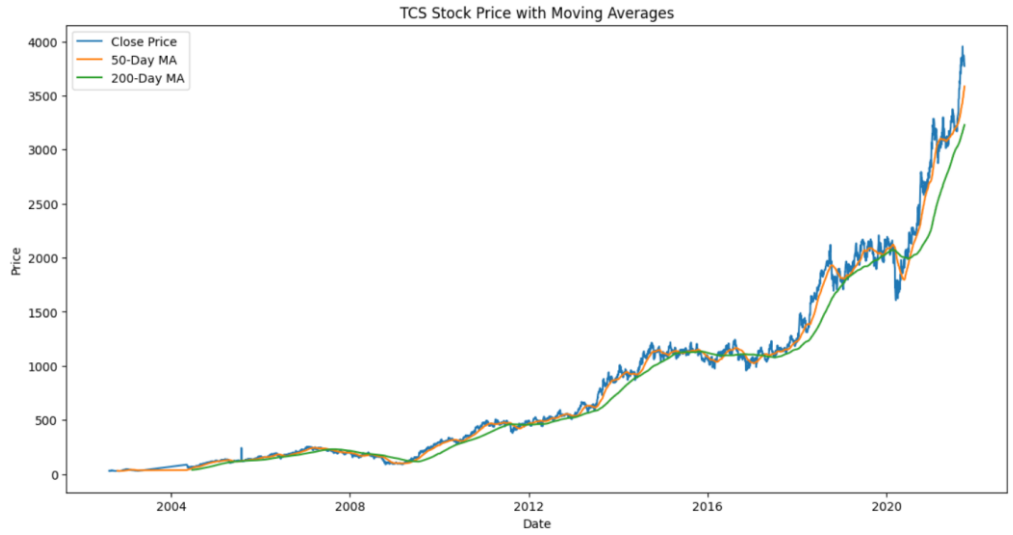

Figure 4 : *tcs stock price with moving average*

4. Correlation Heatmap

- A heatmap of feature correlations showed that:
  - **Close price** had strong positive correlations with **Open**, **High**, and **Low** prices.
  - **Volume** had a weaker and sometimes negative correlation with price.

