Amazon Bestsellers Data Analysis Report

Ayush B Raj

SUMMARY

This report presents an analysis of the **Amazon Top 50 Bestselling Books (2009–2019)** dataset, aiming to uncover trends and insights related to book popularity and inventory behaviour. Beyond exploring the dataset's structure and key attributes like **user ratings**, **reviews**, **price**, and **genre**, we introduced a simulated **inventory** column to approximate real-world stock conditions. Using this, we calculated **stock turnover** as a proxy for sales efficiency and categorized books into **stock health** and **inventory risk** classes to highlight potential overstocking or stockout scenarios.

INTRODUCTION

The Amazon Bestsellers dataset offers insights into top-selling books on Amazon, including attributes such as book name, author, user rating, number of reviews, price, year of publication, and genre (Fiction or Non-Fiction).

This analysis aims to:

➢ Understand the dataset's structure and content.

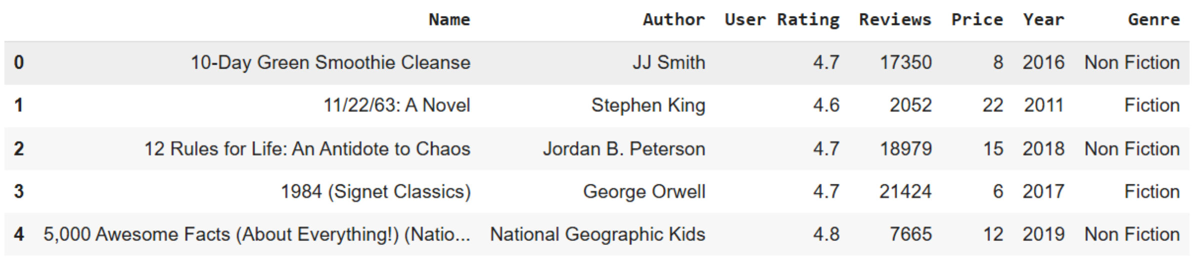
➢ Summarize key statistical characteristics.

➢ Visualize patterns and relationships in the data.

➢ Simulate inventory behaviour to assess stock status and associated risks.

➢ Develop and evaluate a predictive model to estimate the number of reviews based on book attributes.

This report combines data preprocessing, visualization, and machine learning techniques to extract insights and demonstrate the potential application of data science in the publishing and e-commerce industry. The findings are supported with Python code, visual outputs, and clear interpretations to ensure clarity and relevance.



	Name	Author	User Rating	Reviews	Price	Year	Genre
0	10-Day Green Smoothie Cleanse	JJ Smith	4.7	17350	8	2016	Non Fiction
1	11/22/63: A Novel	Stephen King	4.6	2052	22	2011	Fiction
2	12 Rules for Life: An Antidote to Chaos	Jordan B. Peterson	4.7	18979	15	2018	Non Fiction
3	1984 (Signet Classics)	George Orwell	4.7	21424	6	2017	Fiction
4	5,000 Awesome Facts (About Everything!) (Natio...	National Geographic Kids	4.8	7665	12	2019	Non Fiction

DATASET OVERVIEW

The dataset, stored in a CSV file (bestsellers with categories.csv), was loaded into a Pandas DataFrame for analysis. The dataset contains 550 entries with the following columns:

• Name: Title of the book.

• Author: Author of the book.

• User Rating: Average user rating (out of 5).

• Reviews: Number of reviews received.

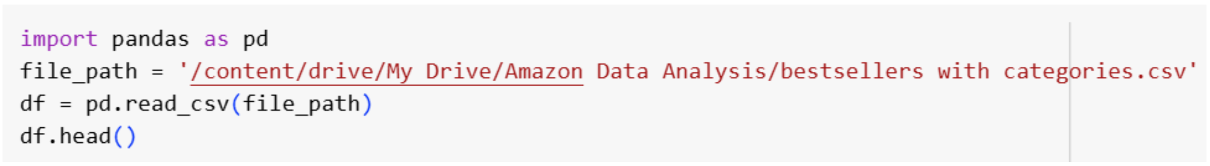
• Price: Price of the book in USD.

• Year: Year of publication or bestseller ranking.

• Genre: Fiction or Non-Fiction.

DATASET EXPLORATION

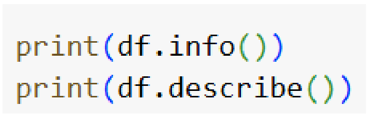
The initial exploration involved loading the dataset and examining its structure using Pandas. The following code snippet demonstrates the loading process and initial inspection:



```
import pandas as pd
file_path = '/content/drive/My Drive/Amazon Data Analysis/bestsellers with categories.csv'
df = pd.read_csv(file_path)
df.head()
```

This code outputs the first five rows of the dataset, revealing books such as "10-Day Green Smoothie Cleanse" by JJ Smith and "1984 (Signet Classics)" by George Orwell, with their respective ratings, reviews, prices, years, and genres.

