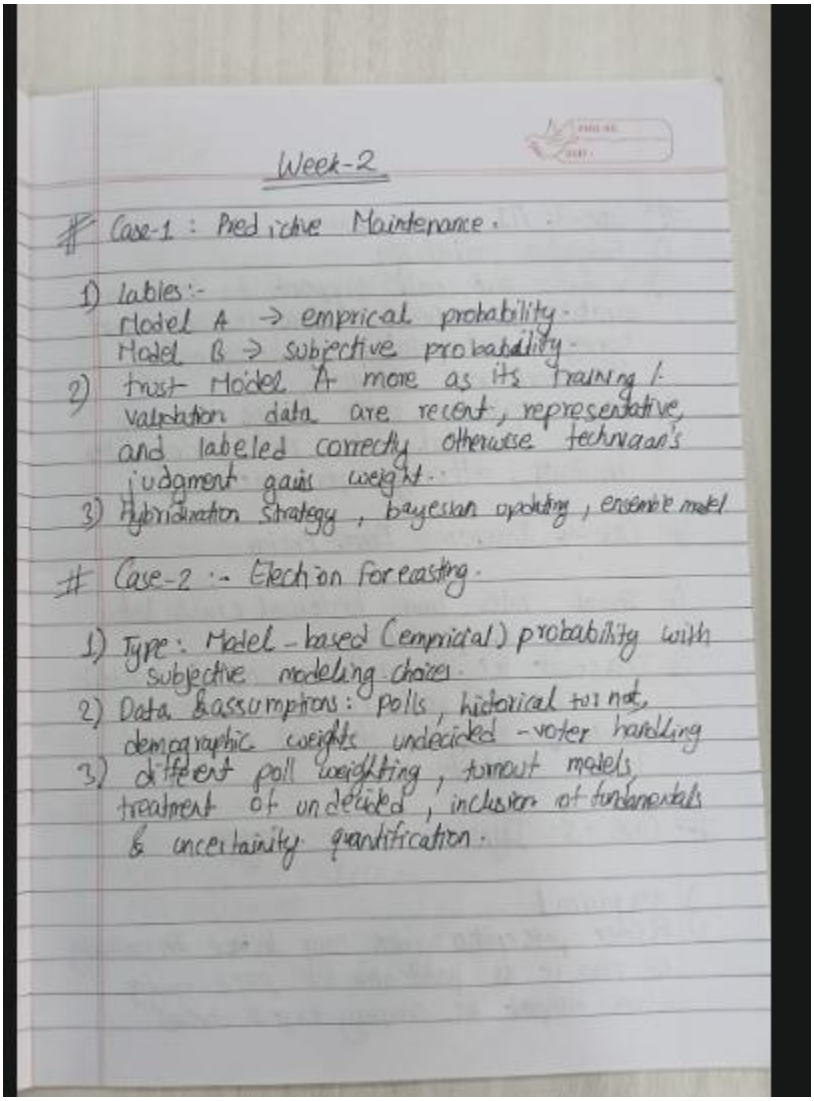
Week-2

Workshop-2

AI



Week-2

Case-1 : Predictive Maintenance.

1) Labels :-
Model A → emprical probability.
Model B → subjective probability.
2) trust Model A more as its training / validation data are recent, representative, and labeled correctly otherwise technician's judgment gains weight.
3) Hybridization strategy, bayesian updating, ensemble model

Case-2 :- Election Forecasting.

1) Type : Model - based (Empirical) probability with subjective modeling choices.
2) Data & assumptions : polls, historical turnout, demographic weights undecided - voter handling
3) different poll weighting, turnout models, treatment of undecided, inclusion of fundamentals & uncertainty quantification.

3) Test surge = demand - driven vs exploitative: run A/B pricing tests compare acceptance / cancellation rates, examine elasticity, control for weather & driver supply, and use difference-in differences across matched areas.

3) Q.E.D

1) $P(A^c) = 1 - P(A)$
$\Omega = A \cup A^c$ & $A \cap A^c = \emptyset$
By additivity: $P(\Omega) = P(A) + P(A^c)$
$P(\Omega) = 1$
$P(A^c) = 1 - P(A)$ //

2) $P(A \cap B) = P(A|B) \cdot P(B)$
$P(A|B) = P(A \cap B) / P(B)$
rearrange :- $P(A \cap B) = P(A|B) \cdot P(B)$.