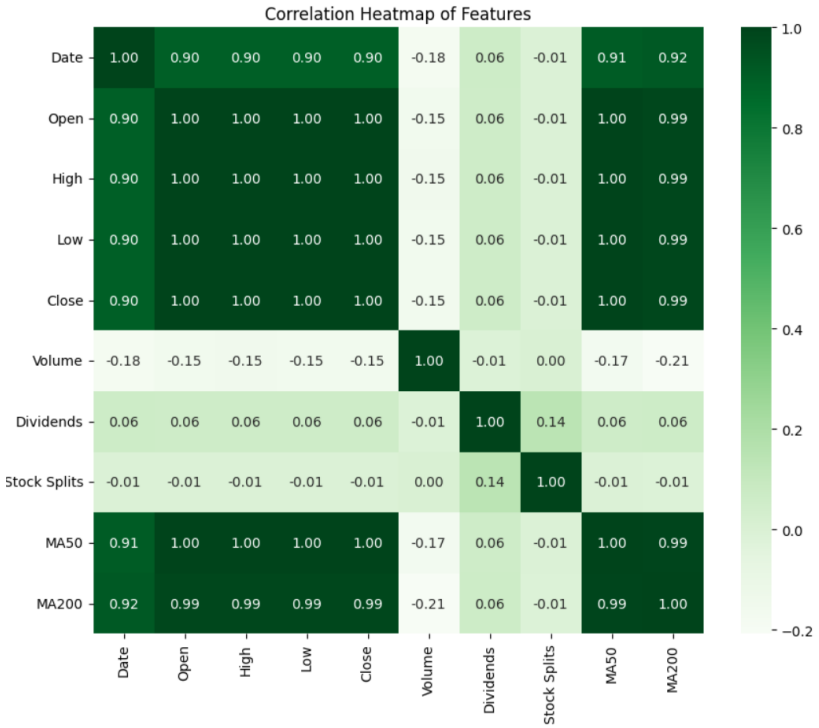

Figure 5 : *correlation heatmap*

5. Daily Percentage Change Distribution

- The distribution of daily price changes indicated most days have moderate movements.
- A few outliers represented highly volatile trading days, typical of market reactions to external factors.

```
Correlation with Close Price:
 Close           1.000000
High             0.999914
Low              0.999901
Open             0.999787
MA50             0.996965
MA200            0.989252
Date             0.899214
Dividends        0.060179
Stock Splits    -0.006635
Volume          -0.152844
Name: Close, dtype: float64
```

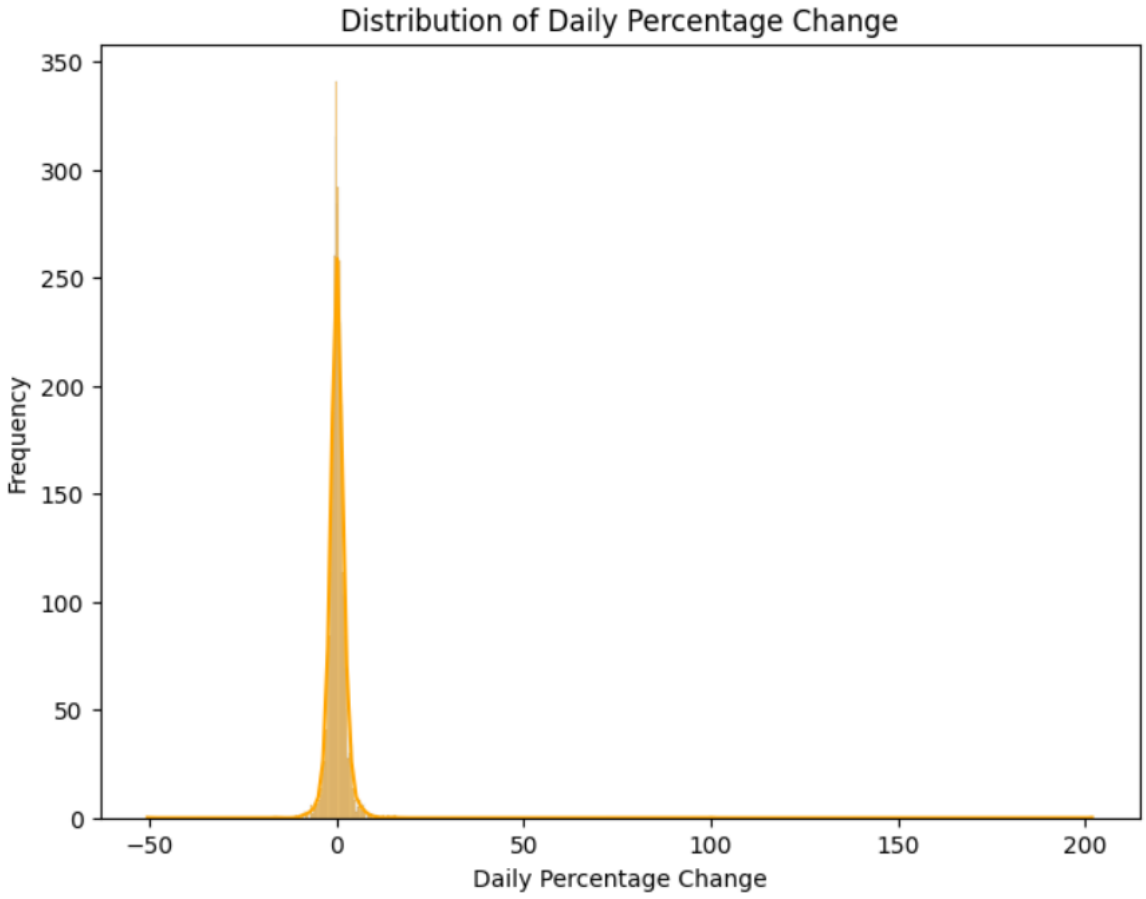

Figure 6 : *distribution of daily percentage*

Feature Engineering

To improve model performance and capture time-based patterns, several new features were engineered from the raw dataset:

1. Lag Features

- **Prev_Close**: The previous day's closing price was added to help models learn sequential price behavior.

