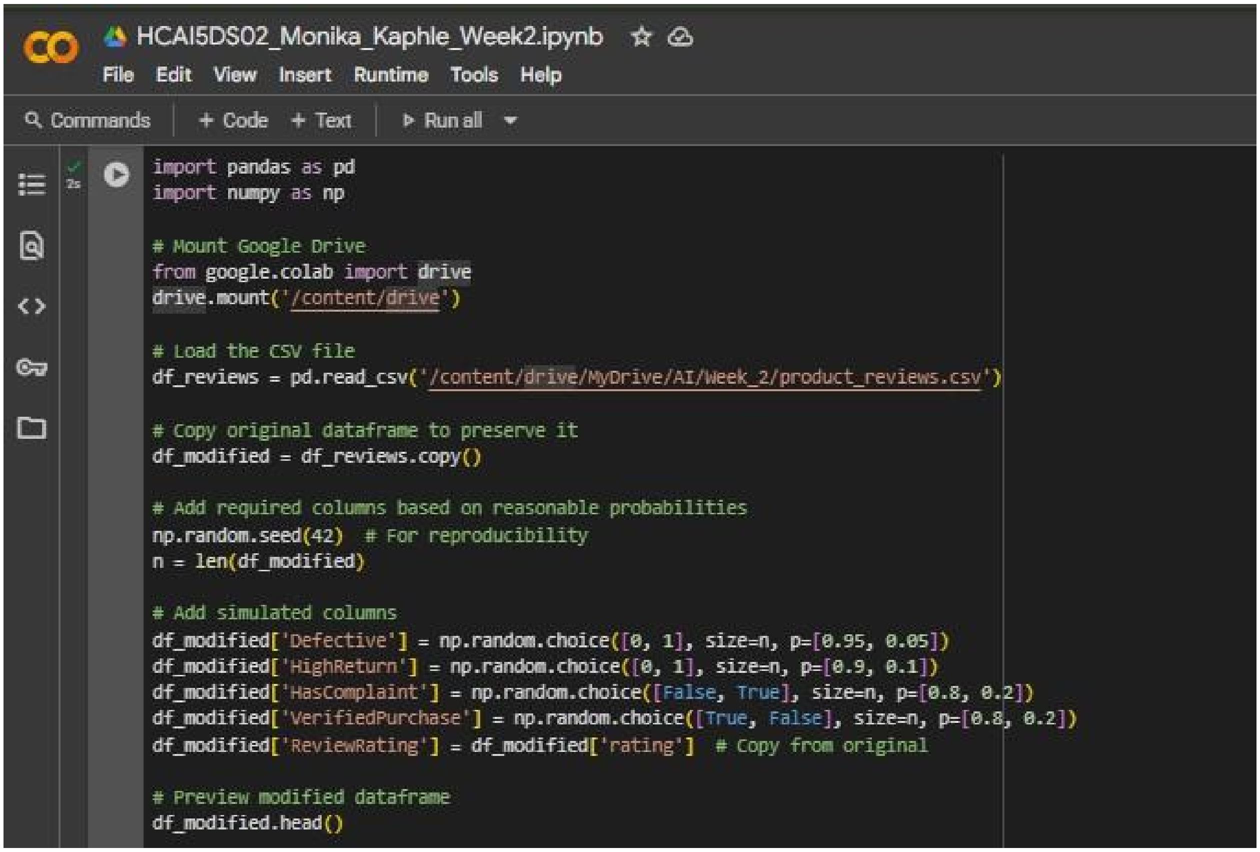
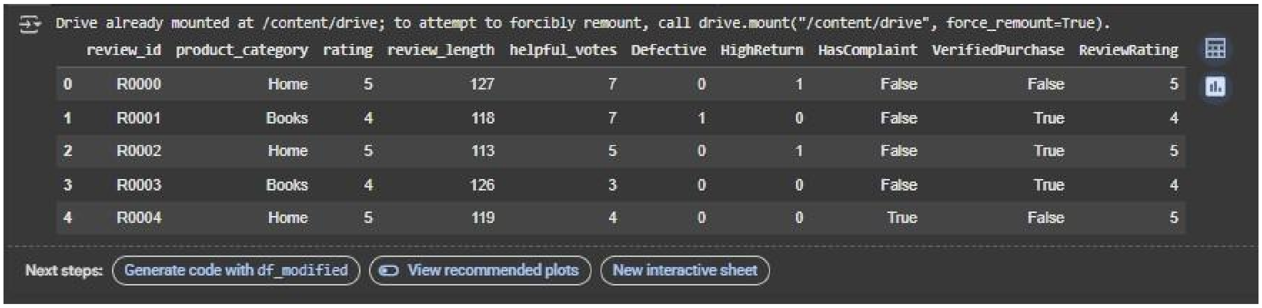
Monika Kaphle
2408878