3) $P(A|B) = P(A)$
$P(A \cap B) = P(A) P(B)$
$P(A|B) = P(A \cap B) / P(B)$
$P(A) P(B) / P(B)$
$P(A)$. //

Case-3: AI system failure
1) Subjective probability.
2) Evidence that could support it: trends in capability, empirical failure incident, expert surveys, extrapolated capability curves theoretical risk analyses.
3) deep uncertainty, vague failure definition, sparse historical data, model extrapolation sensitivity, differing expert priors.

Case-4: Insurance ~~Price~~ Pricing.

1) Initial rates from historical crash data → Empirical probability.
2) Discount for course: empirical if backed by observed post-course claim reductions. Subjective if based on expert belief without data.

Case-5: Dynamic Pricing

1) empirical.
2) Rider perception: riders may dispute the probability or view it as justification for price-gouging often interpret as company expects demand.

3) $P(A \cup P) = P(A) + P(P) - P(A \cap P)$

$0.25 + 0.15 - 0.10 = 0.30$

$P(A \cup P) = 0.30 = 30\%$ //

Case-2 : College Course Enrollment

→ Given, out of 1000 students:

$DS = 600$ std

$P(DS) = \frac{600}{1000} = 0.60$

$AI = 450$ std

$P(AI) = \frac{450}{1000} = 0.45$

$P(DS \cap AI) = \frac{250}{1000} = 0.25$

1) at least one course (use inclusion-exclusion)

$P(DS \cup AI) = P(DS) + P(AI) - P(DS \cap AI)$

$0.60 + 0.45 - 0.25 = 0.80$

$= 0.80 = 80\%$ //

2) neither course

$P(\text{neither}) = 1 - P(DS \cup AI) = 1 - 0.80 = 0.20 = 20\%$

3) done in 1.

4) Bayes Rule:-

Conditional probabilities:

$P(A|B) = P(B|A)\,P(A)/P(B)$

$P(A|B)P(B)$ & $P(B|A)P(A)$

5) law of total probability

Decompose

$A = (A \cap B) \cup (A \cap B^c)$

disjoint

$P(A) = P(A \cap B) + P(A \cap B^c) = P(A|B)P(B) + P(A|B^c)P(B^c)$

4) Probability Puzzlers: Computational Casebook.

Case-1 : E-Commerce Purchase Behavior

→ Given

$P(V) = 0.60$ (views products).

$P(A) = 0.25$ (adds to cart)

$P(P) = 0.15$ (makes purchases)

$P(A \cap P) = 0.10$ (adds to cart & purchase.

1) $P(A \& !P) = P(A) - P(A \cap P)$

$0.25 - 0.10 = 0.15 = 15\%$ //

2) $P(A \cap P) = P(A) \cdot P(P)$.

$P(A) \cdot P(P) = 0.25 \times 0.15$

$= 0.0375$

$P(A \cap P) = 0.10 \neq 0.0375$ so not independent //

Case-4:- Online Streaming Habits.

given

$P(M) = 0.60$

$P(S) = 0.40$

$P(M \cap S) = 0.20$

1) $P(\text{only movies}) = P(M) - P(M \cap S)$

$= 0.60 - 0.20 = 0.40 = 40\%$.

$P(\text{only series}) = P(S) - P(M \cap S)$

$= 0.40 - 0.20 = 0.20 = 20\%$.

$P(\text{neither}) = 1 - P(M \cup S)$

$P(M \cup S) = P(M) + P(S) - P(M \cap S)$

$= 0.60 + 0.40 - 0.20$

$= 0.80 = 80\%$.

So, $P(\text{neither}) = 1 - 0.80 = 0.20 = 20\%$.

2) movie watching & series watching mutually exclusive?

No as $P(\text{both}) = 0.20 > 0$.

Case-3: University Library Usage.

⇒ given: out of 1200 stds.

$P(D) = \frac{700}{1200} = 0.583$ (Digital resource)

$P(PB) = \frac{500}{1200} = 0.416$ (Physical books)

$P(D \cap PB) = \frac{300}{1200} = 0.25$ (Both)

1) probability a student uses neither.

Union :-

$P(D \cup PB) = P(D) + P(PB) - P(D \cap PB)$

$= 0.583 + 0.416 - 0.25$

$= 0.75$

neither : $1 - 0.75 = 0.25 = 25\%$.

2) events "uses digital" & "uses physical books" disjoint?

- No as $P(D \cap PB) = 0.25 > 0$ They overlab.

3) Axioms used : inclusion - exclusion (additivity for disjoint unions), complement rule.

4) What's overall chance of seeing this result?
→ Evidence also called marginal likelihood.

5) Case study.

Case-1: Fraud or False Alarm.