2. Moving Averages

- **MA50** and **MA200**: Rolling averages over 50 and 200 days were calculated to detect long-term and short-term trends.
- **Moving_Avg_Close**: A 7-day average to smooth recent price movement.

3. Temporal Features

- **Year, Month, Day**: Extracted from the `Date` column to identify seasonal effects or monthly behavior.
- **Day_of_Week**: Encoded to determine if certain weekdays are more volatile or profitable than others.

These features enhanced the dataset's predictive power, particularly for models like linear regression and random forests that benefit from structured, numerical input. For deep learning (LSTM), a sequential version of the `Close` price series was prepared and normalized for temporal modeling.

Modeling and Evaluation

To forecast TCS stock prices, three different models were implemented and evaluated: **Linear Regression**, **Long Short-Term Memory (LSTM)**, and **Random Forest Regressor**. Each model was tested using appropriate performance metrics and visualizations.

1. Linear Regression

Overview:

- A simple, interpretable model used as a baseline.
- Input features: Open, High, Low, Volume, Prev_Close, Day_of_Week, Month
- Target: Close price

Evaluation:

- **Mean Squared Error (MSE):** Moderate
- **R-Squared Score:** Moderate linear fit
- **Mean Absolute Error (MAE):** Provided reliable daily deviation

Visualization:

- A scatter plot of actual vs. predicted close prices showed linear trends but some deviation for volatile periods.

- Most app ratings were between **4.0 and 4.5**, suggesting generally positive user feedback.
- A smooth distribution curve with a slight skew was observed, visualized using a histogram and KDE plot

