Answer to Sample Exam 1

Question 1

(1.1) Implement from scratch a Python function for simple numerical encoding. This function takes a list of string values as input and returns a vector of integers as output. Write down the Python code.

Ans:

```
def simple_encoding(values):
    unique_values = list(set(values))
    encoding = {val: idx for idx, val in enumerate(unique_values)}
    return [encoding[val] for val in values]
# Example:
print(simple_encoding(["cat", "dog", "cat", "mouse"]))
```

(1.2) Implement from scratch a Python function to compute the Gini index of a list. This function takes a list of categorical values as input and returns the Gini index as output. Write down the Python code.

Ans:

```
def gini_index(values):
    from collections import Counter
    count = Counter(values)
    total = len(values)
    gini = 1 - sum((freq / total) ** 2 for freq in count.values())
    return gini
# Example:
print(gini_index(["low", "low", "high", "high", "high"]))
```

Question 2

(2.1) Explain why pre-processing is important in big data.

Ans:

Importance of Pre-processing:
Pre-processing improves data quality and model performance by:

- **Removing noise and handling missing values.**
- **Normalizing features to ensure fair comparisons.**
- **Encoding categorical data into numerical form for ML models.**

(2.2) Explain the advantages and disadvantages of data aggregation.

Ans:

Advantages:

- **Reduces data volume.**
- **Simplifies analysis.**
- **Enhances pattern visibility.**

Disadvantages:

- **Loss of detail.**
- **Possible distortion of patterns.**
- **Risk of misleading results.**

(2.3) Explain undersampling and oversampling, and when you will apply them.

Ans:

Undersampling: Reduces majority class size.

- ***Used when*: Large datasets and computation needs to be optimized.**

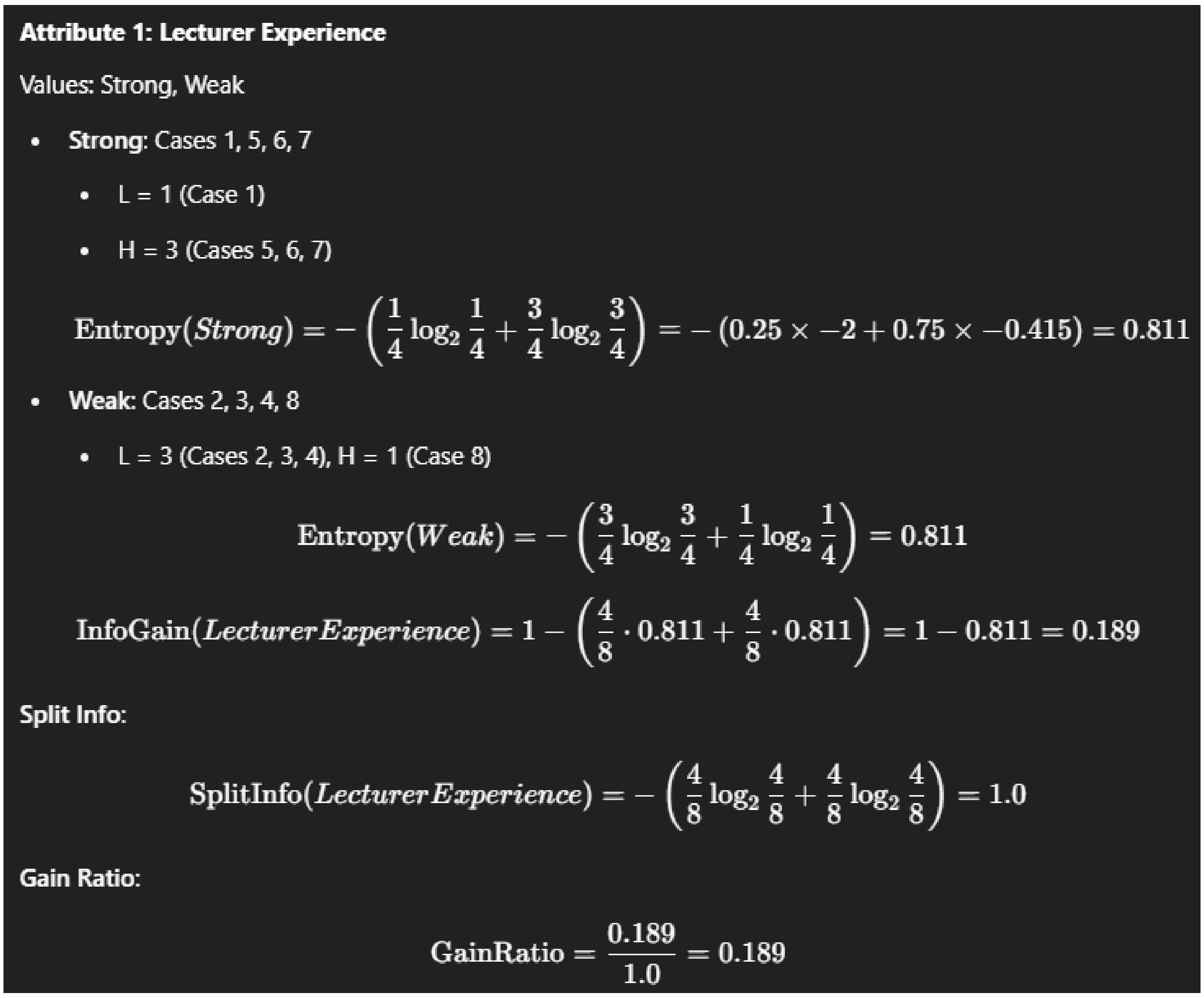
Oversampling: Increases minority class size (e.g., SMOTE).