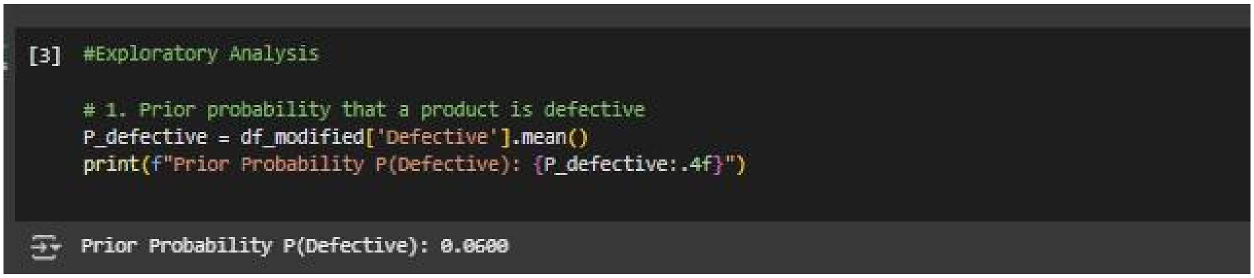

HCAI5DS02_Monika_Kaphle_Week2.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code + Text
Run all
2s

```python
import pandas as pd
import numpy as np

# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Load the CSV file
df_reviews = pd.read_csv('/content/drive/MyDrive/AI/Week_2/product_reviews.csv')

# Copy original dataframe to preserve it
df_modified = df_reviews.copy()

# Add required columns based on reasonable probabilities
np.random.seed(42)  # For reproducibility
n = len(df_modified)

# Add simulated columns
df_modified['Defective'] = np.random.choice([0, 1], size=n, p=[0.95, 0.05])
df_modified['HighReturn'] = np.random.choice([0, 1], size=n, p=[0.9, 0.1])
df_modified['HasComplaint'] = np.random.choice([False, True], size=n, p=[0.8, 0.2])
df_modified['VerifiedPurchase'] = np.random.choice([True, False], size=n, p=[0.8, 0.2])
df_modified['ReviewRating'] = df_modified['rating']  # Copy from original

# Preview modified dataframe
df_modified.head()
```



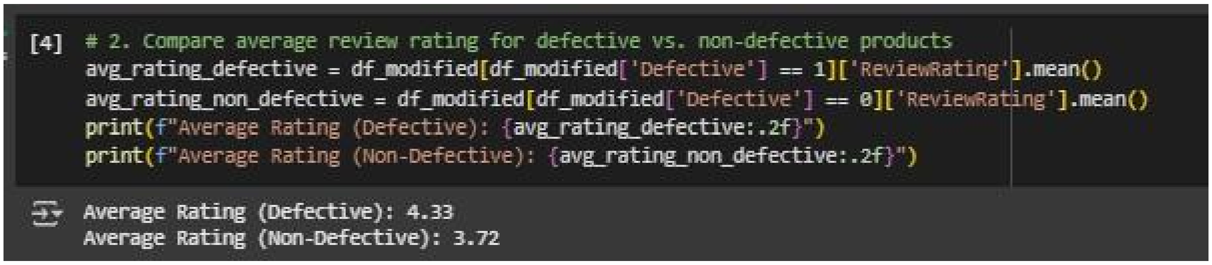
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

	review_id	product_category	rating	review_length	helpful_votes	Defective	HighReturn	HasComplaint	VerifiedPurchase	ReviewRating
0	R0000	Home	5	127	7	0	1	False	False	5
1	R0001	Books	4	118	7	1	0	False	True	4
2	R0002	Home	5	113	5	0	1	False	True	5
3	R0003	Books	4	126	3	0	0	False	True	4
4	R0004	Home	5	119	4	0	0	True	False	5