P(A) = 2000/100000 = 0.02
P(F) = 500/100000 = 0.005
P(A|F) = 490/500 = 0.98
P(A|F') = 1510/99500 = 0.01517

~~Bayes~~ Bayes formula:

$$P(F|A) = \frac{P(A|F) \cdot P(F)}{P(A)} = \frac{0.98 \times 0.005 = 0.0049}{0.02 \ (2000/100000)}$$

$$= \frac{0.0049}{0.02}$$
$$= 0.245$$
$$= 24.5\%.$$

probability of flagged transaction not fraud:
1 - 0.245 = 0.755 = 75.5%.

Case-5: Smartphone Failure Report.

Given.
P(B) = 0.05
P(S) = 0.03
P(B∩S) = 0.01

1) probability a phone has either issue
P(B∪S) = P(B) + P(S) - P(B∩S) = 0.05 + 0.03 - 0.01
= 0.07 = 7%.

2) Axiom relate:
non-negativity: all probabilities ≥ 0
additivity / inclusion-exclusion for union of two events: used above.

5) Data Dilemmas

1) ~~Common~~ How common is the condition in population
→ Prior.
2) If condition is true, how likely is this test result?
→ likelihood
3) If we observed this result, how likely is the condition?
→ Posterior

Case-3: Is Clicking the Email a Good Predictor of Conversion?

N = 25

P(S) = 8125 = 0.32

P(C) = 10125 = 0.40

P(C | S) = 718 = 0.875

P(S | C)

Bayes formula.

$$P(S|C) = \frac{P(C|S)\,P(S)}{P(C)} = \frac{0.875 \times 0.32}{0.40}$$

$$= \frac{0.28}{0.40}$$

$$= 0.70 = 70\%$$ //

Case-2:- Is this a Problem product?

① Prior P(Defective) = 0.04

P(HighReturn | Defective) = 0.90

P(HighReturn | ~Defective) = 0.05

P(Defective | HighReturn) = ?

P(HighReturn) = P(HighReturn | Defective) P(Defective) + P(HighReturn | ~Defective) P(~Defective)

② P(~Defective) = 1 - 0.04 = 0.96

First term = 0.90 × 0.04 = 0.036

Second term = 0.05 × 0.96 = 0.048

Sum = 0.036 + 0.048 = 0.084.

P(HighReturn) = 0.084

③ P(Defective | HighReturn) = $\frac{0.90 \times 0.04}{0.084} = \frac{0.036}{0.084}$

= 0.428

= 42.8%

HCAI5DS02_AnanyaDahal_2408840_Week-2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Files

- ..
- drive
- sample_data
  - README.md
  - anscombe.json
  - california_housing_test.csv
  - california_housing_train.csv
  - mnist_test.csv
  - mnist_train_small.csv
- product_reviews.csv

```
#Ananya Dahal
#2408840

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/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

HCAI5DS02_AnanyaDahal_2408840_Week-2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Files

- ..
- drive
- sample_data
  - README.md
  - anscombe.json
  - california_housing_test.csv
  - california_housing_train.csv
  - mnist_test.csv
  - mnist_train_small.csv
- product_reviews.csv

```
[13] #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
```

```
[14] from google.colab import drive
     drive.mount('/content/drive')
```

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