To understand the dataset's structure and summary statistics, the following code was executed:



```
print(df.info())
print(df.describe())
```

Findings from df.info():

▪ The dataset contains 550 entries with no missing values across all columns.

▪ Data types: Name, Author, and Genre are objects (strings); User Rating is a float; and Reviews, Price, and Year are integers.

▪ Memory usage: Approximately 30.2 KB.

Findings from df.describe():

▪ User Rating: Ranges from 3.3 to 4.9, with a mean of 4.62 and a standard deviation of 0.23, indicating generally high ratings with low variability.

▪ Reviews: Ranges from 37 to 87,841, with a mean of 11,953 and a high standard deviation of 11,731, suggesting significant variability in review counts.

▪ Price: Ranges from $0 to $105, with a mean of $13.10 and a standard deviation of $10.84, indicating a wide range of book prices.

▪ Year: Spans from 2009 to 2019, with a mean of 2014, reflecting a decade of bestseller data.

DATA CLEANING

The dataset has no missing values and also no duplication in traditional sense, Standardized column names by converting them into lower case and replacing ‘ ‘ with ‘_’ . changed the type of price from int64 to float64

```
print("\nMissing values per column:\n", df.isnull().sum())
```

```
Missing values per column:
 Name           0
Author         0
User Rating    0
Reviews        0
Price          0
Year           0
Genre          0
dtype: int64
```

```
df = df.drop_duplicates()
print("After dropping duplicates:", df.shape)
```

After dropping duplicates: (550, 7)

```
df.columns = df.columns.str.lower().str.strip().str.replace(' ', '_')
print("Updated column names:", df.columns.tolist())
```

Updated column names: ['name', 'author', 'user_rating', 'reviews', 'price', 'year', 'genre']

```
df['price'] = df['price'].replace('[$,]', '', regex=True).astype(float)
```

Converted price datatype from integer to float

```
df.to_csv("/content/drive/My Drive/Amazon Data Analysis/cleaned_amazon_data.csv", index=False)
```

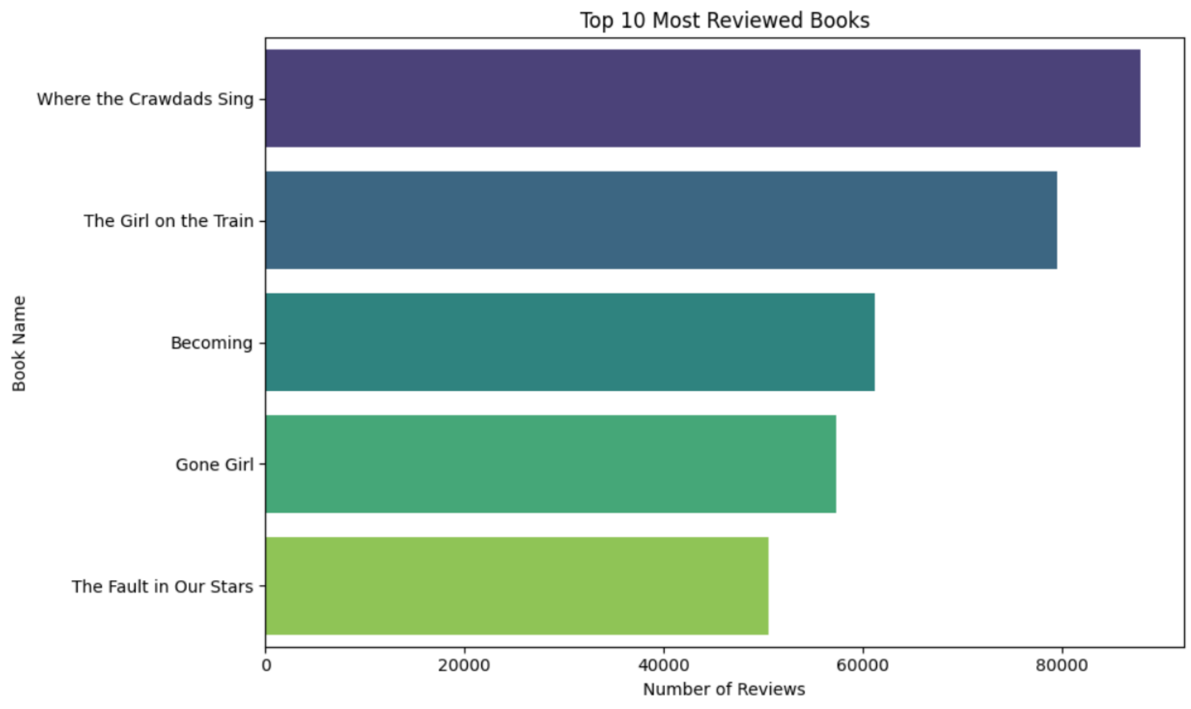
Created a cleaned version of the dataset cleaned_amazon_data.csv

DATA ANALYSIS AND VISUALIZATIONS

To extract meaningful insights, the dataset underwent thorough exploratory data analysis (EDA). Key trends and patterns were explored using standard visualizations.

