Rohit Dutta
1010683

27/9/25

SI 104 Machine learning.

(1) $A \in \{0, 1\}$.
$B \in \{0, 1, 2\}$.

$P(A=0, B=0) = \frac{1}{24}$ (P_{00}).

$P(A=0, B=1) = \frac{1}{12}$ (P_{01}).

$P(A=0, B=2) = \frac{1}{8}$ (P_{02}).

$P(A=1, B=0) = \frac{1}{8}$ (P_{10})

$P(A=1, B=1) = \frac{1}{4}$ (P_{11})

$P(A=1, B=2) = (P_{12})$.

(a) $\sum P_{ij} = 1$. (condition for JOINT PROBABILITY).

$\therefore \frac{1}{24} + \frac{1}{12} + \left(\frac{1}{8} + \frac{1}{8}\right) + \frac{1}{4} + P_{12} = 1$,

$\Rightarrow \frac{1}{24} + \frac{1}{12} + \frac{1}{2} + P_{12} = 1$,

$\Rightarrow P_{12} + \frac{12 + 2 + 1}{24} = 1$,

$\Rightarrow P_{12} = \frac{24 - 12 - 2 - 1}{24} = \frac{\cancel{9}\,3}{\cancel{24}\,8}$

$\Rightarrow \boxed{P(A=1, B=2) = \left(3/8\right)}$.

(b) $P(B=0) = P(B=0, A=1) + P(B=0, A=0)$

$$= \frac{1}{24} + \frac{1}{8} = \frac{4}{24} = \frac{1}{6}.$$

$P(B=1) = P(A=0, B=1) + P(A=1, B=1).$

$$= \frac{1}{12} + \frac{1}{4} = \frac{4}{12} = \frac{1}{3}$$

$P(B=2) = P(A=0, B=2) + P(A=1, B=2)$

$$= \frac{1}{8} + \frac{3}{8} = \frac{4}{8} = \frac{1}{2}.$$

(c) $\boxed{\text{Expectation}}$ (for discrete variable B).

$$\longrightarrow \sum_{x \in B \in \{0,1,2\}} x \cdot P(x) = 0 \cdot \frac{1}{6} + 1 \cdot \frac{1}{3} + 2 \cdot \frac{1}{2}$$

$$= 1 + \frac{1}{3} = 4/3.$$

$\boxed{\text{Variance}}$ (for discrete variable B).

$$\longrightarrow \sum x^2 \cdot P(x) - \left[\sum x \cdot P(x)\right]^2.$$

$$\Rightarrow \left(0^2 \cdot \frac{1}{6} + 1^2 \cdot \frac{1}{3} + 2^2 \cdot \frac{1}{2}\right) - \left(4/3\right)^2.$$

$$= \frac{1}{3} + 2 - \frac{16}{9} = \frac{3}{9} + \frac{18}{9} - \frac{16}{9} = \frac{3}{9} + \frac{2}{9}$$

$$= 5/9.$$

(d) Marginal distribution of A.

$P(A=0) = \cancel{P(A=0, B}$

$p_{00} + p_{01} + p_{02} = \frac{1}{24} + \frac{1}{12} + \frac{1}{8} =$

$= \frac{3+2+1}{24} = \frac{6}{24} = \frac{1}{4}$

$P(A=1) = p_{10} + p_{11} + p_{12} = \frac{1}{8} + \frac{1}{4} + \frac{3}{8}$

$= \frac{4}{8} + \frac{2}{8} = \frac{6}{8} = \frac{3}{4}$

For joint distributions, the condition for both of the random variables to be independent, the marginal dists must be COMPLETE, and the JOINT DISTS should be product of individual MARGINAL DISTS.

$P(A=0, B=2) = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8} = P(A=0) \times P(B=2)$

→ $P(A=i, B=j) = P(A=i) \cdot P(B=j) \ \forall\ i, j \in A, B$

∴ A & B are independent variables.