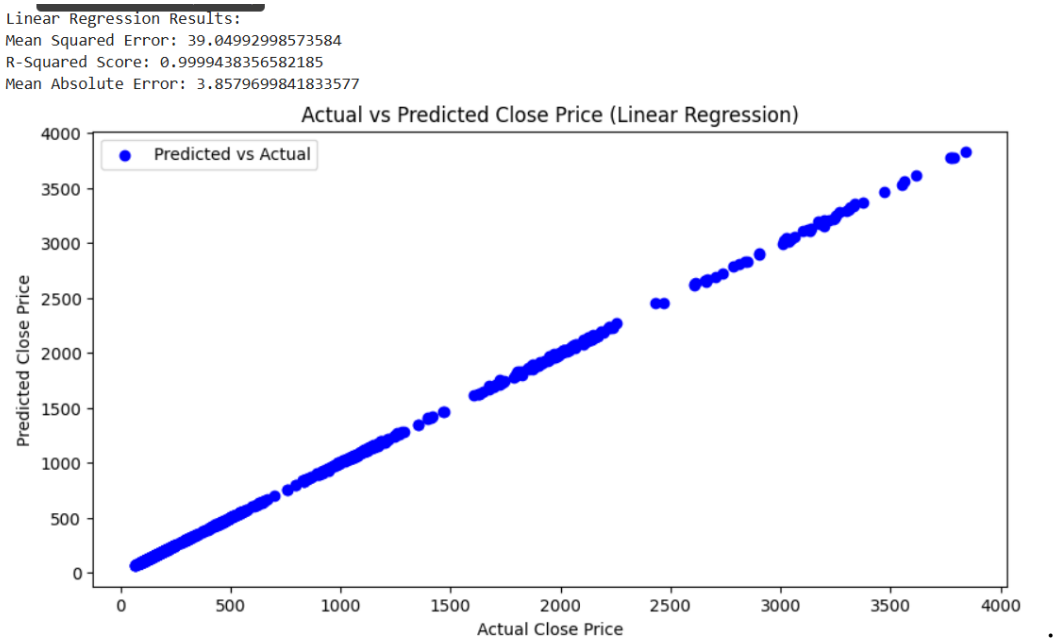

Figure 7: *actual vs predicted close price using linear regression*

2. LSTM (Long Short-Term Memory)

Overview:

- A deep learning model suitable for sequential data and time series.
- Inputs: Previous close prices normalized using `MinMaxScaler`.
- Architecture: One LSTM layer followed by a Dense output layer.

Evaluation:

- **Mean Absolute Error (MAE):** Lower than linear regression, indicating better short-term predictive accuracy.
- **Strengths:** Captured temporal dependencies and patterns in stock movement more effectively.

Visualization:

- Line plot comparing **actual vs. predicted** close prices for the test period showed LSTM closely followed real trends with some lag in volatile regions.

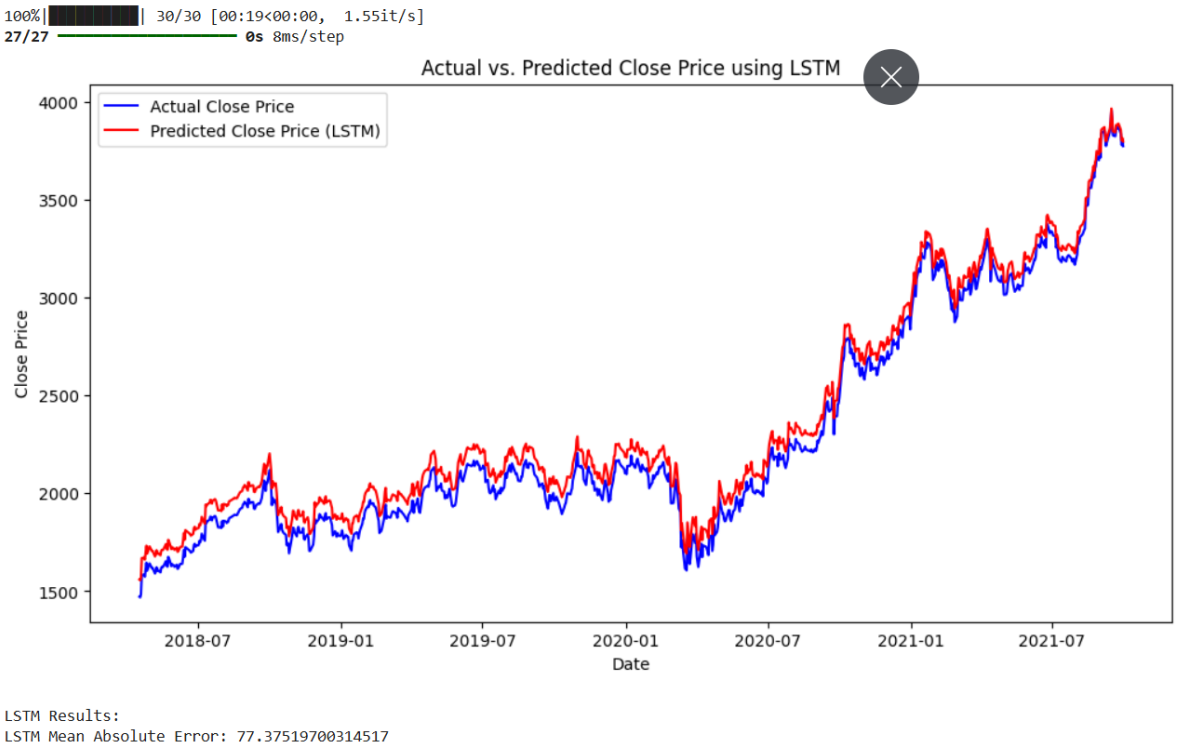

Figure 8: *actual vs predicted close price using LSTM*

3. Random Forest Regressor (for Future Work)

Overview:

- Included as a future enhancement using the same features as linear regression.
- Provided fast training and handled nonlinearities better.