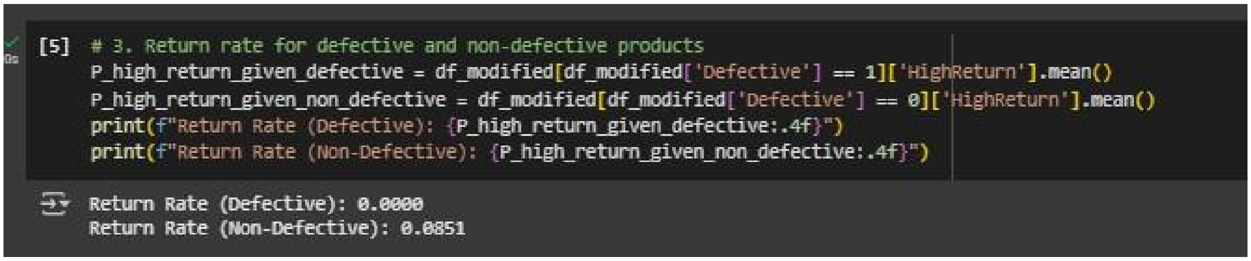
```python
[3] #Exploratory Analysis

    # 1. Prior probability that a product is defective
    P_defective = df_modified['Defective'].mean()
    print(f"Prior Probability P(Defective): {P_defective:.4f}")
```

```
Prior Probability P(Defective): 0.0600
```



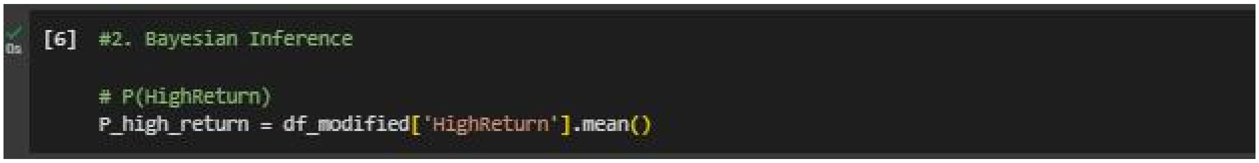
```
[4] # 2. Compare average review rating for defective vs. non-defective products
    avg_rating_defective = df_modified[df_modified['Defective'] == 1]['ReviewRating'].mean()
    avg_rating_non_defective = df_modified[df_modified['Defective'] == 0]['ReviewRating'].mean()
    print(f"Average Rating (Defective): {avg_rating_defective:.2f}")
    print(f"Average Rating (Non-Defective): {avg_rating_non_defective:.2f}")
```

```
Average Rating (Defective): 4.33
Average Rating (Non-Defective): 3.72
```

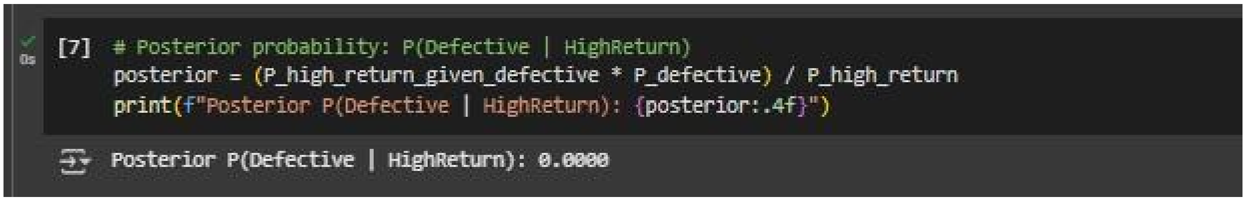


```
[5] # 3. Return rate for defective and non-defective products
    P_high_return_given_defective = df_modified[df_modified['Defective'] == 1]['HighReturn'].mean()
    P_high_return_given_non_defective = df_modified[df_modified['Defective'] == 0]['HighReturn'].mean()
    print(f"Return Rate (Defective): {P_high_return_given_defective:.4f}")
    print(f"Return Rate (Non-Defective): {P_high_return_given_non_defective:.4f}")
```

```
Return Rate (Defective): 0.0000
Return Rate (Non-Defective): 0.0851
```