Bar Plot

```
top_books = df.sort_values(by='reviews', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(data=top_books, x='reviews', y='name', hue='name', palette='viridis', legend=False)
plt.title("Top 10 Most Reviewed Books")
plt.xlabel("Number of Reviews")
plt.ylabel("Book Name")
plt.tight_layout()
plt.show()
```

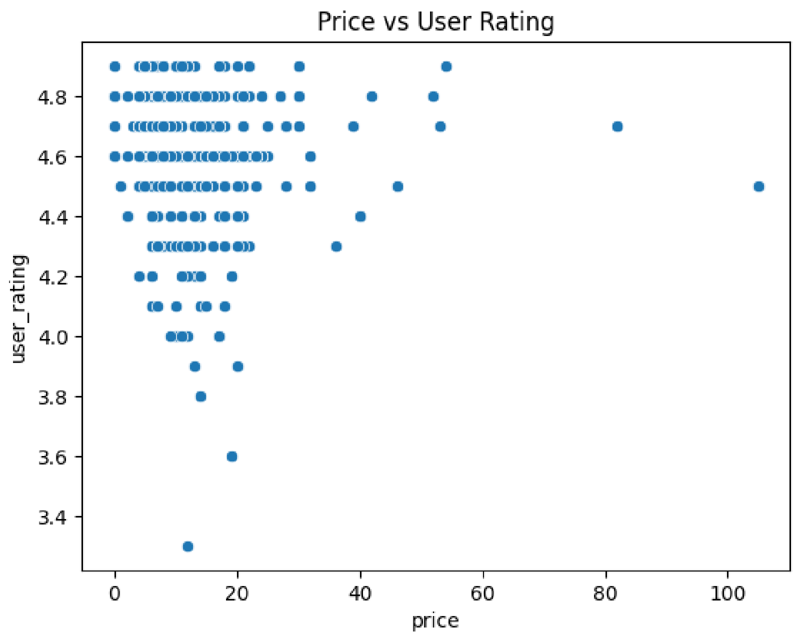


Interpretation: Found the top 5 most reviewed books from the dataset.

Where the Crawdads Sing is the most reviewed book ie 87841 reviews

Scatter Plot

```
sns.scatterplot(x='price', y='user_rating', data=df)
plt.title("Price vs User Rating")
plt.show()
```

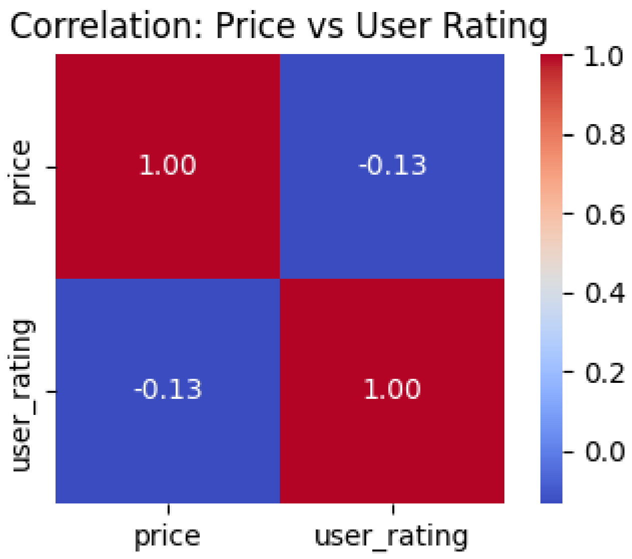


Most books, regardless of price, have user ratings **between 4.3 and 4.9**.

There's a heavy concentration of books with ratings close to **4.8–4.9**, indicating overall customer satisfaction.

Correlation Matrix

This heatmap displays the **correlation coefficient** between the price and user_rating of books.