1.) $P(X=x) = (1-\theta)\cdot\theta^{x}$ $\left[\theta \in (0,1)\right]$.

(a) x are discrete ~~varia~~ integers.

$P(x_1)\cdot P(x_2)\cdot P(x_3)\cdots\cdots P(x_n)$

$$\text{MLE} \Rightarrow \prod_{x=1}^{n}(1-\theta)\,\theta^{x} = (1-\theta)^{n}\,\theta^{\sum_i^n x_i}.$$

$$\log(\text{MLE}) = n\log(1-\theta) + \log(\theta)\left(\sum_i^n x_i\right).$$

$$\frac{\partial}{\partial\theta}\log(\text{MLE}) = -\frac{n}{1-\theta} + \frac{\sum_i^n x_i}{\theta} = 0.$$

$$\Rightarrow \frac{n}{1-\theta} = \frac{\sum_i x_i}{\theta}.$$

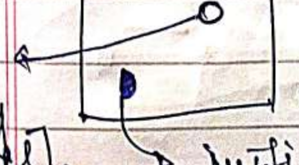
$$\Rightarrow \frac{1}{n} - \frac{\theta}{n} = \frac{\theta}{\sum_i x_i}$$

$$= \frac{1}{n} = \theta\left(\frac{1}{n} + \frac{1}{\sum_i x_i}\right) \Rightarrow \theta = \frac{1/n}{\frac{1}{n} + \frac{1}{\sum_i x_i}}$$

$$\boxed{\hat{\theta} =} \frac{1}{1 + \frac{n}{\sum_i x_i}} = \frac{\sum_i x_i}{\sum_i x_i + n} \rightarrow (\text{ANS}).$$

$$\text{ANS} \rightarrow \hat{\theta} = \frac{\sum_i x_i}{\sum_i x_i + n} \text{ is the MLE}$$

(b) $\theta = \frac{1}{2}$ (equal probability of small/big)



P. [~~First initial~~ ~~initial~~ First removed item being small].

= P(second item chosen) $\times$ P(second item being small)

+ P(first item chosen) $\times$ P(first item being small).

$= \frac{1}{2} \times 1 + \frac{1}{2} \times \frac{1}{2} = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$.

$\therefore B = 3/4$.

P(remaining item is small)

= P(~~initial~~ first item chosen) . P(final item is small)

+ P(final first item chosen) . P(initial item is small)

$= \frac{1}{2} \times 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. $\therefore C = 3/4$.

Question: what is the probability that the remaining item is small given that the first item was definitely small. = P(C | B)

$$\cancel{P(C|B) = \frac{P(C)}{P(B)} = \frac{3/4}{3/4} = 1}.$$

$$P(C|B) = \frac{P(C \cap B)}{P(B)}$$ (Baye's theorem / conditional probability law)

$\therefore P(C \cap B)$ = first item is small AND remaining item is also small.

$$= \underbrace{\frac{1}{2}}_{\text{first item (small)}} \cdot \left(\frac{1}{2} \cdot \underbrace{1}_{\text{second item (small)}}\right) + \underbrace{\frac{1}{2}}_{\text{first item (small)}} \left(1 \cdot \underbrace{\frac{1}{2}}_{\text{second item (small)}}\right) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.$$

$$\therefore P(C|B) = \frac{P(B \cap C)}{P(B)} = \frac{1/2}{3/4} = 2/3$$

---

(3)

(a)

$$\mu_1 = \frac{2+8}{\sum_i c_i} = \frac{2+8}{2} = 1+4 = 5.$$

$$\mu_2 = \frac{1+3+9+10}{\sum_i c_i} = \frac{23}{4} \approx 5.75.$$

(b) Nearest centroids to each pt.

(*) Since there are two centroids we MAY have 2 clusters.

