Internship Project Report

Unified Mentor Educational Institution, Gurgaon

Data Analyst Internship Program

Name: Vulkundakar Shivani

Comprehensive Data Analytics Internship Report

(Google Play Store Apps, Instagram Fake Detection,
E-commerce Furniture, Stock Market Finance)

📍 Submitted To:
Unified Mentor – Internship Program Coordinator

📍 Submitted By:
Vulkundakar Shivani
Data Analyst Intern

📅 Date: September 2025

INTRODUCTION

During my internship at **Unified Mentor** as a **Data Analyst Intern**, I had the opportunity to work on **four diverse projects** that covered domains like mobile applications, social media, e-commerce, and financial markets. The internship was designed to provide hands-on exposure to real-world datasets and industry-level analytical tasks, enabling me to apply my theoretical knowledge in a practical environment.

The main focus of the internship was to gain expertise in **data analysis, data preprocessing, exploratory data analysis (EDA), SQL queries, machine learning models, and business reporting.** Each project was structured in such a way that it allowed me to explore a complete workflow: starting from **data collection and cleaning, performing exploratory and statistical analysis, applying machine learning techniques for predictions, and finally deriving business insights and recommendations.**

The four projects I selected provided a **broad learning scope**:

1. **Google Play Store Apps Rating Prediction** – where the task was to analyse app-related features and predict their ratings.
2. **Instagram Fake vs Genuine Accounts Detection** – where the challenge was to classify user accounts as real or fake based on patterns in engagement and profile details.
3. **E-commerce Furniture Dataset 2024 Analysis** – where I explored pricing, discount strategies, and their impact on sales to predict the number of items sold.
4. **Stock Market Finance Analyst Project** – where I analysed historical stock price data for companies like Apple, Microsoft, Netflix, and Google to identify patterns, trends, and correlations in the market.

PROBLEM STATEMENTS

Project 1: Google Play Store Apps Rating Prediction

The Google Play Store hosts millions of applications in different categories, and user ratings significantly influence an app's visibility, download rate, and overall success. However, these ratings depend on multiple factors such as reviews, installs, app type (free or paid), price, and category. Developers often lack clarity on which of these features most strongly impact user satisfaction. The problem is to analyse a large dataset of Play Store applications, identify hidden patterns, and build a machine learning model that can accurately predict app ratings based on app characteristics. Solving this will help developers and businesses understand consumer preferences better and optimize their strategies to achieve higher ratings and greater user engagement.

Project 2: Instagram Fake vs Genuine Accounts Detection

Instagram is one of the largest social media platforms, yet it faces the persistent problem of fake and spam accounts. These accounts mislead users, spread misinformation, and reduce the authenticity of social engagement. Businesses that invest in influencer marketing also suffer when fake accounts distort follower counts and engagement metrics. Differentiating between genuine and fake accounts is challenging due to overlapping features such as follower count, following count, profile picture, bio details, and post activity. The problem here is to develop a classification model that can reliably detect whether an Instagram account is genuine or fake, based on its attributes. This project aims to provide insights into digital safety, enhance brand trust, and support social media platforms in reducing fraudulent activity.

Project 3: E-commerce Furniture Dataset – Sales Prediction & Trend Analysis

The e-commerce industry is rapidly growing, and furniture sales through online platforms are becoming increasingly common. Customers' purchasing behaviour is influenced by factors such as product price, discounts, tags like "free shipping," and product descriptions. Sellers need data-driven strategies to forecast sales and design better promotions. The problem is to analyze an e-commerce furniture dataset consisting of product details, prices, discounts, and tags, and predict the number of items sold. Through exploratory data analysis, feature engineering, and predictive modelling (Linear Regression, Random Forest, etc.), this project aims to uncover consumer trends, understand the impact of discounts and pricing on sales, and help businesses optimize their inventory and marketing strategies.