user rating vs price

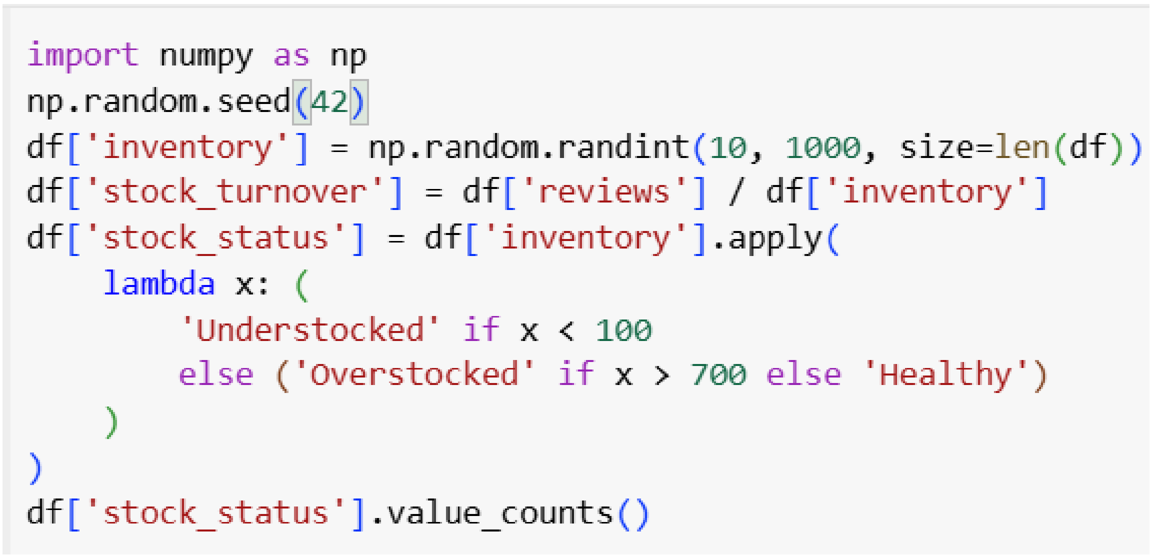
-0.13 slight negative correlation (higher priced books may have less rating)

This indicates a **very weak negative relationship**.

As price increases, user rating slightly tends to decrease — but the effect is **minimal** and not significant.

INVENTORY DATA ANALYSIS

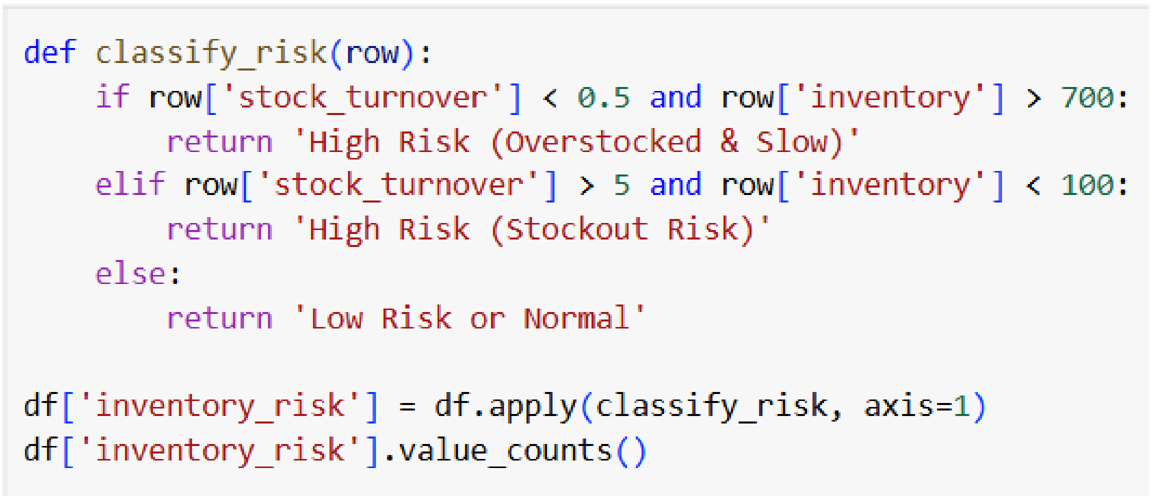
Introduced a simulated **inventory** column as a proxy to analyze stock health since actual inventory data wasn't available. Using this, we calculated a **stock_turnover** metric and classified items into **Understocked**, **Healthy**, or **Overstocked** based on inventory levels.



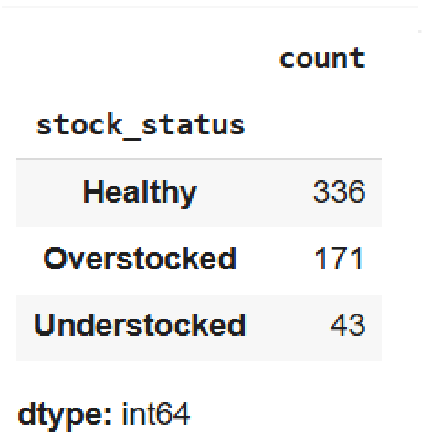
```python
import numpy as np
np.random.seed(42)
df['inventory'] = np.random.randint(10, 1000, size=len(df))
df['stock_turnover'] = df['reviews'] / df['inventory']
df['stock_status'] = df['inventory'].apply(
    lambda x: (
        'Understocked' if x < 100
        else ('Overstocked' if x > 700 else 'Healthy')
    )
)
df['stock_status'].value_counts()
```

To assess inventory risk, we created a new feature inventory_risk by evaluating the relationship between stock turnover and inventory levels.