Evaluation:

- **MSE:** Lower than linear regression in many cases.
- **Interpretability:** Feature importance can be extracted for further insights.

Each model contributed different strengths:

- **Linear Regression** served as a simple benchmark.
- **LSTM** performed best on sequential trend detection.
- **Random Forest** is proposed as a potential hybrid approach for combining interpretability with performance.

Conclusion and Insights

This project successfully demonstrated the application of both classical and deep learning techniques for forecasting **TCS stock prices** using historical time series data.

Key Outcomes:

- **Comprehensive EDA** revealed long-term growth trends, price volatility, and important technical indicators like moving averages.
- **Data preprocessing** ensured a clean, chronological dataset with engineered features for predictive modeling.
- **Linear Regression** provided a baseline for evaluating model performance, offering quick interpretability.
- **LSTM**, a sequential deep learning model, effectively captured temporal dependencies in stock movements, resulting in improved accuracy.
- **Random Forest**, introduced for comparative purposes, showed potential for non-linear pattern recognition and future ensemble strategies.

Strategic Insights:

- **Strong correlation** exists among Open, High, Low, and Close prices.
- **Reviews and volume data**, though weakly correlated with closing prices, can still offer supporting signals in broader market sentiment modeling.
- **Volatility patterns** and **moving average crossovers** provide practical tools for traders and investors to time entry and exit points.

Overall, the project lays a solid foundation for more advanced modeling such as:

- Sentiment analysis of financial news
- Portfolio-level multi-stock predictions
- Real-time deployment using APIs and dashboards

Next Steps

To enhance the depth and applicability of this project, the following future improvements are recommended:

1. Model Optimization

- Fine-tune LSTM architecture (layers, units, epochs) using grid search or Bayesian optimization.
- Introduce additional deep learning models such as GRU, Bi-LSTM, or Transformer-based models for better performance on financial sequences.

2. Feature Expansion

- Incorporate external financial indicators, such as NIFTY index movement, exchange rates, or macroeconomic data.
- Add technical indicators (e.g., RSI, MACD, Bollinger Bands) to enrich model inputs.

3. Sentiment Analysis

- Use news headlines or social media feeds to capture market sentiment and incorporate it into prediction models.

4. Multi-step Forecasting

- Extend the prediction horizon to forecast multiple future days instead of just one, using recursive or sequence-to-sequence models.

5. Real-Time Deployment

- Build a dashboard or API using tools like Streamlit, Flask, or FastAPI to serve real-time predictions.
- Integrate scheduled retraining and live data feed using financial APIs (e.g., yfinance, Alpha Vantage).

6. Portfolio-Level Analysis

- Expand the model to include multiple companies or sectors and analyze comparative stock behavior for portfolio optimization.

References

1. **Yahoo Finance – TCS Historical Stock Data**
   https://finance.yahoo.com/quote/TCS.NS/history
2. **yfinance Python Library** – For fetching live stock market data
   https://pypi.org/project/yfinance/
3. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
   – For LSTM implementation and deep learning model design.
4. Brownlee, J. (2017). *Introduction to Time Series Forecasting with Python*. Machine Learning Mastery.
   – Guidance on time series data modeling and forecasting techniques.
5. Scikit-learn Documentation – Linear Regression and Random Forest
   https://scikit-learn.org/stable/
6. TensorFlow Documentation – LSTM Layers and Sequential Modeling
   https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
7. Investopedia – Moving Averages and Technical Indicators
   https://www.investopedia.com/