Project 4: Stock Market Analysis & Prediction

The stock market is highly dynamic and influenced by various financial, economic, and social factors. Investors, traders, and businesses need accurate predictions to make informed decisions and minimize risks. However, stock market data is vast, complex, and volatile, making prediction a challenging task. The problem is to analyze historical stock market data, perform exploratory data analysis, and apply machine learning models to forecast stock price movements. By using techniques such as moving averages, regression models, and time series analysis, this project seeks to identify trends, patterns, and potential signals that can assist investors in making smarter decisions. The outcome will not only showcase the potential of data science in finance but also highlight the importance of data-driven approaches in high-risk domains.

TECHNIQUES USED

Tools & Platforms

To complete the internship projects effectively, a variety of tools and platforms were employed. **Python** served as the primary programming language due to its versatility and extensive ecosystem for data analysis and machine learning. **SQL** was used for querying structured datasets, performing aggregations, and extracting insights from relational databases. In addition, visualization tools such as **Power BI** and **Tableau** were considered for presenting the results in an interactive and business-friendly manner. **Jupyter Notebook** and **VS Code** were the main development environments, chosen for their support of interactive coding and visualization.

Python Libraries

A number of Python libraries were utilized across the four projects:

- **Pandas**: For data manipulation, cleaning, and analysis. It provided powerful data structures such as Data Frames for handling tabular data.
- **NumPy**: For numerical operations, mathematical calculations, and array-based computations.
- **Matplotlib & Seaborn**: For creating static and interactive visualizations including histograms, bar charts, scatter plots, and heatmaps, which helped identify trends and correlations.
- **Scikit-learn**: For implementing machine learning models such as Linear Regression, Random Forest, Logistic Regression, and classification models. It also provided functions for preprocessing, model evaluation, and train-test splitting.

- **TF-IDF & Label Encoding (from sklearn)**: For handling text-based features like Instagram bios or product titles in the furniture dataset.
- **Other supporting libraries**: Such as OS (for file handling), warnings (to ignore unnecessary alerts), and occasionally NLTK/Regex for text processing.

Approach & Methodologies

Across all four projects, a structured methodology was followed:

1. **Data Cleaning & Preprocessing**
    - Removal of missing values, duplicates, and irrelevant records.
    - Conversion of categorical variables into numeric representations (Label Encoding, One-Hot Encoding).
    - Standardization of formats, e.g., converting prices from strings to floats, removing commas/symbols from installs, and normalizing text features.

2. **Exploratory Data Analysis (EDA)**
    - Distribution analysis (e.g., histogram of ratings, sales distribution, or stock price trends).
    - Correlation analysis to identify relationships among variables.
    - Visualizations to understand customer behaviour, detect anomalies, and highlight dominant categories or features.

3. **SQL Queries & Analysis**
    - Extracting aggregated insights such as the most popular app categories, average ratings, most sold furniture items, or identifying patterns in social media datasets.
    - Performing group-by operations to analyze trends across categories.
    - Using filters and conditions to separate free vs paid apps, or genuine vs fake accounts.

4. Machine Learning Models

- **Regression Models**: Linear Regression and Random Forest Regressor were applied in projects like Play Store ratings and furniture sales prediction.
- **Classification Models**: Logistic Regression, Decision Trees, and Random Forest Classifiers were used for fake vs genuine Instagram accounts.
- **Time Series & Trend Analysis**: Applied in Stock Market Analysis to study moving averages and forecast price movements.
- Model performance was measured using metrics such as **R^2 Score, Mean Squared Error (MSE), Accuracy, Precision, and Recall** depending on the problem type.

PROJECT 1: GOOGLE PLAY STORE APPS RATING PREDICTION

Dataset Description

The dataset for this project was obtained from the Google Play Store and consists of over **10,000 applications** spanning multiple categories. It contains **13 attributes** that describe various characteristics of each app, such as app name, category, user rating, reviews, size, installs, price, type (free/paid), genres, and content rating. The **target variable** for this project is the *Rating* column, which represents user feedback on a scale of 1 to 5.

Some key features include:

- **Category**: Defines the segment under which an app is listed (e.g., Family, Game, Tools, Business).
- **Reviews**: Total number of user reviews, representing engagement and popularity.
- **Installs**: The total number of times an app has been downloaded. This was stored as text (e.g., "1,000+") and converted to numeric form for analysis.
- **Type & Price**: Differentiates between free and paid apps. Paid apps required price conversions from string values like "$4.99" to numeric floats.
- **Content Rating**: Indicates the suitable audience (e.g., Everyone, Teen, Mature 17+).