Items with **high inventory but low turnover** are flagged as **"High Risk (Overstocked & Slow)"**, while those with **low inventory but high turnover** are marked **"High Risk (Stockout Risk)"**. All others are considered **"Low Risk or Normal"**, helping prioritize inventory management actions

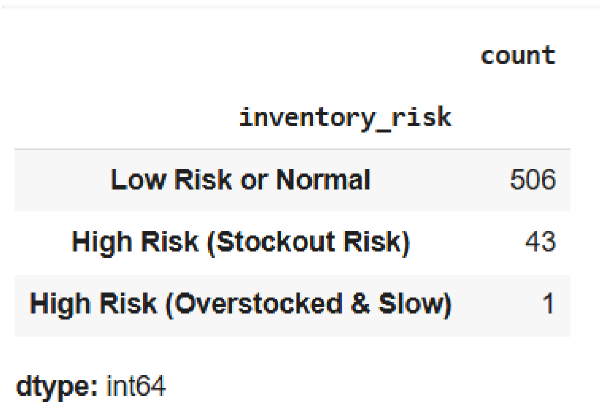


```python
def classify_risk(row):
    if row['stock_turnover'] < 0.5 and row['inventory'] > 700:
        return 'High Risk (Overstocked & Slow)'
    elif row['stock_turnover'] > 5 and row['inventory'] < 100:
        return 'High Risk (Stockout Risk)'
    else:
        return 'Low Risk or Normal'

df['inventory_risk'] = df.apply(classify_risk, axis=1)
df['inventory_risk'].value_counts()
```



stock_status	count
Healthy	336
Overstocked	171
Understocked	43

dtype: int64

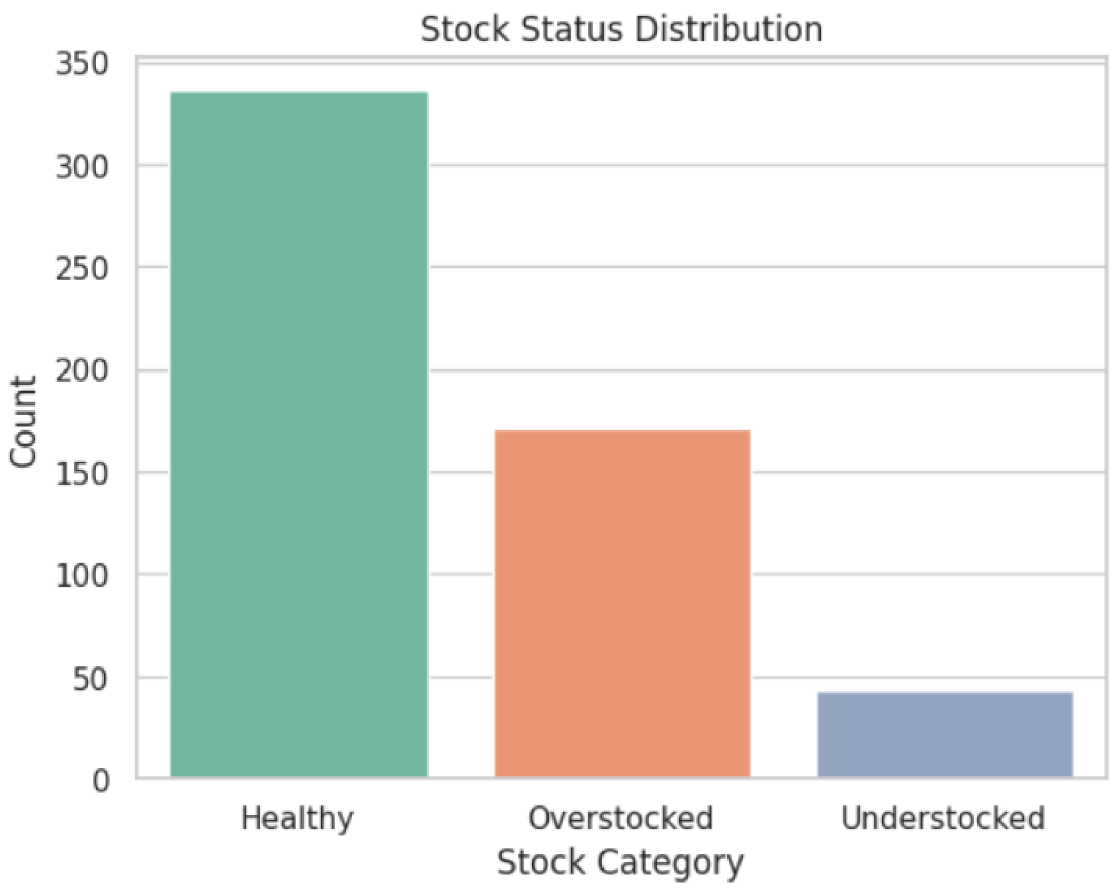


inventory_risk	count
Low Risk or Normal	506
High Risk (Stockout Risk)	43
High Risk (Overstocked & Slow)	1