① $\min(\underbrace{|1-5|}_{D_1}, \underbrace{|1-5.75|}_{D_2}) = 4 \ (D_1)$

② $\min(\underbrace{|2-5|}_{D_1}, \underbrace{|2-5.75|}_{D_2}) = 3 \ (D_1)$

③ $\min(\underbrace{|3-5|}_{D_1}, \underbrace{|3-5.75|}_{D_2}) = 2 \ (D_1)$

④ $\min(\underbrace{|8-5|}_{D_1}, \underbrace{|8-5.75|}_{D_2}) = 2.25 \ (D_2)$

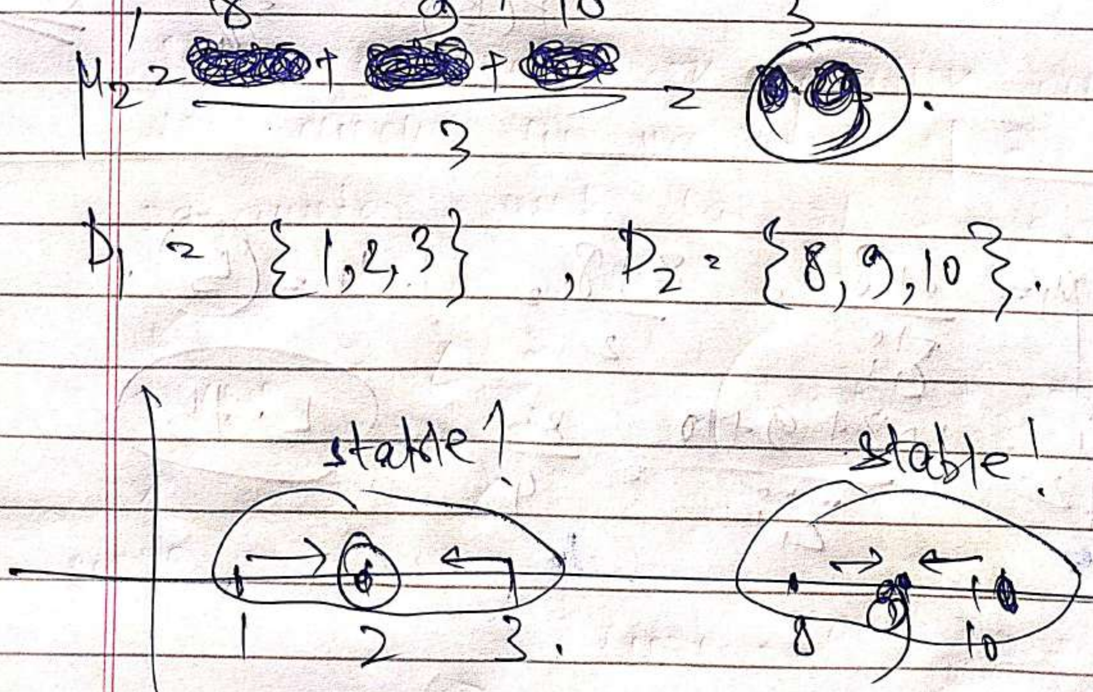
⑤ $\min(\underbrace{|9-5|}_{D_1}, \underbrace{|9-5.75|}_{D_2}) = 3.25 \ (D_2)$

⑥ $\min(\underbrace{|10-5|}_{D_1}, \underbrace{|10-5.75|}_{D_2}) = 4.25 \ (D_2)$.

(c) New centroids, $\mu_1' = \frac{1+3+2}{3} = 2$.

$\mu_2' = \frac{8+9+10}{3} = 9$.

$D_1 = \{1,2,3\}$, $D_2 = \{8,9,10\}$.



(4.) (a). $f(x,y) = \begin{cases} Kx^3y^2 \; ; \text{if } x \in [0,1], y \in [0,2]. \\ 0 \quad \text{if } x \notin [0,1], y \notin [0,2]. \end{cases}$

$\therefore \int_0^1 dx \int_0^2 dy \; Kx^3y^2 = 1.$ [Probability law].