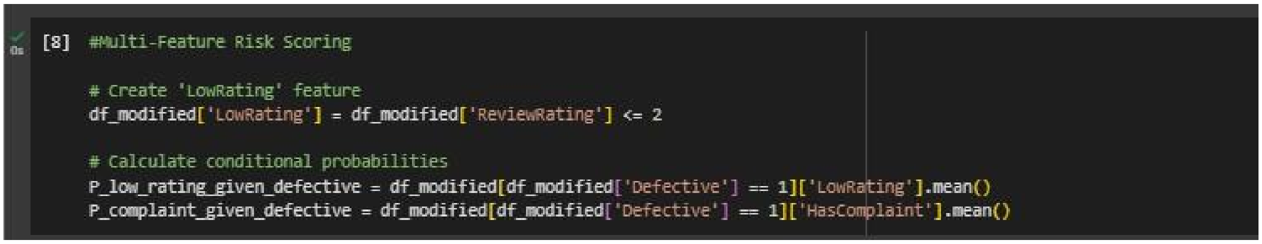


```
[6] #2. Bayesian Inference

    # P(HighReturn)
    P_high_return = df_modified['HighReturn'].mean()
```

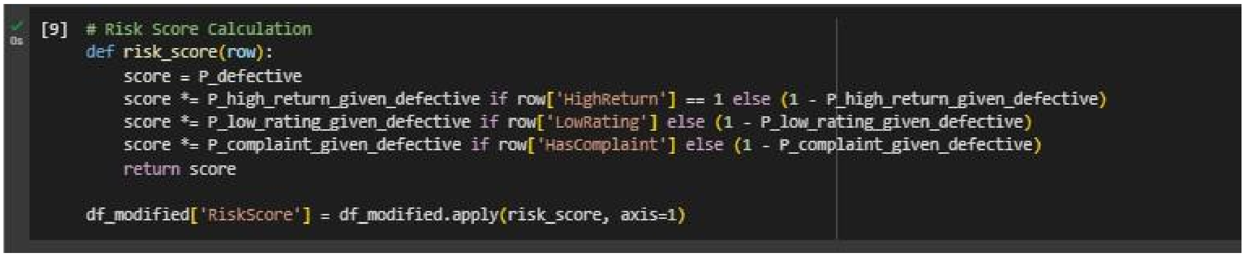


```
[7] # Posterior probability: P(Defective | HighReturn)
    posterior = (P_high_return_given_defective * P_defective) / P_high_return
    print(f"Posterior P(Defective | HighReturn): {posterior:.4f}")
```

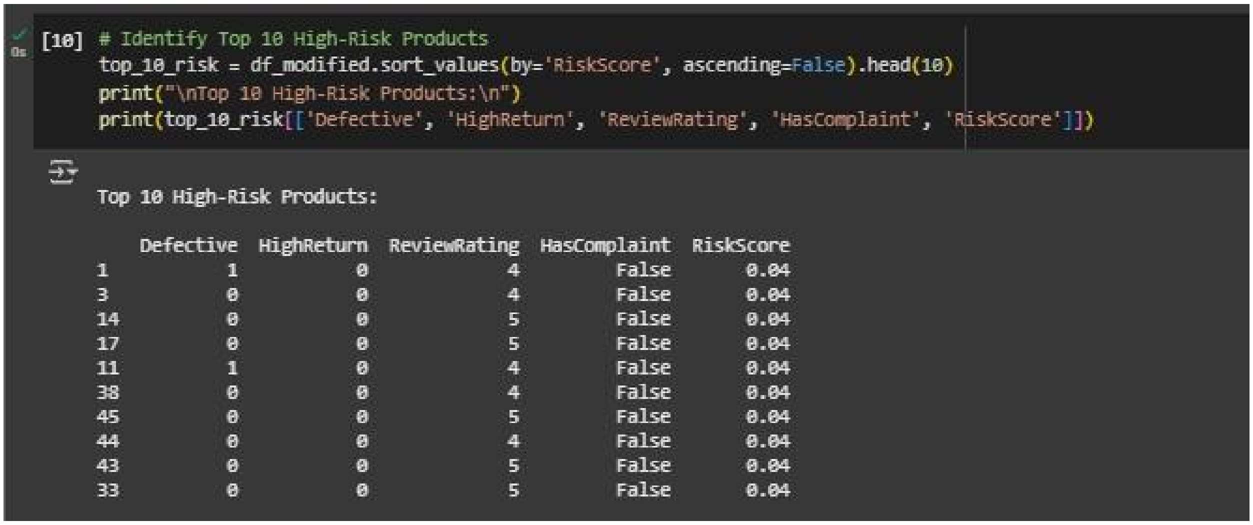
```
Posterior P(Defective | HighReturn): 0.0000
```



```
[8] #Multi-Feature Risk Scoring

    # Create 'LowRating' feature
    df_modified['LowRating'] = df_modified['ReviewRating'] <= 2

    # Calculate conditional probabilities
    P_low_rating_given_defective = df_modified[df_modified['Defective'] == 1]['LowRating'].mean()
    P_complaint_given_defective = df_modified[df_modified['Defective'] == 1]['HasComplaint'].mean()
```