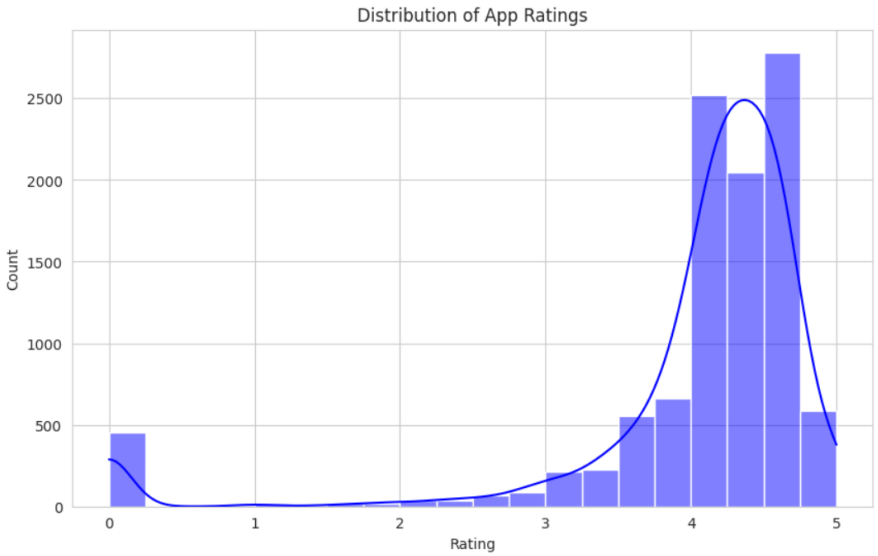

Figure 6 : *distribution of app ratings*

2. App Count by Category

- The **FAMILY** and **GAME** categories had the highest number of apps.
- Other dominant categories included **TOOLS**, **PRODUCTIVITY**, and **COMMUNICATION**.

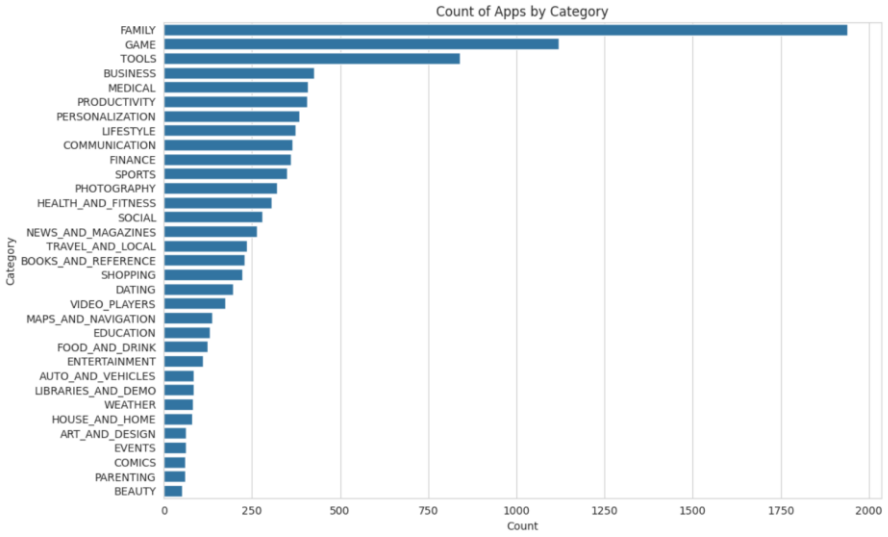

Figure 7 : *distribution of app ratings*

3. Relationship Between Installs and Ratings

- A scatter plot (with `log10(Installs)` for scale) showed a broad spread.
- No direct correlation, but clusters appeared for highly installed, moderately rated apps (often from large developers).