dtype: int64

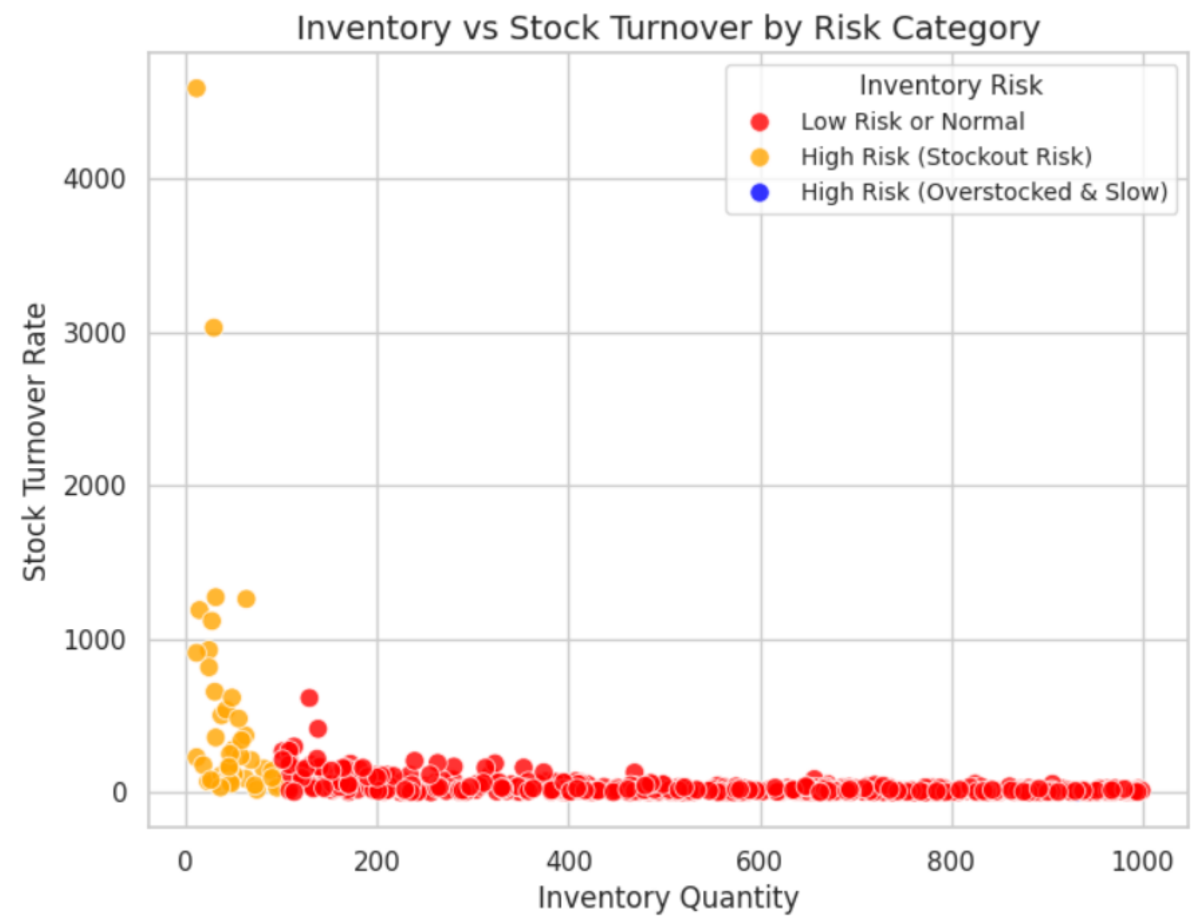
This count plot visualizes the distribution of books across different stock categories: Understocked, Healthy, and Overstocked. It helps quickly identify which stock status is most common in the dataset.

```
sns.countplot(data=df, x='stock_status', palette='Set2', hue='stock_status')
plt.title('Stock Status Distribution')
plt.xlabel('Stock Category')
plt.ylabel('Count')
plt.show()
```



This scatter plot shows the relationship between inventory levels and stock turnover, coloured by inventory risk categories. It visually highlights patterns such as overstocked items with low turnover and potential stockout risks, aiding in identifying critical inventory issues.

```
import seaborn as sns
import matplotlib.pyplot as plt
custom_palette = {
    'High Risk (Overstocked & Slow)': 'blue',
    'High Risk (Stockout Risk)': 'orange',
    'Low Risk or Normal': 'red'}
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df,x='inventory',y='stock_turnover',hue='inventory_risk',
                palette=custom_palette,s=70,alpha=0.8)
plt.title('Inventory vs Stock Turnover by Risk Category', fontsize=14)
plt.xlabel('Inventory Quantity', fontsize=12)
plt.ylabel('Stock Turnover Rate', fontsize=12)
plt.legend(title='Inventory Risk', fontsize=10, title_fontsize=11)
plt.show()
```



MODEL BUILDING

In this code, we're preparing data to predict **stock_turnover** using a linear regression model. Since **genre** is a categorical feature, it's converted into numerical form using **LabelEncoder** and stored as **genre_encoded**. The selected input features — **reviews**, **user_rating**, **price**, and **genre_encoded** — are used to train the model to estimate the **stock turnover rate**.