- ***Used when*: Want to retain full dataset while balancing class distribution.**

Question 3
(3.1) Assume that you are given a set of records as shown in the following table, where the last column contains the target variable. Present the procedure of using Gain Ratio to identify which attribute should be split. You need to show all steps of your calculation in detail.

Case	Lecturer experience	Programming Subject?	Student satisfaction
1	Strong	No	Low
2	Weak	No	Low
3	Weak	Yes	Low
4	Weak	Yes	Low
5	Strong	No	High
6	Strong	No	High
7	Strong	Yes	High
8	Weak	Yes	High

Ans:



Attribute 1: Lecturer Experience

Values: Strong, Weak

- **Strong**: Cases 1, 5, 6, 7
  - L = 1 (Case 1)
  - H = 3 (Cases 5, 6, 7)

$$\text{Entropy}(Strong) = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) = -(0.25 \times -2 + 0.75 \times -0.415) = 0.811$$

- **Weak**: Cases 2, 3, 4, 8
  - L = 3 (Cases 2, 3, 4), H = 1 (Case 8)

$$\text{Entropy}(Weak) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$$

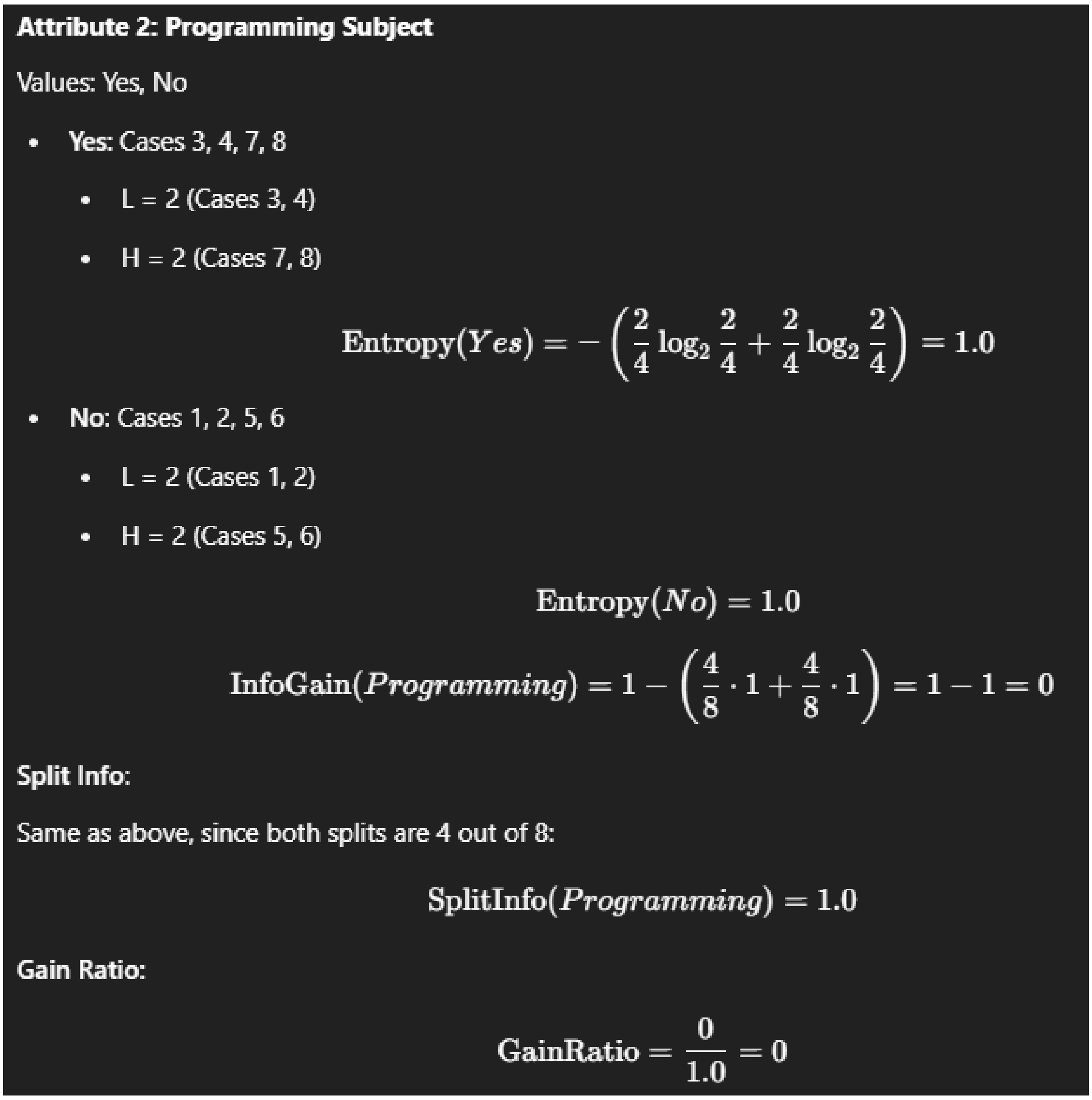
