Question 1

Given a list named X which contains words (as strings), implement a Python function to compute the word(s) with the highest frequency in X. Write down the Python code.

ANS:

```python
def most_frequent_words(X):
    from collections import Counter
    count = Counter(X)
    max_freq = max(count.values())
    return [word for word, freq in count.items() if freq == max_freq]
```

Question 2

(2.1) Explain the advantages of stratified sampling over standard random sampling.

ANS:

Advantages of stratified sampling:

- **Ensures proportional representation of all classes.**
- **Reduces sampling bias in imbalanced datasets.**
- **Improves model evaluation reliability**

(2.2) Describe three common ways of handling missing values.

ANS:

Three common ways of handling missing values:

1. **Deletion: Remove records or features with missing data (risky for small datasets).**
2. **Imputation: Fill in with mean, median, mode, or a constant.**
3. **Prediction: Use models (e.g., kNN or regression) to predict missing values based on other features.**

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 Gini index to identify which attribute should be split. You need to show all steps of your calculation in detail.

Record ID	Size of class?	Lecturer experience	Programming Subject?	Student satisfaction
1	Large	Strong	No	Low
2	Small	Weak	No	Low
3	Average	Weak	Yes	Low
4	Small	Weak	Yes	Low
5	Average	Strong	No	High
6	Small	Strong	No	High
7	Small	Strong	Yes	High
8	Large	Weak	Yes	High

ANS:

Step 1: Gini of the whole dataset

We have 8 records:

- **Low**: 4
- **High**: 4

$$Gini(S) = 1 - \left(\frac{4}{8}\right)^2 - \left(\frac{4}{8}\right)^2 = 1 - 0.25 - 0.25 = 0.5$$

1. Attribute: Size of Class

Values: Large, Small, Average

- **Large**: ID 1 (Low), ID 8 (High)
  → 1 Low, 1 High → Gini = 1 - 0.5^2 - 0.5^2 = 0.5
- **Small**: ID 2 (Low), 4 (Low), 6 (High), 7 (High)
  → 2 Low, 2 High → Gini = 0.5
- **Average**: ID 3 (Low), 5 (High)
  → 1 Low, 1 High → Gini = 0.5

$$Gini_{split} = \frac{2}{8} \cdot 0.5 + \frac{4}{8} \cdot 0.5 + \frac{2}{8} \cdot 0.5 = 0.5$$

Info gain = 0.5 - 0.5 = 0.0

2. Attribute: Lecturer Experience

Values: Strong, Weak

- **Strong**: ID 1 (Low), 5 (High), 6 (High), 7 (High)
  → 1 Low, 3 High

$$Gini = 1 - (1/4)^2 - (3/4)^2 = 1 - 0.0625 - 0.5625 = 0.375$$

- **Weak**: ID 2 (Low), 3 (Low), 4 (Low), 8 (High)
  → 3 Low, 1 High

$$Gini = 1 - (3/4)^2 - (1/4)^2 = 0.375$$

$$Gini_{split} = \frac{4}{8} \cdot 0.375 + \frac{4}{8} \cdot 0.375 = 0.375$$

Info gain = 0.5 - 0.375 = 0.125

3. Attribute: Programming Subject

Values: Yes, No

- **Yes**: ID 3 (Low), 4 (Low), 7 (High), 8 (High)
  → 2 Low, 2 High → Gini = 0.5
- **No**: ID 1 (Low), 2 (Low), 5 (High), 6 (High)
  → 2 Low, 2 High → Gini = 0.5

$$Gini_{split} = \frac{4}{8} \cdot 0.5 + \frac{4}{8} \cdot 0.5 = 0.5$$

Info gain = 0.5 - 0.5 = 0.0

(3.2) Present the pseudo-code of a decision tree induction algorithm for a data set with categorical and continuous features. You can also support the pseudo-code with explanations.

ANS:

python Copy Edit

```
function build_tree(data):
    if all records have same label or no attributes left:
        return leaf node
    else:
        select best attribute (based on gain/gini)
        split data by attribute values
        for each split:
            build_tree(subset)
    return decision node with branches
```

Question 4

(4.1) Explain in which situations sensitivity and specificity are more important than accuracy as performance metrics of a classifier.