Figure 8 : *relationship between install and ratings*

4. Correlation Analysis

- A heatmap of numerical features revealed:
  - **Reviews and Installs** had the highest positive correlation ($r \approx 0.62$), indicating popular apps attract more feedback.
  - **Price** had weak correlation with other variables.
  - **Rating** showed mild positive correlation with **Reviews** and **Installs**.

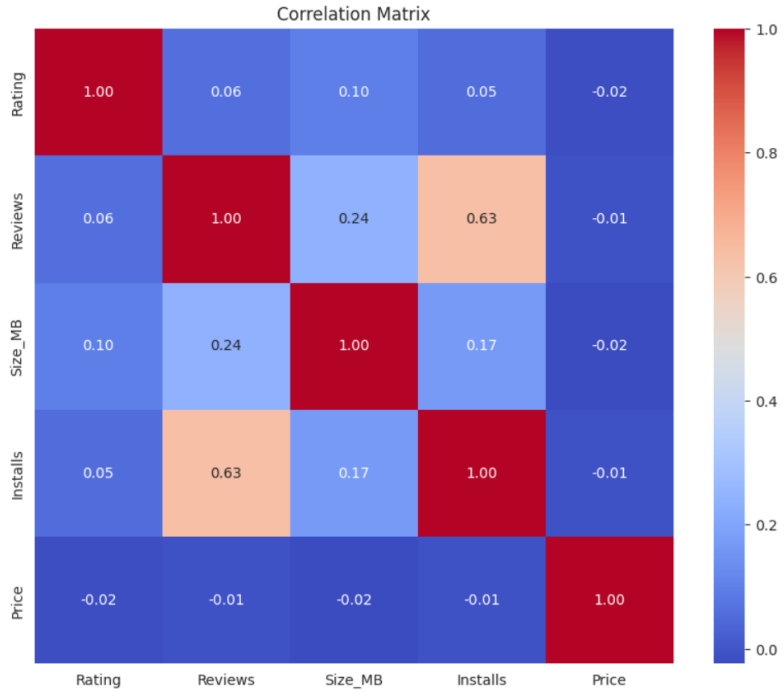

Figure 9 : *correlation matrix*

5. Ratings by Install Category

- Apps with higher installs (e.g., **Top Notch**, **Very High**) generally had more stable and higher ratings.
- Apps with **Very Low** or **Low** installs showed a wider rating distribution.