$$\text{InfoGain}(LecturerExperience) = 1 - \left(\frac{4}{8}\cdot 0.811 + \frac{4}{8}\cdot 0.811\right) = 1 - 0.811 = 0.189$$

Split Info:

$$\text{SplitInfo}(LecturerExperience) = -\left(\frac{4}{8}\log_2\frac{4}{8} + \frac{4}{8}\log_2\frac{4}{8}\right) = 1.0$$

Gain Ratio:

$$\text{GainRatio} = \frac{0.189}{1.0} = 0.189$$



Attribute 2: Programming Subject

Values: Yes, No

- **Yes:** Cases 3, 4, 7, 8
  - L = 2 (Cases 3, 4)
  - H = 2 (Cases 7, 8)

$$\text{Entropy}(Yes) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) = 1.0$$

- **No:** Cases 1, 2, 5, 6
  - L = 2 (Cases 1, 2)
  - H = 2 (Cases 5, 6)

$$\text{Entropy}(No) = 1.0$$

$$\text{InfoGain}(Programming) = 1 - \left(\frac{4}{8}\cdot 1 + \frac{4}{8}\cdot 1\right) = 1 - 1 = 0$$

Split Info:

Same as above, since both splits are 4 out of 8:

$$\text{SplitInfo}(Programming) = 1.0$$

Gain Ratio:

$$\text{GainRatio} = \frac{0}{1.0} = 0$$

(3.2) Why an ensemble classifier (such as a Random Forest) can enhance the performance of individual classifiers?