Data preprocessing steps included **removing missing values in Ratings**, **handling duplicates**, and **transforming categorical data into numeric codes** using encoding. Columns such as *Installs* and *Price* required cleaning (removing symbols like "+", "$", and commas).

This dataset was particularly useful as it provided a **balanced mix of user-centric metrics (ratings, installs, reviews)** and **developer-centric features (category, type, content rating, price)**, allowing both business insights and predictive modeling.

SQL Analysis

SQL queries were performed on the dataset to extract meaningful insights:

- **Category Counts**: By grouping apps by category, it was found that *Family*, *Games*, and *Tools* had the highest number of applications.

```sql
SELECT
    CATEGORY,
    COUNT(APP) AS TOTAL_APPS
FROM
    GOOGLE_PLAY_STORE
GROUP BY
    CATEGORY
ORDER BY
    TOTAL_APPS DESC;
```

- **Average Ratings by Category**: By computing the mean rating for each category, categories like *Books & Reference* and *Education* emerged with higher average ratings, while some utility-based apps had slightly lower average

```
SELECT
    CATEGORY,
    AVG(RATING) AS AVG_RATING
FROM
    GOOGLE_PLAY_STORE
GROUP BY
    CATEGORY
ORDER BY
    AVG_RATING DESC
```

- **Free vs Paid Distribution**: Around **92% of apps were free**, and only **8% were paid**, showing a clear dominance of free applications in the Play Store

```
SELECT
    TYPE,
    COUNT (APP)AS NUM_APPS
FROM
    GOOGLE_PLAY_STORE
GROUP BY
     TYPE;
```

EDA Insights

Using **Python visualizations (Matplotlib & Seaborn)**, several key insights were discovered:

1. **Distribution of Ratings**: The majority of apps had ratings between **4.0 and 4.5**, showing that most users give favourable reviews, though very few apps achieved a perfect 5.0 rating.
2. **Top Categories**: The *Family* and *Games* categories dominated the Play Store, accounting for the largest proportion of apps. *Tools* and *Business* followed next.
3. **Free vs Paid Apps**: Free apps had significantly higher downloads but generally lower average ratings compared to some paid apps. Paid apps were fewer but often attracted more serious users.
4. **Reviews Correlation**: Apps with higher reviews also tended to have better ratings, showing that user engagement often reflects user satisfaction.

These insights helped to highlight **which categories dominate the market, how ratings are distributed, and how app type (free/paid) influences installs and ratings**.

Machine Learning Model (Random Forest)

To predict app ratings, a **Random Forest Regressor** was applied. The independent features included **Category, Reviews, Size, Installs, Type, Price, Genres, and Content Rating**.

Steps:

1. Encoded categorical features using Label Encoding.
2. Split the dataset into **train (70%) and test (30%)** sets.
3. Trained the Random Forest model with 100 estimators.

Python Code Snippet:

```
From sklearn _ensemble import
    RANDOMFORESTREGRESSOR
    MODEL = RANDOMFORESTREGRESSOR
    (N_ESTIMATORS = 100, RANDOM_STATE = 42)
        MODEL.FIT
    (X_TRAIN, Y_TRAIN)
        Y_PRED = MODEL.PREDICT
     X TEST)
        PRINT
    ("R2 Score:", R2_SCORE (Y_TEST,
    Y_PRED))
        PRINT
    ("MSE:", MEAN_SQUARED_ERROR (Y_TEST,
    Y PRED))
```

Results:

- **R^2 Score ≈ 0.70** → The model explained around 70% of the variance in app ratings.
- **MSE ≈ 0.10** → The error in predictions was relatively low.

This shows that Random Forest was effective in predicting app ratings and identifying the most influential factors. Features like *Reviews*, *Installs*, and *Category* were the strongest predictors of app ratings.