$\Rightarrow K \left[\frac{y^3}{3}\right]_0^2 \left[\frac{x^4}{4}\right]_0^1 \Rightarrow K \frac{8}{3} \cdot \frac{1}{4} = 1.$

$\Rightarrow \boxed{K = 3/2}$

$\therefore \underset{(x,y)}{pdf} \Rightarrow 3/2 \, x^3 y^2 \; \forall \, x \in [0,1], y \in [0,2].$

(b) pdf(x) $\Rightarrow$ marginal probability of x.

$\Rightarrow 3/2 \, x \int_0^2 y^2 dy = 3/2 \cdot x^3 \cdot \left[\frac{y^3}{3}\right]_0^2 = 4x^3.$

$\therefore$ pdf$(x) = 4x^3$.

(c) Multivariate function : $3/2 \, x^3 y^2$.

we have to find the region where $(y<x)$ in order to find $P(X>Y)$.

We must remember $y \in [0, 2]$.

$\therefore$ for $x > y$, $y \in [0, 1]$.

Let's take the y component.

$\phi(y) = y^2$. $\therefore$ $\phi(y < x) = \int_0^x y^2 \, dy$

$= \left[\frac{y^3}{3}\right]_0^x = \frac{x^3}{3}$.

Total area of THAT

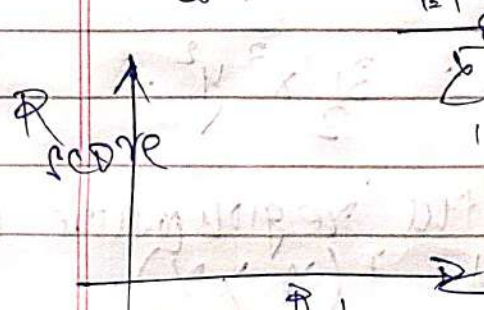
$= \int_0^1 \frac{3}{2} x^3 \cdot \frac{x^3}{3} \, dx = \frac{1}{2} \int_0^1 x^6 \, dx = \frac{1}{14}$.

---

(5) RIASEC codebook.

It's a regression question and to check the variation of R_1 parameter with that of R_{score}.

$R_{score} = \frac{\sum_{i=1}^{8} R_i}{\sum_1^8 i} = \frac{R_1 + R_2 + R_3 + R_4 + R_5 \dots R_8}{8}$.



$\boxed{R_{score} = \alpha R_1 + \beta}$.

RIASEC ANALYSIS

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [2]:
```
#df = pd.read_csv("RIASEC.csv")
```

Reading and Cleaning

In [3]:
```
df = pd.read_csv("RIASEC.csv", sep=None, engine="python")

print(df.shape)
print(df.head())
```

```
(8855, 55)
   implementation  R1  R2  R3  R4  R5  R6  R7  R8  I1  ...  C5  C6  C7  C8  \
0               2   3   1   4   2   1   2   1   1   5  ...   2   1   1   2
1               2   1   1   1   1   1   1   1   1   4  ...   1   1   1   1
2               2   3   2   1   1   1   1   2   1   5  ...   3   4   4   4
3               2   3   2   1   2   2   3   1   2   5  ...   1   3   2   1
4               2  -1   2   3   2   3   2   1   3   5  ...   4   3   3   3

   accuracy  elapse  country  fromsearch  age  gender
0        90     222       PT           0   -1      -1
1       100     102       US           0   -1      -1
2        95     264       US           1   -1      -1
3        60     189       SG           0   -1      -1
4        90     197       US           0   -1      -1

[5 rows x 55 columns]
```

In [4]:
```
cols = [f"R{i}" for i in range(1, 9)]
print(cols)
R_data = df[cols]
R_data.shape
```

```
['R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8']
```

Out[4]: (8855, 8)