ANS:

Sensitivity/Specificity vs Accuracy:

- **Sensitivity (Recall) is critical when false negatives are dangerous (e.g., disease detection).**
- **Specificity is important when false positives are costly (e.g., spam email classification).**
- **Accuracy can be misleading with imbalanced datasets.**

(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.

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

What is the largest true positive rate when the false positive rate equals 0.4, and what is the smallest false positive rate when the true positive rate equals 0.8? Present the process of your calculation.

ANS:

From the table:

- **Total Positives (P) = 5 → Tuples: 1, 2, 3, 5, 7**
- **Total Negatives (N) = 5 → Tuples: 4, 6, 8, 9, 10**

Threshold ≥	Predicted Positives	TP	FP	TPR (TP/5)	FPR (FP/5)
0.90	1	1	0	0.20	0.00
0.80	1, 2	2	0	0.40	0.00
0.70	1, 2, 3	3	0	0.60	0.00
0.60	1, 2, 3, 4	3	1	0.60	0.20
0.55	1, 2, 3, 4, 5	4	1	0.80	0.20
0.54	1, 2, 3, 4, 5, 6	4	2	0.80	0.40
0.53	1, 2, 3, 4, 5, 6, 7	5	2	1.00	0.40
0.51	1, 2, 3, 4, 5, 6, 7, 8	5	3	1.00	0.60
0.50	1, 2, 3, 4, 5, 6, 7, 8, 9	5	4	1.00	0.80
0.40	1, 2, 3, 4, 5, 6, 7, 8 , 9, 10	5	5	1.00	1.00

1. Largest TPR when FPR = 0.4

- **Two rows satisfy FPR = 0.4:**
  - **At threshold 0.54 → TPR = 0.80**
  - **At threshold 0.53 → TPR = 1.00**
- **Answer: TPR = 1.00 (at threshold = 0.53)**

2. Smallest FPR when TPR = 0.8

- **At threshold 0.55 → TPR = 0.80, FPR = 0.20**
- **Answer: FPR = 0.20 (at threshold = 0.55)**

Question 5

(5.1) Why is Apache Spark suitable for data-parallel computation? What is the bottleneck of model-parallel computation for Apache Spark? Also use an example to support your answer.

ANS:

Apache Spark - Data vs Model Parallelism:

- **Data-parallel: Efficient with large datasets split across cluster nodes.**
- **Model-parallel bottleneck: Coordination and synchronization overhead for model parameters.**
- ***Example*: Training logistic regression on terabytes of data → data-parallelism is effective, but training large neural networks may suffer from sync delay.**

(5.2) 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 *domestic* flights.
(Note. A domestic flight has the same original and destination country.)

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

(6.1) Why a classical Perceptron (i.e., a single layer of linear threshold units) is not preferable to use? Provide your reasons.

ANS:

Classical Perceptron Limitations:

- **Only solves linearly separable problems.**
- **Fails for problems like XOR.**
- **Cannot model complex, non-linear boundaries.**

(6.2) Based on the transfer learning pipeline example in the file named “11 Spark to TensorFlow: Transfer learning pipeline” on this subject’s Moodle site (which can also be directly accessed via the following link), explain the advantages of transfer learning.

https://documents.uow.edu.au/~guoxin/CSCI316/Petastorm-Spark-Converter-TensorFlow.html
In particular, discuss the situations when transfer learning is most applicable.
Note. Your answer must relate to the example; otherwise, no mark will be provided.

ANS:

Transfer Learning from Spark to TensorFlow:

- **Advantages:**
  - **Leverages pre-trained models → reduces training time.**
  - **Effective when labelled data is scarce.**
  - **Useful in similar domains (e.g., using ImageNet model for medical images).**
- **Example Context:**
  - **Data processed in Spark using Petastorm.**
  - **Converted to TensorFlow to fine-tune models.**
  - **Ideal when training deep models on massive data is expensive.**