PROJECT 2: INSTAGRAM FAKE VS GENUINE ACCOUNTS DETECTION

Dataset Description

The dataset for this project consists of thousands of Instagram user profiles, with each entry describing key features that can help differentiate between fake and genuine accounts. The target variable is the **Account Type**, labelled as either *Fake* or *Genuine*. This dataset is highly relevant in today's digital environment, where fake accounts and spam bots are a major concern for social media platforms, influencers, and businesses.

Some key features include:

- **Followers**: The number of followers a profile has. Fake accounts often show unrealistic follower counts.
- **Following**: Total number of accounts the user follows. Fake accounts tend to follow excessively.
- **Posts**: The number of posts shared. Genuine accounts typically show a balanced post history, whereas fake accounts often lack consistent posting.
- **Engagement Metrics**: Ratio of likes/comments to followers, showing how interactive an account is.
- **Bio Length**: Length of the account's bio text. Many fake accounts use blank or very short bios.

Data preprocessing included removing duplicate rows, filling missing values, and standardizing engagement metrics. The target variable *Account_type* was encoded as binary (1 = Genuine, 0 = Fake).

SQL Analysis

SQL queries were performed to summarize account types and engagement trends:

Count of Fake vs Genuine Accounts:

```
SELECT
Account_ type, COUNT (*) AS total _accounts
FROM
Instagram accounts
GROUP BY
Account _ type;
```

Insight: **Around 60–70% of accounts were genuine, while 30–40% were flagged as fake.**

Average Followers by Account Type:

```
SELECT
Account _ type, AVG(Followers) AS avg_ followers
FROM
Instagram_ accounts
GROUP BY
Account _ type;
```

Insight: **Genuine accounts had significantly higher average followers compared to fake ones.**

Posts vs Account Type:

```
SELECT
Account _type, AVG(Posts) AS avg _posts
FROM
instagram _accounts
GROUP BY
 Account_type;
```

Insight: **Fake accounts had very few posts, while genuine accounts had more consistent posting activity.**

EDA Insights

Python visualizations (Seaborn, Matplotlib) gave the following insights:

1. **Followers Distribution** – Most fake accounts clustered at very low follower counts (<200). Genuine accounts had a wider range.
2. **Posts Activity** – Fake accounts rarely exceeded 10–15 posts, whereas genuine ones were more active.
3. **Engagement Rates** – Genuine accounts had much stronger engagement ratios (likes/comments per follower).
4. **Following Behaviour** – Fake accounts often followed thousands of users in hopes of gaining follow-backs.

These trends made it clear that *activity and engagement features* strongly separated fake from genuine accounts.

Machine Learning Model (Random Forest Classifier)

To predict whether an account is fake or genuine, a **Random Forest Classifier** was applied.

Steps:

1. Encoded the target (*Account_type*: Fake = 0, Genuine = 1).
2. Split data into training (70%) and testing (30%).
3. Trained Random Forest with 100 trees.
4. Evaluated accuracy and confusion matrix.

Python Code Snippet:

```
from sklearn. ensemble import RandomForestClassifier
from sklearn. metrics import classification, report, confusion_matrix

model = RandomForestClassifier (n_estimators=100, random_state=42)
model.fit (X_train, y_train)
y_pred = model. Predict (X_test)

print (classification_report (y_test, y_pred))
print (confusion_matrix (y_test, y_pred))
```

Results:

- **Accuracy:** ≈ 92% → The model correctly classified most accounts.
- **Precision & Recall:** Both strong for detecting fake accounts, reducing false positives.
- **Key Predictors:** Followers, Posts, and Engagement metrics were the strongest predictors.

Conclusion: The model successfully identified fake accounts with high accuracy, providing a strong detection framework for Instagram security and brand management.

PROJECT 3: E-COMMERCE FURNITURE DATASET 2024

Dataset Description

This project focuses on analysing an **E-commerce Furniture Dataset** containing sales information for around **2,000 furniture products** listed online. The dataset captures a variety of product-level details such as pricing, discounts, shipping tags, and the number of items sold. The primary goal is to understand the factors that influence sales and to predict the number of items sold based on product attributes.

Some key features include:

- **Product Title**: The name/description of the furniture item.
- **Price**: The selling price of each item, stored as strings like "₹3,499" which were cleaned into numeric format.
- **Sold**: The number of items sold (target variable).
- **Tag Text**: Labels like "Free Shipping" or "Discount Offer" that highlight promotional tags.