```
# 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
```

```
# 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}")
```

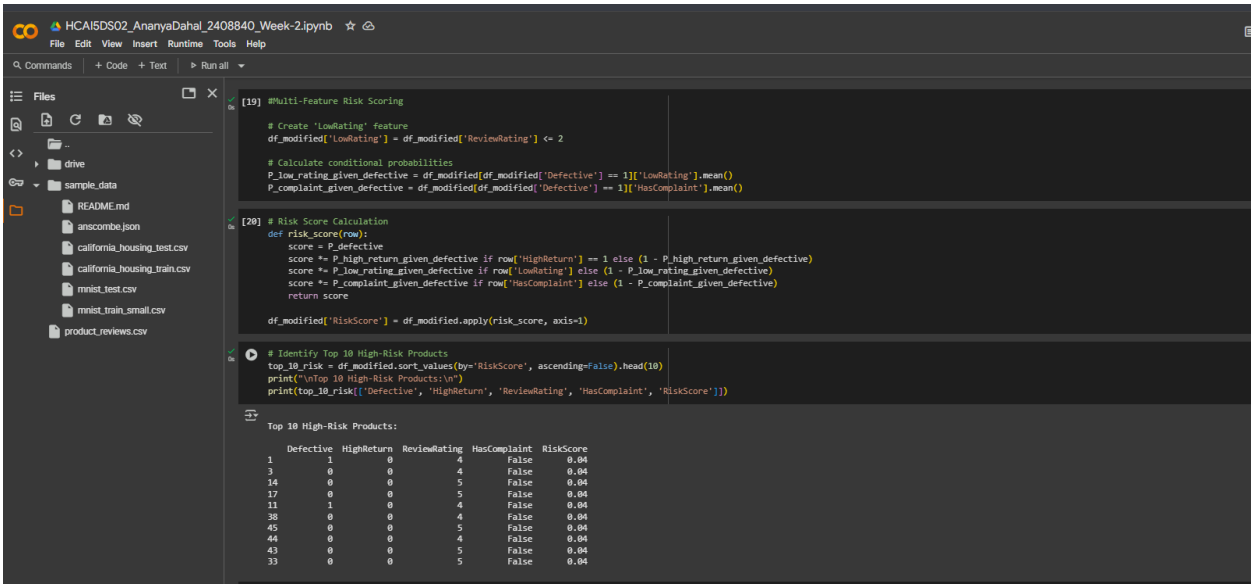
```
Return Rate (Defective): 0.0000
Return Rate (Non-Defective): 0.0851
```

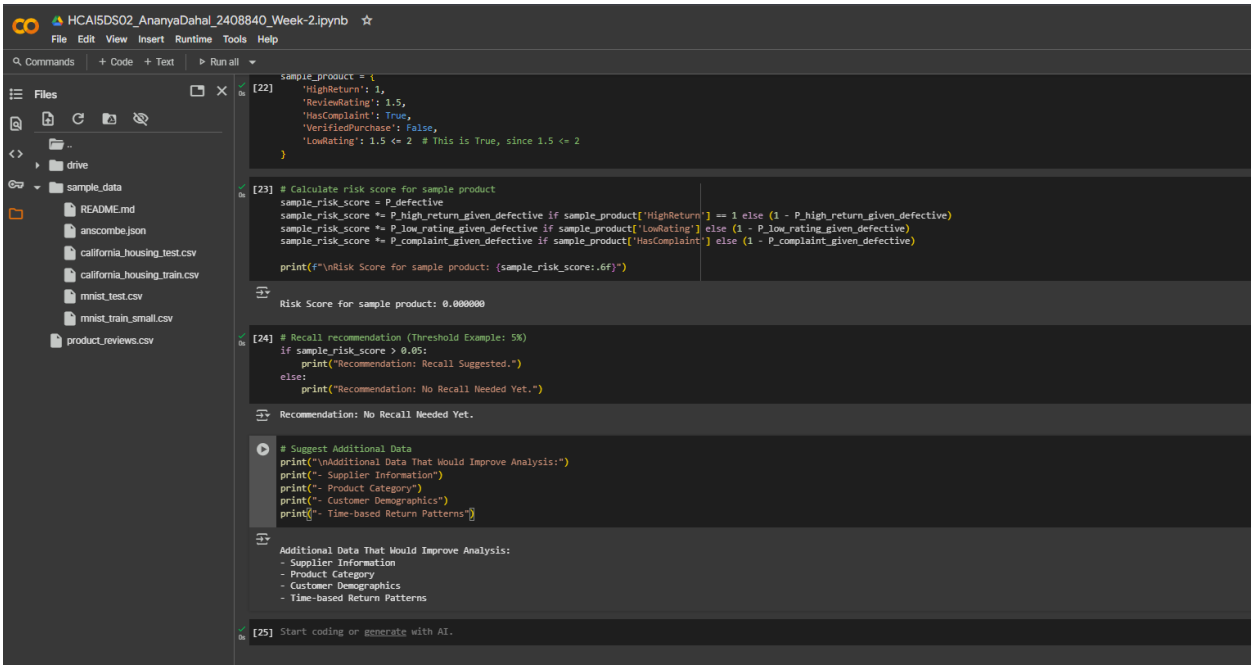
```
[17] #2. Bayesian Inference

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

```
[18] # 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
```


HCAI5DS02_AnanyaDahal_2408840_Week-2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
drive
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
product_reviews.csv
[19] #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()
[20] # 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)
# 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


HCAI5DS02_AnanyaDahal_2408840_Week-2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
drive
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
product_reviews.csv
[22] sample_product = {
'HighReturn': 1,
'ReviewRating': 1.5,
'HasComplaint': True,
'VerifiedPurchase': False,
'LowRating': 1.5 <= 2 # This is True, since 1.5 <= 2
}
[23] # Calculate 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
[24] # Recall recommendation (Threshold Example: 5%)
if sample_risk_score > 0.05:
print("Recommendation: Recall Suggested.")
else:
print("Recommendation: No Recall Needed Yet.")
Recommendation: No Recall Needed Yet.
# Suggest Additional Data
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
[25] Start coding or generate with AI.