AI/ML Assessment Test-1 Part II
Section 1 – Descriptive based question

Name: Sumer S Sadawarti Date: 23/01/2026

Section 1 – Descriptive based question

<table>
<tr><td>Answer-1:</td><td>In AI two major types of data exist viz. Numerical and Categorical.
Numerical data type includes numbers of integer or float types. Numerical data type is further classified as Continuous and Discrete. In Continuous data type the data can take continuous values and is mainly of float datatype, e.g. stock price values can take continuous values like 24.36, 33.87 etc. In Discrete data type the data can take only discrete or step values and is of integer type, e.g. number of persons failing the test could be 5, 4, etc.
Categorical data type consists of data that are grouped into definite and defined classes. The labels associated with categorical data type have distinctive markers not necessarily of numerical value. For example, the samples in the dataset could have markers like dog, cat, human, first, second, etc. Categorical data type is further classified as ordinal and cardinal. Ordinal data type has a inherent order in the labels used, e.g. customer experience could be rated as worst, bad, neutral, good, best. Cardinal data type has distinctive markers not having any internal order associated with the makers, e.g. image sample could have labels like cat, dog, human etc.</td></tr>
<tr><td>Answer-2:</td><td>Confusion matrix refers to the matrix of actual Positive and Negative values vs the predicted Positive and Negative values. The matrix is utilised for evaluating the classification models for various metrics like accuracy, precision, recall etc.
The Confusion Matrix is as follows:
<table>
<tr><td></td><td></td><td colspan="2">Predicted</td></tr>
<tr><td></td><td></td><td>Positive</td><td>Negative</td></tr>
<tr><td rowspan="2">Actual</td><td>Positive</td><td>True Positive</td><td>False Negative</td></tr>
<tr><td>Negative</td><td>False Positive</td><td>True Negative</td></tr>
</table>
According to the matrix, a case of True Positive or True Negative is considered when the model predicts accurately the Positive or Negative cases against the Actual Positive or Actual Negative labels respectively.
A case of False Positive arises when the model predicts the label as Positive but the actual label in the data is Negative.</td></tr>
</table>

	Similarly False Negative arises when the model predicts the label as Negative while the actual label is Positive. The Confusion Matrix is used for deriving different evaluation metrics for the classification models like: 1. Accuracy $=\frac{True\ Positive}{True\ Positive+True\ Negative}$ 2. Precision $=\frac{True\ Positive}{True\ Positive+False\ Negative}$
Answer 3:	RNN refers to Recurrent Neural Network, is a deep learning technique used for processing of sequential and textual data. RNN include a recurrent neuron which stores the essence of the previously processed data as a memory. Drawbacks of RNN: 1. Vanishing Gradient Problem: The problem refers to the retention of the data previously encountered. As the RNN is introduced to newer inputs it stores the memory of the previously introduced inputs as well. This works well in short term while for the long sentences RNN could not accurately capture the context and tends to forget the most of its memory. Due to this, RNN could not maintain a strong association over longer sentences. 2. Exploding Gradient Problem: The problem arises when a particular chunk of data is least significant but keeps on repeating in the data which overloads the memory with its essence and other significant data chunks are left out. 3. Semantic problem: RNN does not distinguish between the meaning of the words used in different positions. The sematic value of a particular words differs according to its position in the given sentence. This sematic essence is not captured in the RNN. Types of RNN: 1. LSTM – Long Short Term Memory RNN solves the vanishing gradient problem wherein it decides on the significance of a particular word to be retained over longer term. It includes Input Gate, Forget Gate and Output Gate. 2. GRU – Gradient Relay Unit includes Reset and Update Gates for better capturing of the context and significant words in the sentence.
Answer 4:	Normalization in Natural Language Processing refers to transforming the sentence or a paragraph into a standardised format so that the further processing and absorption by a machine learning or deep learning model becomes simplified and computationally less expensive. Normalization Techniques include: 1. Stemming – Stemming incudes converting the words to their root words while preserving meaning. E.g.

	“running” will be converted to “run” but “ran” will be remain same. 2. Lemmetization – It is similar to Stemming but it converts every form of the word to its root word. In this process the meaning of the word and its association usage is not retained. E.g. “ran” will be converted to “run”. 3. Conversion to lower case – all the alphabets are converted to lower case. E.g. “Apple” is converted to “apple” 4. Removing punctuations – “Sumer’s laptop” will be converted to “Sumer s laptop”

Section 2 – Code

Code 1:	import pandas as pd
	import numpy as np
	
	df = pd.DataFrame([[1,2,3],
	                [4,5,6],
	                [7,8,9]])
	data = df.to_numpy()
	type(data)

Code 2:	import pandas as pd df = pd.read_csv("Salary_dataset.csv") df.dropna(inplace=True) df['Salary'] = df['Salary'].fillna(df['Salary'].mean()) df['YearsExperience'] = df['YearsExperience'].fillna(df['YearsExperience'].median())

Code 3:	from sklearn.preprocessing import OneHotEncoder
	import pandas as pd
	
	df = pd.read_csv("amazon.csv")
	data = df['Sentiment'].to_numpy().reshape(-1,1)
	
	ohe = OneHotEncoder()
	ohe.fit_transform(data)

Code 4:	from sklearn.ensemble import RandomForestClassifier
	from sklearn.model_selection import train_test_split
	from sklearn.metrics import accuracy_score
	import pandas as pd
	
	df = pd.read_csv("Iris.csv")
	
	X = df.iloc[:,:-1].to_numpy()
	y = df.iloc[:,-1].to_numpy()

```
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=50)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy Score: ", acc)
```