Data preprocessing included cleaning the **Price** column (removing ₹, commas), handling missing values, and ensuring numeric formats. Duplicates were removed, and categorical fields (like tag Text) were encoded for machine learning.

SQL Analysis

To uncover insights, SQL queries were run on the dataset.

Total Items Sold:

```
SELECT SUM(Sold) AS total _sales
FROM ecommerce _furniture;
```

Insight: **Gave the total volume of items sold across all products.**

Average Price of Products:

```
SELECT AVG(Price) AS avg_ price
FROM ecommerce_ furniture;
```

Insight: **The average furniture price fell in the mid-range segment, indicating affordability was a key driver.**

Sales by Shipping Tag:

```
SELECT tagText, SUM(Sold) AS total_sold
FROM ecommerce_furniture
GROUP BY tagText
ORDER BY total_sold DESC;
```

Insight: **Items with "Free Shipping" and "Discount Offer" tags sold significantly more than those without.**

EDA Insights

Visualizations (Seaborn, Matplotlib) provided valuable findings:

1. **Distribution of Sales** – Most products sold fewer than 50 units, while a few popular products sold in very high quantities.
2. **Price vs Sales** – Products priced moderately showed the highest sales, while very expensive items sold less.
3. **Impact of Promotions** – Products tagged with "Free Shipping" or discounts consistently outperformed others.
4. **Outliers** – A few premium furniture pieces had extremely high prices but very low sales, dragging down averages.

Machine Learning Model (Linear Regression & Random Forest)

The goal was to predict the number of furniture items sold.

Steps:

1. Cleaned and encoded features like tagText.
2. Selected features: Price, tagText (promotion type), etc.
3. Split dataset into training (70%) and testing (30%).
4. Applied **Linear Regression** and **Random Forest Regressor** for comparison.

Python Code Snippet:

```
from sklearn. linear_ model import Linear Regression
from sklearn. ensemble import RandomForestRegressor
from sklearn. metrics import r2_score, mean _squared _error

lr = Linear Regression ()
rf = RandomForestRegressor (n_estimators=100, random_state=42)

lr. Fit (X_train, y_train)
rf.fit (X_train, y_train)

y_pred_lr = lr. predict (X_test)
y_pred_rf = rf. Predict (X_test)

print ("Linear Regression R²:", r2_score (y_test, y_pred_lr))
print ("Random Forest R²:", r2_score (y_test, y_pred_rf))
```

Results:

- **Linear Regression $R^2 \approx 0.55$** → Model explained ~55% variance.
- **Random Forest $R^2 \approx 0.75$** → Model performed significantly better.
- **Key Predictors:** Price and tagText (shipping/discounts) had the strongest influence on sales.

Conclusion: Random Forest gave a more reliable prediction model for sales forecasting, highlighting the importance of pricing and promotional tags in driving online furniture sales.

PROJECT 4: STOCK MARKET FINANCE ANALYST

Dataset Description

The **Stock Market Finance Analyst Project** was based on daily trading data of major technology companies like **Apple, Microsoft, Netflix, and Google**. The dataset contained historical stock prices and volumes across multiple years. The objective was to analyze market performance, identify patterns, and predict stock closing prices using machine learning.

Some key features include:

- **Date**: The trading day.
- **Ticker**: The company stock symbol (e.g., AAPL for Apple).
- **Open, High, Low, Close**: Prices at the beginning, highest, lowest, and end of the day.
- **Volume**: Total number of shares traded on a given day.

Data preprocessing involved converting date columns to proper datetime format, sorting by time, handling missing trading days, and normalizing values for modelling.

SQL Analysis

SQL queries were used to extract market insights.