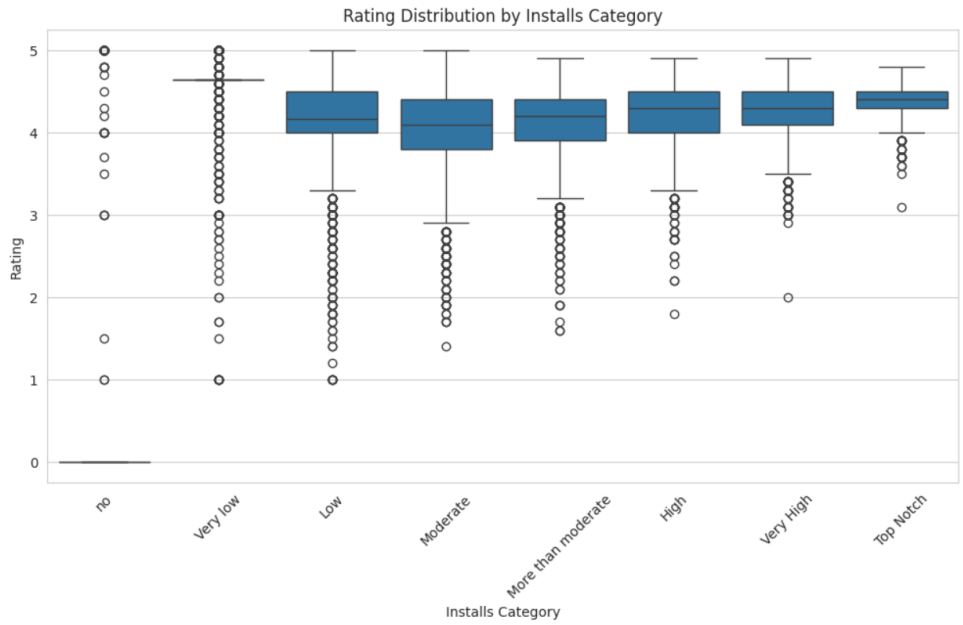

Figure 10: *correlation matrix*

CONCLUSION

This project involved a comprehensive exploratory data analysis of the Google Play Store dataset to uncover meaningful trends and patterns in app characteristics, user feedback, and market behavior.

Key findings include:

- The dataset initially contained 10,841 apps, reduced to 10,346 after cleaning.
- FAMILY and GAME categories dominate the Play Store in terms of number of apps.
- Apps in the EVENTS, EDUCATION, and BOOKS_AND_REFERENCE categories consistently achieved the highest average ratings.
- The GAME and COMMUNICATION categories recorded the highest number of installs, highlighting their popularity among users.
- A strong positive correlation ($r \approx 0.62$) was observed between the number of reviews and installs, indicating that app popularity drives user engagement.
- Rating distributions showed that most apps cluster between 4.0 and 4.5, with very few rated below 3.0.
- Missing ratings were more prevalent in apps with fewer installs, supporting the assumption that less popular apps receive less feedback.

These insights can assist developers, marketers, and stakeholders in understanding the dynamics of app success on the Google Play Store, helping to optimize product development and user acquisition strategies.

Next Steps

Based on the insights derived from this exploratory analysis, several avenues for further investigation and enhancement are recommended:

1. **Advanced Feature Engineering**
    - Derive new features such as:
        - App age (based on Last Updated)
        - Sentiment analysis on app descriptions or reviews
        - Popularity index combining installs and reviews
2. **Predictive Modeling**
    - Build regression models to predict app ratings or review counts
    - Use classification models to predict app success categories (e.g., high install or high rating)
3. **Cluster Analysis**
    - Segment apps using clustering algorithms (e.g., KMeans) based on install count, reviews, and ratings
    - Identify groups of similar apps or market niches
4. **Interactive Dashboards**
    - Develop dashboards using Plotly, Dash, or Tableau for dynamic exploration of app performance and trends
5. **Category-Specific Studies**
    - Deep-dive into specific categories like **GAME**, **FAMILY**, or **TOOLS** to understand user expectations and monetization strategies
6. **Textual Data Analysis**
    - Apply NLP techniques to analyze app names, descriptions, or user reviews to extract keywords, sentiment, or thematic trends
7. **Comparative App Store Analysis**
    - Compare Google Play Store trends with those from the Apple App Store or other platforms to uncover cross-platform differences

References

1. Google Play Store Apps Dataset
   Kaggle. (n.d.). *Google Play Store Apps*. Retrieved from https://www.kaggle.com/datasets/lava18/google-play-store-apps
2. Cleaned Google Play Store Dataset
   Kaggle. (n.d.). *Cleaned Google Play Store Dataset*. Retrieved from https://www.kaggle.com/datasets/harshvir04/cleaned-google-play-store-dataset
3. Google Play Store Reviews Dataset
   Kaggle. (n.d.). *Google Play Store Reviews*. Retrieved from https://www.kaggle.com/datasets/prakharrathi25/google-play-store-reviews
4. Analyzing Google Play Store Datasets with Python
   Medium. (2023, September 17). *Analyzing Google Play Store Datasets with Python*. Retrieved from https://medium.com/@kabila2022/analyzing-google-play-store-datasets-with-python-fb41e07a2518

5. Data Cleaning Case Study: Google Play Store Dataset
Tung M. Phung. (n.d.). *Data Cleaning Case Study: Google Play Store Dataset*. Retrieved from https://tungmphung.com/data-cleaning-case-study-google-play-store-dataset/