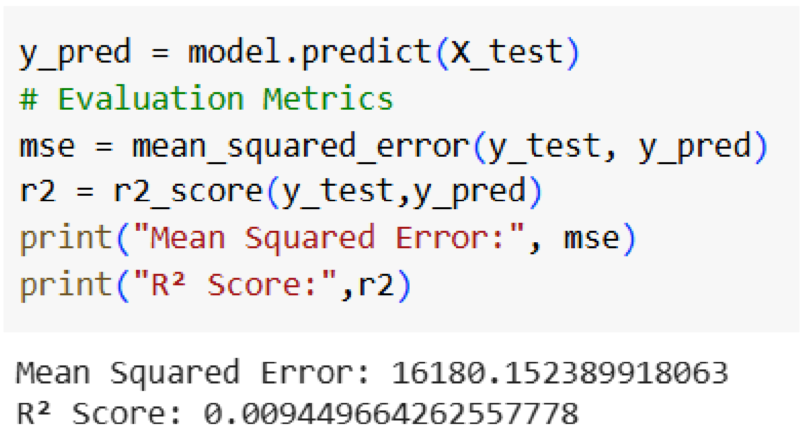
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder

# Encode Genre (categorical) into numbers
le = LabelEncoder()
df['genre_encoded'] = le.fit_transform(df['genre'])

# Feature columns
features=['reviews', 'user_rating', 'price', 'genre_encoded']

# Target column
target='stock_turnover'
```

The feature matrix X and target variable y are defined using the selected columns. The data is then split into **training (80%)** and **testing (20%)** sets to evaluate model performance. A **Linear Regression** model is created and trained on the training data using model.fit(), allowing it to learn the relationship between the input features and stock turnover.

```
X=df[features]
y=df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

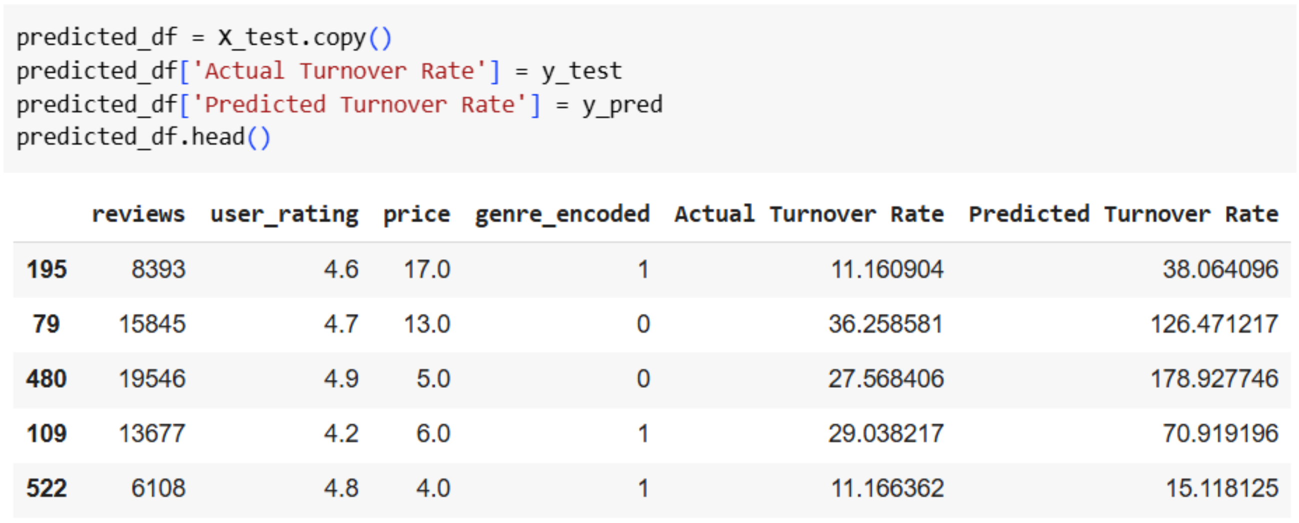
The model predicts **stock turnover** for the test data using model.predict(), and the results are compared to the actual values. **Mean Squared Error (MSE)** measures the average squared difference between predicted and actual values — lower is better. **R² Score** indicates how well the model explains the variance in the target variable — a value closer to 1 means better predictive performance.



```
y_pred = model.predict(X_test)
# Evaluation Metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test,y_pred)
print("Mean Squared Error:", mse)
print("R² Score:",r2)
```

```
Mean Squared Error: 16180.152389918063
R² Score: 0.009449664262557778
```