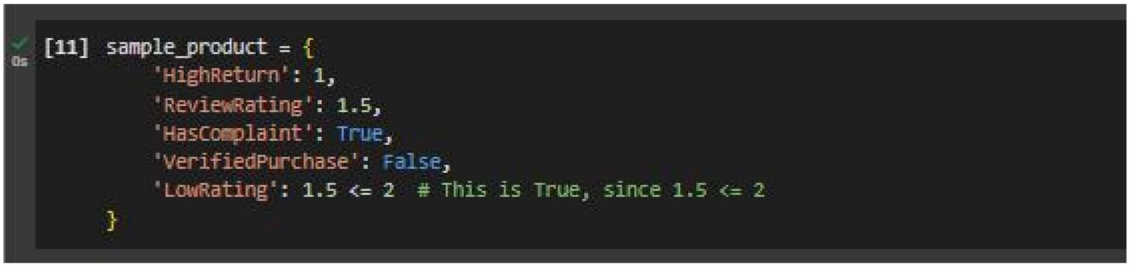


```
[9] # Risk Score Calculation
    def risk_score(row):
        score = P_defective
        score *= P_high_return_given_defective if row['HighReturn'] == 1 else (1 - P_high_return_given_defective)
        score *= P_low_rating_given_defective if row['LowRating'] else (1 - P_low_rating_given_defective)
        score *= P_complaint_given_defective if row['HasComplaint'] else (1 - P_complaint_given_defective)
        return score

    df_modified['RiskScore'] = df_modified.apply(risk_score, axis=1)
```

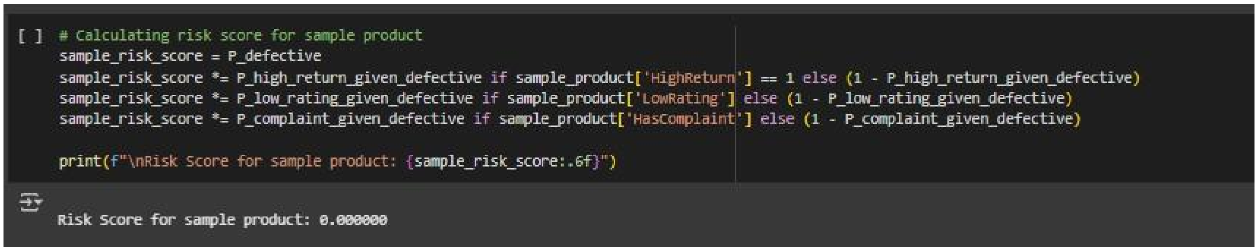


```
[10] # Identify Top 10 High-Risk Products
     top_10_risk = df_modified.sort_values(by='RiskScore', ascending=False).head(10)
     print("\nTop 10 High-Risk Products:\n")
     print(top_10_risk[['Defective', 'HighReturn', 'ReviewRating', 'HasComplaint', 'RiskScore']])
```

```
Top 10 High-Risk Products:

    Defective  HighReturn  ReviewRating  HasComplaint  RiskScore
1           1           0             4         False       0.04
3           0           0             4         False       0.04
14          0           0             5         False       0.04
17          0           0             5         False       0.04
11          1           0             4         False       0.04
38          0           0             4         False       0.04
45          0           0             5         False       0.04
44          0           0             4         False       0.04
43          0           0             5         False       0.04
33          0           0             5         False       0.04
```