In [5]:
```
df = R_data[(R_data != -1).all(axis=1)] # cleans the data and gets rid of rows w
print("Shape after cleaning:", df.shape)
```

```
Shape after cleaning: (8478, 8)
```

In [6]:
```
# the entries have reduced from 8855 people to 8478 meaning 377 people had inval
```

In [7]:
```
8855-8478
```

Out[7]: 377

In [8]:
```
df
```

Out[8]:

	R1	R2	R3	R4	R5	R6	R7	R8
0	3	1	4	2	1	2	1	1
1	1	1	1	1	1	1	1	1
2	3	2	1	1	1	1	2	1
3	3	2	1	2	2	3	1	2
5	3	1	3	4	3	4	3	3
...	...	...	...	...	...	...	...	...
8849	3	3	1	4	2	2	2	1
8851	3	2	3	4	3	2	2	2
8852	4	3	3	3	2	2	1	2
8853	4	4	3	5	4	5	3	4
8854	4	2	4	4	1	4	2	3

8478 rows × 8 columns

In [9]:

```
df["R1"]
```

Out[9]:

```
0       3
1       1
2       3
3       3
5       3
       ..
8849    3
8851    3
8852    4
8853    4
8854    4
Name: R1, Length: 8478, dtype: int64
```

Model selection

In [40]:

```
df = df.copy()
df["R_score"] = df.mean(axis=1) #row wise mean
df.shape
```

Out[40]:

```
(8478, 9)
```

In [41]:

```
train = df.iloc[:6500]
test = df.iloc[6500:]
train.shape, test.shape
```

Out[41]:

```
((6500, 9), (1978, 9))
```

In [42]:

```
x_train = train[["R1"]]
y_train = train["R_score"]
```

In [43]:
```
model = LinearRegression()
model.fit(x_train, y_train)
```

Out[43]:
```
▾ LinearRegression
LinearRegression()
```

In [44]:
```
print("Intercept:", model.intercept_)
print("Coefficient for R1:", model.coef_[0])

model.coef_.shape
```

```
Intercept: 1.018105231431332
Coefficient for R1: 0.42359934763433976
```

Out[44]:
```
(1,)
```

Hence the form of the best-fit regression line is $R_{score} = R_1 \times \text{coef} + \text{intercept}$

In [45]:
```
y_pred_train = model.predict(x_train)
RSS_train = np.sum((y_train - y_pred_train) ** 2) #RSS -> residual sum of square
RSS_train_avg = RSS_train / len(train)
print("Training RSS (total):", RSS_train)
print("Training RSS (average):", RSS_train_avg)
```

```
Training RSS (total): 2902.0393474685015
Training RSS (average): 0.446467591918231
```

Validation

In [46]:
```
x_test = test[["R1"]]
y_test = test["R_score"]
```

In [47]:
```
y_pred_test = model.predict(x_test)
RSS_test = np.sum((y_test - y_pred_test) ** 2)
RSS_test_avg = RSS_test / len(test)
```

In [48]:
```
print("Test RSS (total):", RSS_test)
print("Test RSS (average):", RSS_test_avg)
```

```
Test RSS (total): 1028.7757850021012
Test RSS (average): 0.5201090925187569
```

In [49]:
```
print(f"Average RSS (train): {RSS_train_avg:.5f}")
print(f"Average RSS (test): {RSS_test_avg:.5f}")
```

```
Average RSS (train): 0.44647
Average RSS (test): 0.52011
```