The **Mean Squared Error (MSE)** of **16180.15** indicates a high average error between the predicted and actual stock turnover values, meaning the model's predictions are not very close to the true values. The **R² Score of 0.009** suggests that the model explains less than **1%** of the variation in stock turnover, indicating that the linear regression model performs poorly and is not effective in capturing the underlying patterns in the data.



```
predicted_df = X_test.copy()
predicted_df['Actual Turnover Rate'] = y_test
predicted_df['Predicted Turnover Rate'] = y_pred
predicted_df.head()
```

	reviews	user_rating	price	genre_encoded	Actual Turnover Rate	Predicted Turnover Rate
195	8393	4.6	17.0	1	11.160904	38.064096
79	15845	4.7	13.0	0	36.258581	126.471217
480	19546	4.9	5.0	0	27.568406	178.927746
109	13677	4.2	6.0	1	29.038217	70.919196
522	6108	4.8	4.0	1	11.166362	15.118125

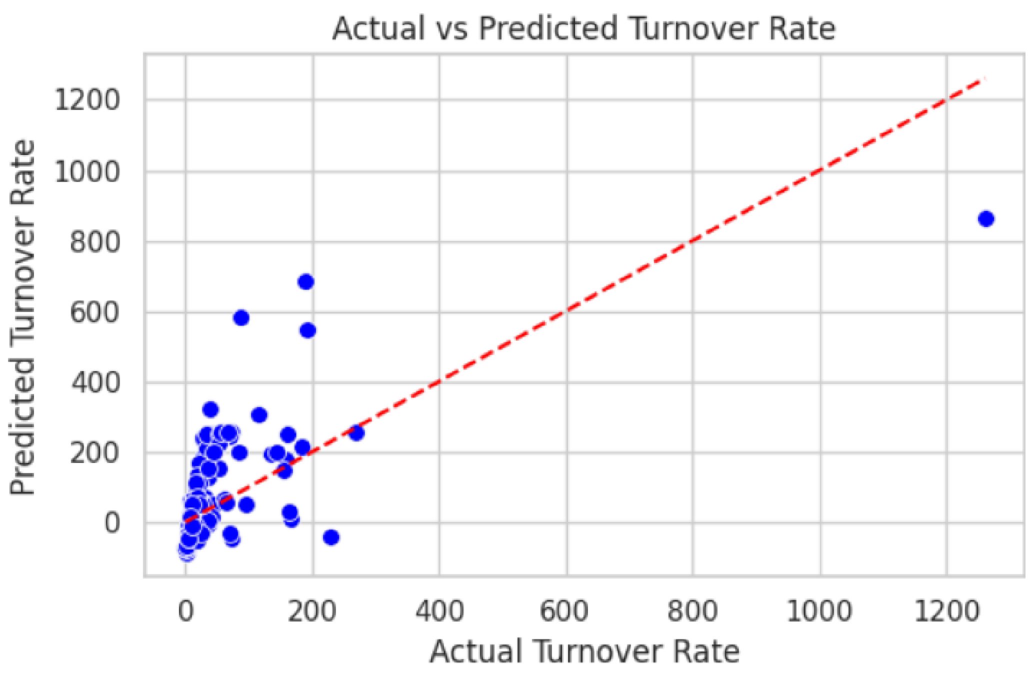
This code creates a new DataFrame predicted_df by copying the test features (X_test) and adding two new columns:

- **Actual Turnover Rate**, which contains the true values from y_test, and
- **Predicted Turnover Rate**, which holds the model's predictions (y_pred).

This allows for easy comparison between actual and predicted stock turnover rates for evaluation and visualization.

This scatter plot visualizes the relationship between the **actual** and **predicted** stock turnover rates. Each blue point represents a prediction; the closer it is to the red dashed line (which represents perfect prediction where actual = predicted), the more accurate the model is. Since the points are scattered far from the line, it visually confirms that the model's predictions are not very accurate.

```python
plt.figure(figsize=(6, 4))
sns.scatterplot(x=y_test, y=y_pred, color='blue', s=50)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
         color='red', linestyle='--')
plt.xlabel("Actual Turnover Rate")
plt.ylabel("Predicted Turnover Rate")
plt.title("Actual vs Predicted Turnover Rate")
plt.grid(True)
plt.tight_layout()
plt.show()
```



CONCLUSION

A linear regression model was developed to predict the **stock turnover rate** using key features such as **reviews**, **user rating**, **price**, and **genre**. However, the model's performance was relatively poor, with an **R^2 score** of approximately **0.009**, suggesting that these features do not sufficiently explain the variations in stock turnover. This indicates that the relationship is likely more complex and may depend on additional factors such as **promotion strategies, seasonal trends, or actual sales data**, which are not available in the current dataset.