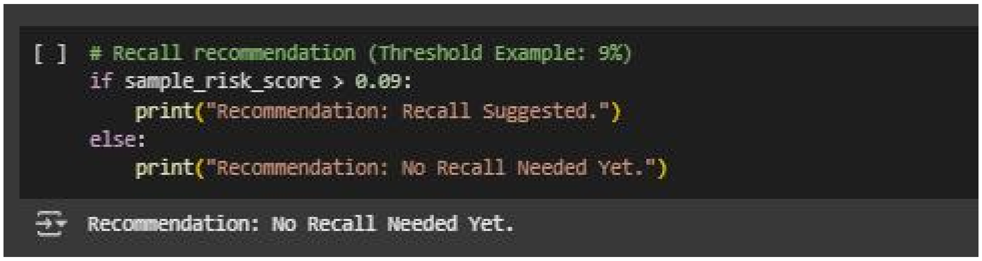


```
[11] sample_product = {
         'HighReturn': 1,
         'ReviewRating': 1.5,
         'HasComplaint': True,
         'VerifiedPurchase': False,
         'LowRating': 1.5 <= 2  # This is True, since 1.5 <= 2
     }
```



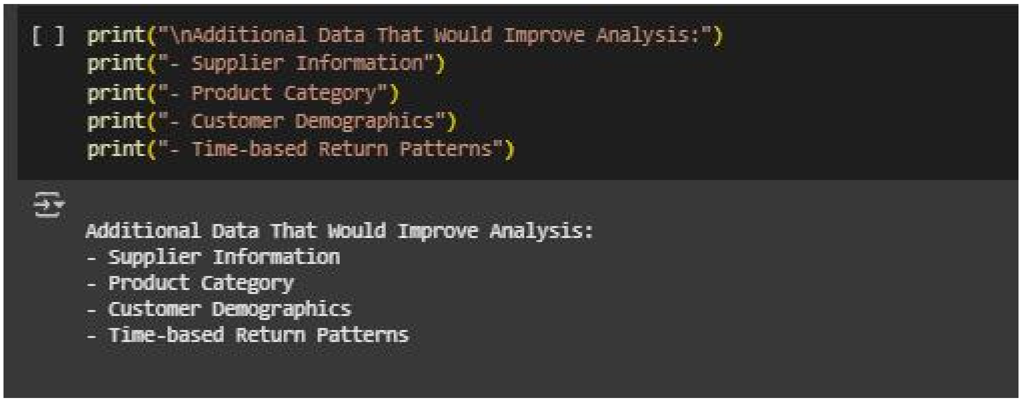
```
[ ] # Calculating risk score for sample product
    sample_risk_score = P_defective
    sample_risk_score *= P_high_return_given_defective if sample_product['HighReturn'] == 1 else (1 - P_high_return_given_defective)
    sample_risk_score *= P_low_rating_given_defective if sample_product['LowRating'] else (1 - P_low_rating_given_defective)
    sample_risk_score *= P_complaint_given_defective if sample_product['HasComplaint'] else (1 - P_complaint_given_defective)

    print(f"\nRisk Score for sample product: {sample_risk_score:.6f}")
```

```
Risk Score for sample product: 0.000000
```



```
[ ] # Recall recommendation (Threshold Example: 9%)
    if sample_risk_score > 0.09:
        print("Recommendation: Recall Suggested.")
    else:
        print("Recommendation: No Recall Needed Yet.")
```

```
Recommendation: No Recall Needed Yet.
```



```
print("\nAdditional Data That Would Improve Analysis:")
print("- Supplier Information")
print("- Product Category")
print("- Customer Demographics")
print("- Time-based Return Patterns")
```

```

Additional Data That Would Improve Analysis:
- Supplier Information
- Product Category
- Customer Demographics
- Time-based Return Patterns
```