Ans:

Ensemble Classifier

- **Random Forests combine multiple decision trees.**
- **Reduce variance and overfitting.**
- **Each tree learns different patterns via bootstrapping and random feature selection.**
- **Outputs are aggregated (e.g., majority voting), improving accuracy and robustness.**

Question 4

(4.1) Use an example to illustrate the conditional independence assumption, and explain why it is important to the Naïve Bayes classifier.

Ans:

Conditional Independence in Naive Bayes

- **Assumption: Features are independent given the class.**
- ***Example*: If spam emails often contain "free" and "money," Naive Bayes assumes the appearance of "free" is independent of "money," given it is spam.**
- **Importance: Simplifies computation; otherwise, we need joint probabilities for all feature combinations.**

(4.2) Assume that a Bayesian classifier returns the following outcomes for a binary classification problem, which are sorted by decreasing probability values. P (resp., N) refers to a record belonging to a positive (resp., negative) class.

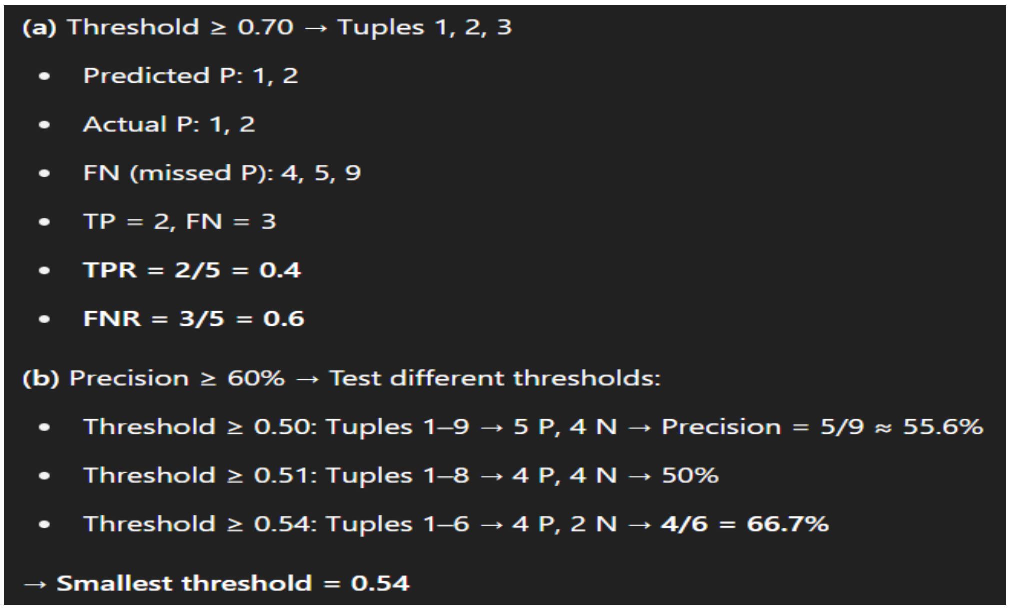
Answer the following questions for the above example, and present all steps of calculation in detail.

Tuple #	Class	Probability
1	P	0.90
2	P	0.80
3	N	0.70
4	P	0.60
5	P	0.55
6	N	0.54
7	N	0.53
8	N	0.51
9	P	0.50
10	N	0.40

a) **What are the true positive (recognition) rate and false negative (recognition) rate if setting the probabilistic classification threshold to ≥ 0.70?**

b) **What is the smallest probabilistic classification threshold such that the precision is at least 60%?**

Ans:



(a) Threshold ≥ 0.70 → Tuples 1, 2, 3

- Predicted P: 1, 2
- Actual P: 1, 2
- FN (missed P): 4, 5, 9
- TP = 2, FN = 3
- **TPR = 2/5 = 0.4**
- **FNR = 3/5 = 0.6**

(b) Precision ≥ 60% → Test different thresholds:

- Threshold ≥ 0.50: Tuples 1–9 → 5 P, 4 N → Precision = 5/9 ≈ 55.6%
- Threshold ≥ 0.51: Tuples 1–8 → 4 P, 4 N → 50%
- Threshold ≥ 0.54: Tuples 1–6 → 4 P, 2 N → **4/6 = 66.7%**

→ **Smallest threshold = 0.54**

Question 5
(5.1) Use an example to explain how the MapReduce model can process the outer join operation.
Ans:

MapReduce Outer Join

- **Map Step: Tag each record by its source (R or S), emit key-value pairs.**
- **Reduce Step: Join all values with same key.**
- **Output includes matching and non-matching keys from both datasets.**

(5.2) Why Apache Spark is suitable for large-scale machine learning? Use an example to support your answer.
Ans:

Apache Spark for ML

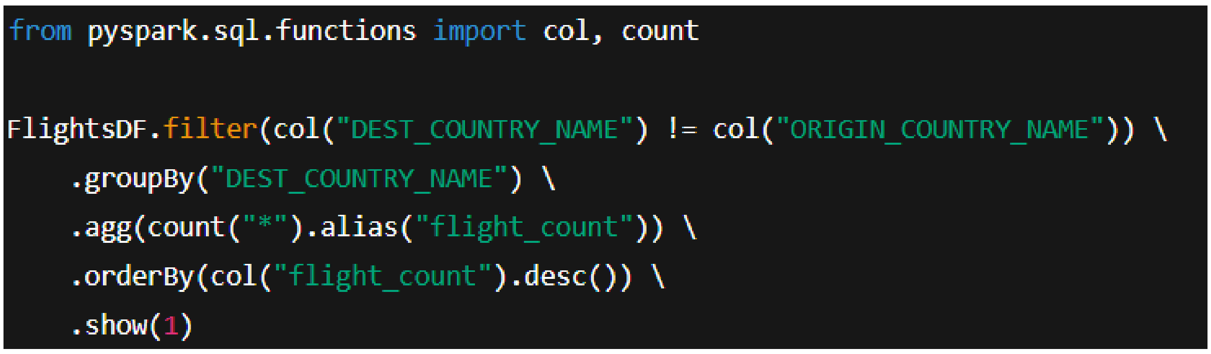
- **In-memory computation.**
- **Distributed ML pipelines (MLlib).**
- ***Example*: Can run logistic regression over millions of records in parallel across clusters.**

(5.3) Assume that a DataFrame named FlightsDF of flight statistics is defined in PySpark, with the following code processed.

```
FlightsDF.printSchema()
Out:
root
  |-- DEST_CITY: string (nullable = true)
  |-- DEST_COUNTRY_NAME: string (nullable = true)
  |-- ORIGIN_CITY: string (nullable = true)
  |-- ORIGIN_COUNTRY_NAME: string (nullable = true)

DF.show(2)
Out:
+---------+-------------+-----------+---------------+
|DEST_CITY|DEST_COUNTRY|ORIGIN_CITY|ORIGIN_COUNTRY|
+---------+-------------+-----------+---------------+
|Sydney   |Australia    |Melbourne  |Australia      |
|Auckland |New Zealand  |Singapore  |Singapore      |
+---------+-------------+-----------+---------------+
only showing top 2 rows
```

Based on FlightsDF, write down the code in PySpark to implement the following operation:
Find the country or countries with most international flights.
(Note. An international flights has different original and destination countries.)
Ans:



```
from pyspark.sql.functions import col, count

FlightsDF.filter(col("DEST_COUNTRY_NAME") != col("ORIGIN_COUNTRY_NAME")) \
    .groupBy("DEST_COUNTRY_NAME") \
    .agg(count("*").alias("flight_count")) \
    .orderBy(col("flight_count").desc()) \
    .show(1)
```

Question 6 (7 marks)
(6.1) Why a classical Perceptron (i.e., a single layer of linear threshold units) is not preferable to use?
Ans:
Limitations of Classical Perceptron

- **Only models linear relationships.**
- **Cannot solve XOR or other non-linearly separable problems.**

(6.2) Implement a feedforward neural network by using the Keras API in TensorFlow for a multi-class classification problem. Assume that the data set has four numerical features and one target variable whose values are 1, 2 and 3. The network has one hidden layer with the sigmoid activation function. Present the Python code.
Ans:

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

model = Sequential()
model.add(Dense(10, input_shape=(4,), activation='sigmoid'))  # hidden layer
model.add(Dense(3, activation='softmax'))  # 3 output classes

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```