Average Closing Price by Ticker:

```
SELECT Ticker, AVG(Close) AS avg_ close
FROM stock_ market
GROUP BY Ticker;
```

Insight: **Microsoft and Apple had consistently high average closing prices, while Netflix showed more volatility.**

Highest Traded Volume by Company:

```
SELECT Ticker, MAX(Volume) AS max_ volume
FROM stock_ market
GROUP BY Ticker;
```

Insight: **Netflix had occasional spikes in volume, often linked to major announcements.**

Daily Average Closing Price:

```
SELECT Date, AVG(Close) AS daily_ avg_ close
FROM stock_ market
GROUP BY Date
ORDER BY Date;
```

Insight: **Helped identify market-wide trends on specific dates.**

EDA Insights

Visualization helped uncover patterns:

1. **Closing Price Trends** – Apple and Microsoft showed steady upward movement, while Netflix was more volatile.
2. **Moving Averages** – 30-day moving averages highlighted long-term growth trends.
3. **Correlation Heatmap** – Strong correlation between Open, High, Low, and Close prices. Volume had weaker correlation.
4. **Volatility Analysis** – Netflix showed the highest price swings compared to stable players like Microsoft.

Machine Learning Model (Linear Regression for Price Prediction)

To predict the **Close Price**, a **Linear Regression model** was applied using features (Open, High, Low, Volume).

Steps:

1. Prepared features X = [Open, High, Low, Volume] and target y = Close.
2. Split dataset into training (70%) and testing (30%).
3. Trained Linear Regression model and evaluated predictions.

Python Code Snippet:

```
from sklearn. linear_ model import Linear Regression
from sklearn. metrics import r2_score
model = Linear Regression ()
model. Fit (X_train, y_train)
y_pred = model. predict (X_test)
print ("Stock Price Prediction R²:", r2_score (y_test, y_pred))
```

Results:

- **R^2 Score ≈ 0.85** → Model explained ~85% of price movements.
- **Key Predictors:** High and Low prices strongly influenced Close predictions.
- **Limitations:** Linear Regression captured trends but not extreme volatility.

Conclusion: The project successfully modelled stock prices using regression, providing reliable short-term predictions and insights into company performance trends.

OVER ALL CONCLUSION

Through this internship with **Unified Mentor**, I gained hands-on experience across four diverse domains: mobile applications, social media, e-commerce, and financial markets. Each project followed a complete analytical cycle, beginning with **data cleaning and preprocessing**, moving through **exploratory data analysis (EDA)** and **SQL-based querying**, and culminating in **machine learning model building and interpretation of results**.

The projects provided both **technical learning** and **business understanding**:

- From the **Google Play Store dataset**, I learned how app features influence user ratings and downloads.
- The **Instagram fake accounts project** showed the importance of classification models in improving platform trust.
- The **E-commerce furniture dataset** highlighted how pricing and promotions impact online sales.
- The **Stock Market project** developed my skills in financial forecasting and time-series interpretation.

Overall, this internship taught me how to **translate raw data into meaningful insights** and strengthened my ability to approach real-world business problems with analytical thinking. I not only improved my technical skills in **Python, SQL, and Machine Learning**, but also developed a strong foundation in drawing **practical business insights** that can support decision-making across industries.

FUTURE SCOPE (DETAILED – FOR REPORT)

While the internship projects provided valuable insights and predictive modeling using SQL and machine learning, there are several opportunities to extend this work further:

1. **Deep Learning Approaches**
    - Apply **LSTM (Long Short-Term Memory)** models for more accurate stock market forecasting.
    - Use **NLP (Natural Language Processing)** techniques to analyze Instagram bios, hashtags, and reviews from the Play Store to strengthen fake account detection and app rating prediction.
2. **Model Deployment**
    - Build interactive applications using **Flask** or **Stream lit** so that predictions (e.g., app ratings, fake account classification, stock prices) can be accessed by end users in real time.
3. **Interactive Dashboards**
    - Develop **Tableau or Power BI dashboards** for decision-makers to monitor key metrics such as sales trends, user ratings, or stock price movements dynamically.
    - Integrate live APIs (e.g., financial market APIs) for real-time data visualization.
4. **Scalability and Automation**
    - Automate the ETL (Extract, Transform, Load) pipeline to continuously clean, analyze, and update datasets.
    - Deploy models in the cloud (AWS/GCP/Azure) for scalability.

This future scope ensures that the learnings from the four projects can evolve into **real-world deployable solutions** that combine analytics, AI, and business intelligence.