Model works better with training data

For other fields

Training

In [50]:
```
models = {}
RSS_train = {}
RSS_train_avg = {}

for i in range(2, 9):   # R2 to R8
    x_train = train[[f"R{i}"]]
    y_train = train["R_score"]

    models[i] = LinearRegression()
    models[i].fit(x_train, y_train)

    y_pred_train = models[i].predict(x_train)

    RSS_train[i] = np.sum((y_train - y_pred_train) ** 2)
    RSS_train_avg[i] = RSS_train[i] / len(train)

    print(f"R{i}:")
    print(f"  Training RSS (total): {RSS_train[i]}")
    print(f"  Training RSS (average): {RSS_train_avg[i]}\n")
```

```
R2:
  Training RSS (total): 2086.9934694791536
  Training RSS (average): 0.32107591838140825

R3:
  Training RSS (total): 2856.1295910903655
  Training RSS (average): 0.43940455247544086

R4:
  Training RSS (total): 2058.2069482183156
  Training RSS (average): 0.31664722280281776

R5:
  Training RSS (total): 2026.8220877210033
  Training RSS (average): 0.3118187827263082

R6:
  Training RSS (total): 1762.7887420860695
  Training RSS (average): 0.2711982680132415

R7:
  Training RSS (total): 1928.2557723088821
  Training RSS (average): 0.29665473420136645

R8:
  Training RSS (total): 1771.9059588635785
  Training RSS (average): 0.2726009167482428
```

In [51]:
```
dict(sorted(RSS_train_avg.items(), key=lambda item: item[1]))
```

Out[51]:
```
{6: np.float64(0.2711982680132415),
 8: np.float64(0.2726009167482428),
 7: np.float64(0.29665473420136645),
 5: np.float64(0.3118187827263082),
 4: np.float64(0.31664722280281776),
 2: np.float64(0.32107591838140825),
 3: np.float64(0.43940455247544086)}
```

In [52]:
```
keys = list(RSS_train_avg.keys())
values = list(RSS_train_avg.values())
best_feature = keys[np.argmin(values)]
print(f"Best feature is {best_feature} based on training data")
```

```
Best feature is 6 based on training data
```

Testing

In [53]:
```
models = {}
RSS_test = {}
RSS_test_avg = {}

for i in range(2, 9):   # R2 to R8
    x_test = test[[f"R{i}"]]
    y_test = test["R_score"]

    models[i] = LinearRegression()
    models[i].fit(x_test, y_test)

    y_pred_test = models[i].predict(x_test)

    RSS_test[i] = np.sum((y_test - y_pred_test) ** 2)
    RSS_test_avg[i] = RSS_test[i] / len(test)

    print(f"R{i}:")
    print(f"  Testing RSS (total): {RSS_test[i]}")
    print(f"  Testing RSS (average): {RSS_test_avg[i]}\n")
```

```
R2:
  Testing RSS (total): 712.0378480768431
  Testing RSS (average): 0.35997868962428875

R3:
  Testing RSS (total): 875.312355121421
  Testing RSS (average): 0.4425239409107285

R4:
  Testing RSS (total): 799.8393275744875
  Testing RSS (average): 0.4043677085816418

R5:
  Testing RSS (total): 697.1299733893734
  Testing RSS (average): 0.3524418470118167

R6:
  Testing RSS (total): 630.9467824594283
  Testing RSS (average): 0.31898219537888184

R7:
  Testing RSS (total): 680.1640236301748
  Testing RSS (average): 0.3438645215521612

R8:
  Testing RSS (total): 678.3905837840575
  Testing RSS (average): 0.3429679392234871
```

In [54]:
```
dict(sorted(RSS_test_avg.items(), key=lambda item: item[1]))
```

Out[54]:
```
{6: np.float64(0.31898219537888184),
 8: np.float64(0.3429679392234871),
 7: np.float64(0.3438645215521612),
 5: np.float64(0.3524418470118167),
 2: np.float64(0.35997868962428875),
 4: np.float64(0.4043677085816418),
 3: np.float64(0.4425239409107285)}
```

In [55]:
```
keys = list(RSS_test_avg.keys())
values = list(RSS_test_avg.values())
best_feature = keys[np.argmin(values)]
print(f"Best feature is {best_feature} based on testing data")
```

```
Best feature is 6 based on testing data
```

We can collectively agree that for mean value based regression, 6 is the best performing fitting parameter.

In [ ]: