

# Python

**Total Marks: 100**

**Each question 10 marks**

**Question 1: -**

Write a program that takes a string as input, and counts the frequency of each word in the string, there might be repeated characters in the string. Your task is to find the highest frequency and returns the length of the highest-frequency word.

**Note** - You have to write at least 2 additional test cases in which your program will run successfully and provide an explanation for the same.

Example input - string = "write write write all the number from from from 1 to 100"

Example output - 5

Explanation - From the given string we can note that the most frequent words are "write" and "from" and the maximum value of both the values is "write" and its corresponding length is 5

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

**Question 2: -**

Consider a string to be *valid* if all characters of the string appear the same number of times. It is also *valid* if he can remove just one character at the index in the string, and the remaining characters will occur the same number of times. Given a string, determine if it is *valid*. If so, return **YES** , otherwise return **NO** .

**Note** - You have to write at least 2 additional test cases in which your program will run successfully and provide an explanation for the same.

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

**Question 3: -**

Write a program, which would download the data from the provided link, and then read the data and convert that into properly structured data and return it in Excel format.

**Note** - Write comments wherever necessary explaining the code written.

**Link** - <https://raw.githubusercontent.com/Biuni/PokemonGO-Pokedex/master/pokedex.json>

**Data Attributes** - **id**: Identification Number - **int** **num**: Number of the

- *Pokémon in the official Pokédex* - **int name**: Pokémon name -
- *string img*: URL to an image of this Pokémon - **string type**:
- *Pokémon type* - **string height**: Pokémon height - float
- **weight**: Pokémon weight - float **candy**: type of candy used to evolve Pokémon or given
- *when transferred* - **string candy\_count**: the amount of candies required to evolve - int
- **egg**: Number of kilometers to travel to hatch the egg - float **spawn\_chance**:
- *Percentage of spawn chance (NEW)* - float **avg\_spawns**: Number of this pokemon on 10.000 spawns **(NEW)** - int
- **spawn\_time**: Spawns most active at the time on this field. Spawn times are the same for all time zones and are expressed in local time. **(NEW)** - “minutes: seconds” **multipliers**: Multiplier of Combat Power (CP) for calculating the CP after evolution See below - list of int
- **weakness**: Types of
- *Pokémon this Pokémon is weak to* - list of strings **next\_evolution**: Number and Name of successive evolutions of Pokémon - list of dict **prev\_evolution**: Number and Name of previous evolutions of Pokémon - - list of dict

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

#### Question 4 -

Write a program to download the data from the link given below and then read the data and convert the into the proper structure and return it as a CSV file.

**Link** - <https://data.nasa.gov/resource/y77d-th95.json>

**Note** - Write code comments wherever needed for code understanding.

**Sample Data** -

```
{
  "name": "Tomakovka",
  "id": "24019",
  "nametype": "Valid",
  "recclass": "LL6",
  "mass": "600",
  "fall": "Fell",
  "year": "1905-01-01T00:00:00.000",
  "reclat": "47.850000",
  "reclong": "34.766670",
  "geolocation": {
    "type": "Point",
    "coordinates": [
      34.76667,
      47.85
    ]
  }
}
```

#### Excerpted Output Data Attributes

- Name of Earth Meteorite - string id - ID of Earth
- Meteorite - int nametype - string recclass - string
- mass - Mass of Earth Meteorite - float year - Year at which Earth
- Meteorite was hit - datetime format reclat - float recclong - float
- point coordinates - list of int

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

#### Question 5 -

Write a program to download the data from the given API link and then extract the following data with proper formatting

**Link** - <http://api.tvmaze.com/singlesearch/shows?q=westworld&embed=episodes>

**Note** - Write proper code comments wherever needed for the code understanding

**Sample Data** -

```
{
  "id": 2326658,
  "url": "https://www.tvmaze.com/episodes/2326658/westworld-4x05-zhuangzi",
  "name": "Zhuangzi",
  "season": 4,
  "number": 5,
  "type": "regular",
  "airdate": "2022-07-24",
  "airtime": "21:00",
  "airstamp": "2022-07-25T01:00:00+00:00",
  "runtime": 60,
  "rating": {
    "average": 7.8
  },
  "image": {
    "medium": "https://static.tvmaze.com/uploads/images/medium_landscape/416/1042460.jpg",
    "original": "https://static.tvmaze.com/uploads/images/original_untouched/416/1042460.jpg"
  },
  "summary": "<p>God is bored.</p>",
  "_links": {
    "self": {
      "href": "https://api.tvmaze.com/episodes/2326658"
    },
    "show": {
      "href": "https://api.tvmaze.com/shows/1371"
    }
  }
}
```

#### Expected Output Data Attributes -

- id - int url - string
- name - string season - int
- number - int
- type - string airdate - date format
- airtime - 12-hour time format
- runtime - float
- average rating - float
- summary - string without html tags
- medium image link - string
- Original image link - string

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

#### Question 6 -

Using the data from **Question 3**, write code to analyze the data and answer the following questions **Note**

1. Draw plots to demonstrate the analysis for the following questions for better visualizations.
2. Write code comments wherever required for code understanding

#### Insights to be drawn -

- Get all Pokemons whose spawn rate is less than 5%
- Get all Pokemons that have less than 4 weaknesses
- Get all Pokemons that have no multipliers at all
- Get all Pokemons that do not have more than 2 evolutions
- Get all Pokemons whose spawn time is less than 300 seconds.

**Note** - spawn time format is "05:32", so assume "minute: second" format and perform the

analysis. ● Get all Pokemon who have more than two types of capabilities

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

#### Question 7 -

Using the data from **Question 4**, write code to analyze the data and answer the following questions **Note**

- 1. Draw plots to demonstrate the analysis for the following questions for better visualizations
2. Write code comments wherever required for code understanding

#### Insights to be drawn -

- Get all the Earth meteorites that fell before the year 2000
- Get all the earth meteorites co-ordinates who fell before the year 1970
- Assuming that the mass of the earth meteorites was in kg, get all those whose mass was more than 10000kg

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

#### Question 8 -

Using the data from **Question 5**, write code to analyze the data and answer the following questions **Note**

- 1. Draw plots to demonstrate the analysis for the following questions and better visualizations
2. Write code comments wherever required for code understanding

#### Insights to be drawn -

- Get all the overall ratings for each season and using plots compare the ratings for all the seasons, like season 1 ratings, season 2, and so on.

- Get all the episode names, whose average rating is more than 8 for every season ●
- Get all the episode names that aired before May 2019
- Get the episode name from each season with the highest and lowest rating
- Get the summary for the most popular ( ratings ) episode in every season

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

### Question 9 -

Write a program to read the data from the following link, perform data analysis and answer the following questions

#### Note -

1. Write code comments wherever required for code understanding

Link - <https://data.wa.gov/api/views/f6w7-q2d2/rows.csv?accessType=DOWNLOAD>

#### Insights to be drawn -

- Get all the cars and their types that do not qualify for clean alternative fuel vehicle ●
- Get all TESLA cars with the model year, and model type made in Bothell City.
- Get all the cars that have an electric range of more than 100, and were made after 2015
- Draw plots to show the distribution between city and electric vehicle type

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

### Question 10 -

Write a program to count the number of verbs, nouns, pronouns, and adjectives in a given particular phrase or paragraph, and return their respective count as a dictionary.

#### Note -

1. Write code comments wherever required for code
2. You have to write at least 2 additional test cases in which your program will run successfully and provide an explanation for the same.

#### Example Output -

```
dic = {  
    "nouns": "count of nouns",  
    "pronouns": "count of pronouns",  
    "verbs": "count of verbs",  
    "adjectives": "count of adjectives"  
}
```

ANSWER. PYTHON- <https://syedbasar14-upgraded-giggle-qj464jj4v44fr4w.github.dev/>

# Statistics

**Total Marks: 120**

**Each question 10 marks**

**Q-1.** A university wants to understand the relationship between the SAT scores of its applicants and their college GPA. They collect data on 500 students, including their SAT scores (out of 1600) and their college GPA (on a 4.0 scale). They find that the correlation coefficient between SAT scores and college GPA is 0.7. What does this correlation coefficient indicate about the relationship between SAT scores and college GPA?

The correlation coefficient of 0.7 indicates a strong positive relationship between SAT scores and college GPA. This means that there is a tendency for students with higher SAT scores to have higher college GPAs, and vice versa. The positive sign of the correlation coefficient indicates that as SAT scores increase, college GPA tends to increase as well.

However, it's important to note that correlation does not imply causation. While the correlation coefficient suggests a strong association between SAT scores and college GPA, it does not provide evidence of a cause-and-effect relationship. Other factors, such as study habits, motivation, and personal circumstances, can also influence college GPA independently of SAT scores.

**Q-2.** Consider a dataset containing the heights (in centimeters) of 1000 individuals. The mean height is 170 cm with a standard deviation of 10 cm. The dataset is approximately normally distributed, and its skewness is approximately zero. Based on this information, answer the following questions:

a. What percentage of individuals in the dataset have heights between 160 cm and 180 cm?

For a height of 160 cm:  $z\text{-score} = (160 - \text{mean}) / \text{standard deviation}$

$$= (160 - 170) / 10 = -1$$

For a height of 180 cm: z-score =  $(180 - \text{mean}) / \text{standard deviation}$

$$= (180 - 170) / 10 = 1$$

Using a standard normal distribution table or calculator, we can find the area to the left of -1 and 1, and then subtract the smaller area from the larger area to find the percentage between -1 and 1.

The area to the left of -1 is approximately 0.1587, and the area to the left of 1 is approximately 0.8413.

So, the area between -1 and 1 is  $0.8413 - 0.1587 = 0.6826$ . Therefore, approximately 68.26% of individuals in the dataset have heights between 160 cm and 180 cm.

b. If we randomly select 100 individuals from the dataset, what is the probability that their average height is greater than 175 cm?

The mean of the sampling distribution of the mean is the same as the population mean, which is 170 cm. The standard deviation of the sampling distribution of the mean, also known as the standard error of the mean, is equal to the population standard deviation divided by the square root of the sample size.

Standard error of the mean =  $\text{standard deviation} / \sqrt{\text{sample size}}$

$$= 10 / \sqrt{100} = 10 / 10 = 1$$

To calculate the probability, we need to find the z-score corresponding to a height of 175 cm in the sampling distribution.

The z-score formula is:  $\text{z-score} = (x - \text{mean}) / \text{standard error z-score}$

$$= (175 - 170) / 1 = 5 / 1 = 5$$

Using a standard normal distribution table or calculator, we can find the area to the right of 5, which represents the probability that the average height is greater than 175 cm. The area to the right of 5 is extremely close to 0.

So, the probability that the average height of 100 randomly selected individuals is greater than 175 cm is approximately 0.

c. Assuming the dataset follows a normal distribution, what is the z-score corresponding to a height of 185 cm?

$$\text{z-score} = (x - \text{mean}) / \text{standard deviation} = (185 - 170) / 10 = 15 / 10 = 1.5$$

Therefore, the z-score corresponding to a height of 185 cm is 1.5.

d. We know that 5% of the dataset has heights below a certain value. What is the approximate height corresponding to this threshold?

$$\text{z-score} = (x - \text{mean}) / \text{standard deviation}$$

Since we are looking for the value below which 5% of the data falls, we need to find the z-score that corresponds to an area of 0.05 to the left of the threshold. Using a standard normal distribution table or calculator, the z-score that corresponds to an area of 0.05 to the left is approximately -1.645.

Now we can use the z-score formula to find the height:  $-1.645 = (x - 170) / 10$

$$\begin{aligned} \text{Solving for } x: & -1.645 * 10 = x - 170 \\ & -16.45 = x - 170 \\ & x = 170 - 16.45 \\ & x \approx 153.55 \end{aligned}$$

e. Calculate the coefficient of variation (CV) for the dataset.

$$\text{CV} = (\text{standard deviation} / \text{mean}) * 100$$

In this case, the mean height is 170 cm and the standard deviation is 10 cm. Plugging these values into the formula, we get:  $\text{CV} = (10 / 170) * 100$   $\text{CV} \approx 5.88$



f. Calculate the skewness of the dataset and interpret the result.

Skewness =  $(3 * (\text{mean} - \text{median})) / \text{standard deviation}$

Given that the dataset is approximately normally distributed and has a skewness of approximately zero, we can infer that the data is symmetrically distributed around the mean.

**Q-3.** Consider the 'Blood Pressure Before' and 'Blood Pressure After' columns from the data and calculate the following

[https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share\\_](https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share_)

a. Measure the dispersion in both and interpret the results.

b. Calculate mean and 5% confidence interval and plot it in a graph

c. Calculate the Mean absolute deviation and Standard deviation and interpret the results.

d. Calculate the correlation coefficient and check the significance of it at 1% level of significance.

**Q-4.** A group of 20 friends decides to play a game in which they each write a number between 1 and 20 on a slip of paper and put it into a hat. They then draw one slip of paper at random. What is the probability that the number on the slip of paper is a perfect square (i.e., 1, 4, 9, or 16)?

**Favorable Outcomes:** The favorable outcomes are the perfect squares: 1, 4, 9, and 16. There are four perfect squares in the range of numbers between 1 and 20.

**Total Possible Outcomes:** The total possible outcomes are the numbers between 1 and 20. Since there are 20 friends, each writing a number between 1 and 20, the total possible outcomes are also 20.

**Calculate the Probability:** Probability = Number of Favorable Outcomes / Total Number of Possible Outcomes

In this case, the probability is :

Probability = 4 (perfect squares) / 20 (total numbers) Probability =  $4/20$  Probability =  $1/5$

Therefore, the probability that the number drawn from the hat is a perfect square (1, 4, 9, or 16) is  $1/5$  or 0.2 (20%).

**Q-5.** A certain city has two taxi companies: Company A has 80% of the taxis and Company B has 20% of the taxis. Company A's taxis have a 95% success rate for picking up passengers on time, while Company B's taxis have a 90% success rate. If a randomly selected taxi is late, what is the probability that it belongs to Company A?

Let's define the following probabilities:

$P(A)$  = Probability that the taxi belongs to Company A = 0.8 (given that Company A has 80% of the taxis)

$P(B)$  = Probability that the taxi belongs to Company B = 0.2 (given that Company B has 20% of the taxis)

$P(L|A)$  = Probability that the taxi is late given that it belongs to Company A =  $1 - 0.95 = 0.05$  (given a 95% success rate)

$P(L|B)$  = Probability that the taxi is late given that it belongs to Company B =  $1 - 0.90 = 0.10$  (given a 90% success rate)

We want to calculate  $P(A|L)$ , which is the probability that the taxi belongs to Company A given that it is late.

According to Bayes' theorem:  $P(A|L) = (P(L|A) * P(A)) / (P(L|A) * P(A) + P(L|B) * P(B))$

Plugging in the values:  $P(A|L) = (0.05 * 0.8) / (0.05 * 0.8 + 0.10 * 0.2)$   $P(A|L) = 0.04 / (0.04 + 0.02)$   $P(A|L) = 0.04 / 0.06$   $P(A|L) = 0.67$

Therefore, the probability that a randomly selected late taxi belongs to Company A is 0.67 or 67%.

**Q-6.** A pharmaceutical company is developing a drug that is supposed to reduce blood pressure. They conduct a clinical trial with 100 patients and record their blood pressure before and after taking the drug. The company wants to know if the change in blood pressure follows a normal distribution.

<https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share>

**Q-7.** The equations of two lines of regression, obtained in a correlation analysis between variables X and Y are as follows:

and  $2x + 3y - 8 = 0$   $2x + y - 5 = 0$  The variance of  $x = 4$  Find the

- Variance of Y
- Coefficient of determination of C and Y
- Standard error of estimate of X on Y and of Y on X.

The given equation of the lines of regression are  $x + 2y - 5 = 0$ .....

(i) and  $2x + 3y - 8 = 0$ .....

(ii) Rewriting the equations (i) and (ii), we have From equation (i)

$$y = 2.5 - 0.5x$$

$$y = -0.5x + 2.5 \text{ *regression line of y on x) } b_{yx} = r \frac{\sigma_y}{\sigma_x} = -0.5 \text{....(iii)}$$

From equation (ii),  $x = 2 - 3y + 28$   $x = -1.5y + 4$  (regression line of x on y)  $b_{xy} = r \sigma_y / \sigma_x$   
 $\therefore r^2 = b_{yx} \times b_{xy} = (-0.5) \times (-1.5) = 0.75$   
 $\therefore r = 0.75 = \pm 0.866$   
 But  $b_{xy}$  and  $b_{yx}$  being both -ve therefore, r is also -ve.  
 Correlation coefficient (r) = -0.866  
 Variance of x i.e.,  $\sigma_x^2 = 12 \therefore \sigma_x = \sqrt{12}$   
 From equation (iii)  $r \sigma_x / \sigma_y = -0.5$   $-0.866 \cdot \sqrt{12} / \sigma_y = -0.5$   $\sigma_y = 0.866 \cdot 0.5 \times \sqrt{12} = 2 \therefore$   
 Variance of y i.e.,  $\sigma_y^2 = 4$

**Q-8.** The anxiety levels of 10 participants were measured before and after a new therapy. The scores are not normally distributed. Use the Wilcoxon signed-rank test to test whether the therapy had a significant effect on anxiety levels. The data is given below: Participant Before therapy After therapy Difference

Participant	Before therapy	After therapy	Difference
1	10	7	-3
2	8	6	-2
3	12	10	-2
4	15	12	-3
5	6	5	-1
6	9	8	-1
7	11	9	-2
8	7	6	-1
9	14	12	-2
10	10	8	-2

**Q-9.** Given the score of students in multiple exams

Name	Exam 1	Exam 2	Final Exam
Karan	85	90	92
Deepa	70	80	85
Karthik	90	85	88
Chandan	75	70	75
Jeevan	95	92	96

Test the hypothesis that the mean scores of all the students are the same. If not, name the student with the highest score.

**Q-10.** A factory produces light bulbs, and the probability of a bulb being defective is 0.05. The factory produces a large batch of 500 light bulbs.

- What is the probability that exactly 20 bulbs are defective?
- What is the probability that at least 10 bulbs are defective?
- What is the probability that at max 15 bulbs are defective?
- On average, how many defective bulbs would you expect in a batch of 500?

To solve these probability questions, we can use the binomial distribution since we

have a fixed number of trials (500 light bulbs) and each trial has two possible outcomes

(defective or non-defective) with a fixed probability of success (0.05 for defective bulbs). The

binomial distribution probability mass function is given by:

$$P(X=k) = C(n, k) * p^k * (1-p)^{(n-k)}, \text{ where: } P(X=k) \text{ is the probability of having exactly}$$

k successes (defective bulbs) n is the total number of trials (number of light bulbs) k is the

number of successful trials (number of defective bulbs) p is the probability of success in a

single trial (probability of a bulb being defective) C(n, k) is the binomial coefficient, which

represents the number of ways to choose k successes from n trials and is calculated as C(n,

$$k) = n! / (k! * (n-k)!) \text{ a.}$$

Probability of exactly 20 defective bulbs:

$$P(X=20) = C(500, 20) * 0.05^{20} * (1-0.05)^{(500-20)}$$

b. Probability of at least 10 defective bulbs:  $P(X \geq 10) = P(X=10) + P(X=11) + \dots +$

$$P(X=500)$$

c. Probability of at most 15 defective bulbs:  $P(X \leq 15) = P(X=0) + P(X=1) + \dots +$

$$P(X=15)$$

d. Average number of defective bulbs: The expected value of a binomial distribution is

given by  $E(X) = n * p$ . Let's calculate these probabilities and the expected value:

a. Probability of exactly 20 defective bulbs:  $P(X=20) = C(500, 20) * 0.05^{20} *$

$$(1-0.05)^{(500-20)}$$

b. Probability of at least 10 defective bulbs:  $P(X \geq 10) = P(X=10) + P(X=11) + \dots +$

$$P(X=500)$$

c. Probability of at most 15 defective bulbs:  $P(X \leq 15) = P(X=0) + P(X=1) + \dots +$

$$P(X=15)$$

d. Average number of defective bulbs:  $E(X) = 500 * 0.05$  Let's calculate these

probabilities and the expected value:

a. Probability of exactly 20 defective bulbs:  $P(X=20) = C(500, 20) * 0.05^{20} *$

$(1-0.05)^{(500-20)}$  To calculate the binomial coefficient  $C(n, k)$ , we use the formula:  $C(n, k) = n!$

$/ (k! * (n-k)!)$  Using the given formula and the values of  $n=500$  and  $k=20$ , we can calculate the

probability of exactly 20 defective bulbs:  $P(X=20) = C(500, 20) * 0.05^{20} * (1-0.05)^{(500-20)}$

b. Probability of at least 10 defective bulbs: To calculate the probability of at least 10 defective bulbs, we need to sum up the probabilities of having 10, 11, 12, ..., up to 500 defective bulbs.

This can be done using the complement rule, where the probability of at least 10 defective bulbs is equal to 1 minus the probability of having less than 10 defective bulbs:  $P(X \geq 10) = 1$

-  $P(X < 10)$  To calculate  $P(X < 10)$ , we sum up the probabilities of having 0, 1, 2, ..., up to 9 defective bulbs:  $P(X < 10) = P(X=0) + P(X=1) + P(X=2) + \dots + P(X=9)$  c. Probability of at most

15 defective bulbs: To calculate the probability of at most 15 defective bulbs, we need to sum up the probabilities of having 0, 1, 2, ..., up to 15 defective bulbs:  $P(X \leq 15) = P(X=0) +$

$P(X=1) + P(X=2) + \dots + P(X=15)$  d. Average number of defective bulbs: The expected value of

a binomial distribution is given by the formula:  $E(X) = n * p$  Using the given values of  $n=500$

and  $p=0.05$ , we can calculate the average number of defective bulbs:  $E(X) = 500 * 0.05$  Now,

let's calculate these probabilities and the expected value using the formulas and values

provided.

**Q-11.** Given the data of a feature contributing to different classes

<https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share>

- Check whether the distribution of all the classes are the same or not.
- Check for the equality of variance/
- Which amount LDA and QDA would perform better on this data for classification and why.
- Check the equality of mean for between all the classes.

**Q-12.** A pharmaceutical company develops a new drug and wants to compare its effectiveness against a standard drug for treating a particular condition. They conduct a study with two groups: Group A receives the new drug, and Group B receives the standard drug. The company measures the improvement in a specific symptom for both groups after a 4-week treatment period.

- a. The company collects data from 30 patients in each group and calculates the mean improvement score and the standard deviation of improvement for each group. The mean improvement score for Group A is 2.5 with a standard deviation of 0.8, while the mean improvement score for Group B is 2.2 with a standard deviation of 0.6. Conduct a t-test to determine if there is a significant difference in the mean improvement scores between the two groups. Use a significance level of 0.05.
- b. Based on the t-test results, state whether the null hypothesis should be rejected or not. Provide a conclusion in the context of the study.

**ANS:** To determine if there is a significant difference in the mean improvement scores between Group A and Group B, we can conduct an independent sample t-test.

The t-test compares the means of two independent groups and assesses whether the observed difference between the means is statistically significant.

Let's perform the t-test using the provided data:

a. Hypotheses: The null hypothesis ( $H_0$ ): There is no significant difference in the mean improvement scores between Group A and Group B. The alternative hypothesis ( $H_a$ ): There is a significant difference in the mean improvement scores between Group A and Group B.

b. Significance level: The significance level ( $\alpha$ ) is given as 0.05. This represents the probability of rejecting the null hypothesis when it is true.

c. Calculation: We will use the formula for an independent samples t-test:  
$$t = \frac{(\text{mean\_A} - \text{mean\_B})}{\sqrt{(\text{squared\_deviation\_A}/n\_A) + (\text{squared\_deviation\_B}/n\_B)}}$$
Where: mean\_A and mean\_B are the means of Group A and Group B, respectively. squared\_deviation\_A and squared\_deviation\_B are the squared standard deviations of Group A and Group B, respectively. n\_A and n\_B are the sample sizes of Group A and Group B, respectively. Given the following values: mean\_A = 2.5, standard deviation\_A = 0.8, n\_A = 30 mean\_B = 2.2, standard deviation\_B = 0.6, n\_B = 30 Let's calculate the t-statistic:  $t = (2.5 - 2.2) / \sqrt{(0.8^2/30) + (0.6^2/30)}$

d. Degrees of freedom: The degrees of freedom for an independent samples t-test is calculated as:  $df = n\_A + n\_B - 2$  Substituting the values:  $df = 30 + 30 - 2$

e. Critical value and p-value: We will compare the calculated t-statistic with the critical value from the t-distribution table at the given significance level ( $\alpha$ ) and degrees of freedom (df). We will also calculate the p-value associated with the t-statistic.

f. Conclusion: If the calculated t-statistic is greater than the critical value or if the p-value is less than the significance level ( $\alpha$ ), we reject the null hypothesis. Otherwise, we fail to reject the null hypothesis. Let's perform the calculations and interpret the results.

# Machine learning

**Total Marks: 210**

**Each question 15 marks**

## INTERMEDIATE QUESTIONS :

**Q-1.** Imagine you have a dataset where you have different Instagram features like `username` , `Caption` , `Hashtag` , `Followers` , `Time_Since_posted` , and `likes` , now your task is to predict the number of `likes` and `Time Since posted` and the rest of the features are your input features. Now you have to build a model which can predict the number of likes and Time Since posted.

[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-2.** Imagine you have a dataset where you have different features like `Age` , `Gender` , `Height` , `Weight` , `BMI` , and `Blood Pressure` and you have to classify the people into different classes like `Normal` , `Overweight` , `Obesity` , `Underweight` , and `Extreme Obesity` by using any 4 different classification algorithms. Now you have to build a model which can classify people into different classes.

[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-3.** Imagine you have a dataset where you have different categories of data, Now you need to find the most similar data to the given data by using any 4 different similarity algorithms. Now you have to build a model which can find the most similar data to the given data.

[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-4.** Imagine you working as a sale manager now you need to predict the `Revenue` and whether that particular revenue is on the `weekend` or not and find the `Informational_Duration` using the Ensemble learning algorithm

[Dataset](#) This is the Dataset You can use this dataset for this question.



**Q-5.** Uber is a taxi service provider as we know, we need to predict the high booking area using an Unsupervised algorithm and price for the location using a supervised algorithm and use some map function to display the data [Dataset](#)  
This is the Dataset You can use this dataset for this question.

**Q-6.** Imagine you have a dataset where you have predicted loan Eligibility using any 4 different classification algorithms. Now you have to build a model which can predict loan Eligibility and you need to find the accuracy of the model and built-in docker and use some library to display that in frontend  
[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-7.** Imagine you have a dataset where you need to predict the Genres of Music using

an Unsupervised algorithm and you need to find the accuracy of the model, built-in docker, and use some library to display that in frontend  
[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-8.** Quora question pair similarity, you need to find the Similarity between two questions by mapping the words in the questions using TF-IDF, and using a supervised Algorithm you need to find the similarity between the questions. [Dataset](#) This is the Dataset You can use this dataset for this question.

**Q-9.** A cyber security agent wants to check the Microsoft Malware so need he came to you as a Machine learning Engineering with Data, You need to find the Malware using a supervised algorithm and you need to find the accuracy of the model. [Dataset](#) This is the Dataset You can use this dataset for this question.

1. An Ad- Agency analyzed a dataset of online ads and used a machine learning model to predict whether a user would click on an ad or not.  
[Dataset](#) This is the Dataset You can use this dataset for this

question. **Advance QUESTIONS :**

**Q-1.** A Social Media Influencer collected data on Facebook friend requests and used a supervised algorithm to predict whether a user would accept a friend request or

not. [Dataset](#) This is the Dataset You can use this dataset for this question. Note : Use only Dask and Use MLflow

**Q-2.** A chemist had two chemical flasks labeled 0 and 1 which consist of two different chemicals. He extracted 3 features from these chemicals in order to distinguish between them, you provided the results derived by the chemicals and your task is to create a model that will label chemical 0 or 1 given its three features and built-in docker and use some library to display that in frontend. Note : Use only pyspark

[Dataset](#) This is the Dataset You can use this dataset for this question.

**Q- 3.** A company wants to predict the sales of its product based on the money spent on different platforms for marketing. They want you to figure out how they can spend money on marketing in the future in such a way that they can increase their profit as much as possible built-in docker and use some library to display that in frontend. [Dataset](#) This is the Dataset You can use this dataset for this question. Note: Use only Dask

**Q-4.** Take any 3 questions and deploy them to AWS using GitHub Actions and show a demo link

**Q-5.** Take any 3 questions and deploy them to AWS using Circle-CI and show a demo link

# Deep Learning

**Total Marks: 100**

**Each question 20 marks**

## Question 1 -

Implement 3 different CNN architectures with a comparison table for the MNIST dataset using the Tensorflow library.

### Note -

1. The model parameters for each architecture should not be more than 8000 parameters

2. Code comments should be given for proper code understanding.
3. The minimum accuracy for each accuracy should be at least 96%

### **Question 2 -**

Implement 5 different CNN architectures with a comparison table for CIFAR 10 dataset using the PyTorch library

#### **Note -**

1. The model parameters for each architecture should not be more than 10000 parameters
- 2 Code comments should be given for proper code understanding

Certainly! I'll provide you with a high-level overview of how you can implement 5 different CNN architectures for the CIFAR-10 dataset using the PyTorch library. I'll also include some code comments to help with understanding. Please keep in mind that I won't be able to provide a complete working code implementation, but I'll guide you through the process. Here's an outline of the steps you can follow:

**Import the necessary libraries:** Start by importing the required libraries, including PyTorch and torchvision.

**Load and preprocess the CIFAR-10 dataset:** Use torchvision to load the CIFAR-10 dataset. Preprocess the data by normalizing pixel values and converting labels to appropriate format.

**Define the first CNN architecture:** Begin by defining a CNN model with the desired architecture. Experiment with different combinations of convolutional layers, pooling layers, and fully connected layers. Ensure that the total number of model parameters does not exceed 10,000.

**Define the training loop:** Create a training loop that iterates over the dataset in batches. Include steps for forward propagation, backward propagation, loss calculation, and parameter updates using an optimizer like stochastic gradient descent (SGD) or Adam.

**Train the model:** Run the training loop for a specified number of epochs to train the model. Monitor the training loss and accuracy during each epoch.

**Evaluate the model:** After training, evaluate the model's performance on the test dataset. Calculate metrics such as accuracy, precision, recall, and F1 score.

**Repeat steps 3 to 6 for four more CNN architectures:** Implement four additional CNN architectures with different designs, making sure to keep the total model parameters within the specified limit. Repeat the training and evaluation process for each architecture.

**Create a comparison table:** Finally, create a comparison table that includes the architecture details (number of layers, filters, etc.), model parameters, and the evaluation metrics achieved for each architecture. This table will allow you to compare the performance of the different models.

### **Question 3 -**

Train a Pure CNN with less than 10000 trainable parameters using the MNIST Dataset having minimum validation accuracy of 99.40%

#### **Note -**

1. Code comments should be given for proper code understanding.
2. Implement in both PyTorch and Tensorflow respectively

**PyTorch Implementation:** Import the necessary libraries: Import PyTorch and torchvision. Load and preprocess the MNIST dataset: Use torchvision to load the MNIST dataset. Preprocess the data by normalizing pixel values and converting labels to the appropriate format. Define the CNN architecture: Design a CNN model with the desired architecture. Keep the total number of trainable parameters under 10,000. Experiment with different combinations of convolutional layers, pooling layers, and fully connected layers.

Define the training loop: Create a training loop that iterates over the dataset in batches. Include steps for forward propagation, backward propagation, loss calculation, and parameter updates using an optimizer like stochastic gradient descent (SGD) or Adam. Train the model: Run the training loop for a sufficient number of epochs to train the model. Monitor the training loss and accuracy during each epoch. Early stopping can be implemented if the validation accuracy reaches the desired threshold. Evaluate the model: After training, evaluate the model's performance on the validation dataset. Calculate the accuracy and compare it to the minimum required accuracy of 99.40%. TensorFlow Implementation: Import the necessary libraries: Import TensorFlow and other required libraries. Load and preprocess the MNIST dataset: Use TensorFlow's built-in dataset module to load the MNIST dataset. Preprocess the data by normalizing pixel values and converting labels to the appropriate format. Define the CNN architecture: Design a CNN model with the desired architecture. Ensure that the total number of trainable parameters is under 10,000. Experiment with different combinations of convolutional layers, pooling layers, and fully connected layers. Compile and train the model: Compile the model by specifying the optimizer, loss function, and evaluation metric. Train the model using the training dataset, and validate it on the validation dataset. Monitor the accuracy during training and consider implementing early stopping if the validation accuracy reaches the desired threshold. Evaluate the model: Once training is complete, evaluate the model on the validation dataset and calculate the accuracy. Compare it to the minimum required accuracy of 99.40%.

#### **Question 4 -**

Design an end-to-end solution with diagrams for object detection use cases leveraging AWS cloud services and open-source tech

##### **Note -**

1. You need to use both AWS cloud services and open-source tech to design the entire solution
2. The pipeline should consist of a data pipeline, ml pipeline, deployment pipeline, and inference pipeline.
3. In the data pipeline, you would be designing how to get the data from external or existing sources and tech used for the same
4. In the ml pipeline, you would be designing how to train the model, and what all algorithms, techniques, etc. would you be using. Again, tech used for the same
5. Since this is a deep learning project, the use of GPUs, and how effectively are you using them to optimize for cost and training time should also be taken into consideration.
6. In the deployment pipeline, you would be designing how effectively and efficiently you are deploying the model in the cloud,
7. In the inference pipeline, consider the cost of inference and its optimization

related to computing resources and handling external traffic

8. You can use any tool to design the architecture

9. Do mention the pros and cons of your architecture and how much further it can be optimized and its tradeoffs.

10. Do include a retraining approach as well.

11. Try to include managed AWS resources for deep learning like AWS Textract, AWS Sagemaker, etc., and not just general-purpose compute resources like S3, EC2, etc. Try to mix the best of both services

Designing an end-to-end solution for object detection use cases leveraging AWS cloud services and open-source tech involves several components and considerations. Here's an overview of the architecture with diagrams and explanations for each pipeline:

**Data Pipeline:** In the data pipeline, the goal is to acquire and preprocess the data for object detection. Here's a high-level architecture diagram for the data pipeline: `rust Copy code` External/Existing Data Sources -> Data Ingestion -> Data Preprocessing -> Storage

**External/Existing Data Sources:** These can include various sources like image databases, video feeds, or IoT devices that capture the data for object detection.

**Data Ingestion:** Use AWS services like AWS S3 or AWS IoT to ingest the data from the sources into the pipeline. Open-source tools like Apache Kafka or RabbitMQ can also be used for streaming data ingestion.

**Data Preprocessing:** Utilize open-source tech such as OpenCV or TensorFlow Transform for data preprocessing tasks like resizing, augmentation, normalization, or feature extraction.

**Storage:** Store the preprocessed data in a scalable and durable storage service like AWS S3 for easy access during training and deployment.

**ML Pipeline:** The ML pipeline involves training the object detection model. Here's a diagram showcasing the ML pipeline: `rust Copy code` Data Storage -> Model Training -> Model Evaluation -> Model Storage

**Data Storage:** Retrieve the preprocessed data from the storage service (e.g., AWS S3) and feed it into the training process.

**Model Training:** Leverage AWS SageMaker, which provides a managed environment for training ML models. Utilize popular open-source object detection frameworks like TensorFlow or PyTorch along with GPU instances to accelerate training. Techniques like transfer learning and state-of-the-art algorithms such as YOLO or SSD can be employed.

**Model Evaluation:** Evaluate the trained model's performance using metrics like precision, recall, and mean Average Precision (mAP) on validation or test datasets.

**Model Storage:** Store the trained model in a model registry or version control system like AWS Model Registry, Amazon ECR, or open-source solutions like MLflow.

**Deployment Pipeline:** The deployment pipeline focuses on deploying the trained model for inference. Here's a diagram illustrating the deployment pipeline: `rust Copy code` Model Storage -> Model Deployment -> Scaling and Autoscaling -> Endpoint Creation

**Model Storage:** Retrieve the trained model from the model registry or version control system.

**Model Deployment:** Utilize AWS SageMaker's hosting functionality or open-source frameworks like TensorFlow Serving or FastAPI to deploy the model as a scalable and managed API endpoint.

**Scaling and Autoscaling:** Configure

autoscaling policies to handle varying traffic loads and scale the deployed model based on demand. Use AWS Application Load Balancer or Kubernetes for load balancing and scaling. Endpoint Creation: Create an API endpoint that can receive image or video inputs and return object detection predictions. Use AWS API Gateway or open-source solutions like Flask or FastAPI to expose the endpoint. Inference Pipeline: The inference pipeline focuses on efficiently performing object detection on new data. Here's a diagram showcasing the inference pipeline: rust Copy code External Traffic -> Load Balancer -> Inference Server(s) -> GPU Utilization -> Predictions External Traffic: Incoming requests for object detection from external sources or client applications. Load Balancer: Utilize AWS Elastic Load Balancer or open-source tools like Nginx or HAProxy for load balancing and distributing requests among multiple inference servers. Inference Server(s): Deploy multiple instances of the deployed model to handle inference requests. Leverage GPU instances for accelerated inference, utilizing AWS EC2 instances with GPU support or AWS Inferentia. GPU Utilization: Optimize GPU utilization by batching multiple inference requests and parallelizing computations using frameworks like TensorFlow or PyTorch. Predictions: Return the object detection predictions to the requesting client application or system. Pros and Cons, Optimization, and Trade-offs: Pros: Leveraging managed AWS services like AWS SageMaker, AWS S3, and AWS API Gateway reduces infrastructure management complexity. Open-source tools provide flexibility, and using GPUs accelerates training and inference. Scaling and autoscaling capabilities ensure high availability and handle variable traffic loads effectively. Cons: Managed services may have cost implications compared to self-managed infrastructure. AWS-specific solutions may lead to vendor lock-in, reducing portability. GPU usage costs should be considered and balanced with training time improvements. Optimization and Trade-offs: Further optimization can be achieved by optimizing hyperparameters, model architecture, or using more advanced techniques like quantization or model compression for inference efficiency. Trade-offs include the need for continuous monitoring and resource allocation to manage costs effectively. Retraining Approach: To incorporate retraining, a feedback loop can be established to periodically collect new labeled data, retrain the model using the updated dataset, and replace the previously deployed model with the updated one. This helps the model adapt to changing scenarios and improve its performance over time. Remember to consider security measures, data privacy, and compliance requirements when designing the solution. These are high-level guidelines for designing an end-to-end solution for object detection leveraging AWS cloud services and open-source tech. Actual implementation details and specific services used may vary based on project requirements and constraints.

## Question 5 -

In **Question 4**, you have designed the architecture for an object detection use case leveraging AWS Cloud, similarly, here you will be designing for Document

Classification use case leveraging Azure Cloud services.

**Note -**

1. Most of the points are the same as in **Question 4**, just cloud services will change

**Data Pipeline:** In the data pipeline, the goal is to acquire and preprocess the data for document classification. Here's a high-level architecture diagram for the data pipeline using Azure services: `rust Copy code`  
**External/Existing Data Sources -> Data Ingestion -> Data Preprocessing -> Storage**  
**External/Existing Data Sources:** These can include various sources like document repositories, file shares, or cloud storage where the documents are stored. **Data Ingestion:** Use Azure services like Azure Data Factory or Event Grid to ingest the data from the sources into the pipeline. These services can handle data ingestion from various sources and provide data integration capabilities. **Data Preprocessing:** Utilize open-source libraries like NLTK or SpaCy along with Azure Databricks for data preprocessing tasks like text cleaning, tokenization, and feature extraction. **Storage:** Store the preprocessed data in Azure Blob Storage or Azure Data Lake Storage for easy access during training and deployment. **ML Pipeline:** The ML pipeline involves training the document classification model. Here's a diagram showcasing the ML pipeline using Azure services: `rust Copy code`  
**Data Storage -> Model Training -> Model Evaluation -> Model Storage**  
**Data Storage:** Retrieve the preprocessed data from Azure Blob Storage or Azure Data Lake Storage and feed it into the training process. **Model Training:** Utilize Azure Machine Learning service for training the document classification model. Azure Machine Learning provides a managed environment for training ML models, and you can leverage various open-source libraries like Scikit-learn or TensorFlow for model training. **Model Evaluation:** Evaluate the trained model's performance using appropriate evaluation metrics like accuracy, precision, recall, or F1 score on validation or test datasets. Azure Machine Learning provides capabilities for model evaluation and tracking. **Model Storage:** Store the trained model in Azure Machine Learning Model Registry or Azure Blob Storage for easy access during deployment. **Deployment Pipeline:** The deployment pipeline focuses on deploying the trained model for inference. Here's a diagram illustrating the deployment pipeline using Azure services: `rust Copy code`  
**Model Storage -> Model Deployment -> Scaling and Autoscaling -> Endpoint Creation**  
**Model Storage:** Retrieve the trained model from Azure Machine Learning Model Registry or Azure Blob Storage. **Model Deployment:** Utilize Azure Container Instances or Azure Kubernetes Service (AKS) to deploy the model as a scalable and managed API endpoint. Azure Container Registry can be used to store container images. **Scaling and Autoscaling:** Configure autoscaling policies in Azure Kubernetes Service (AKS) to handle varying traffic loads and scale the deployed model based on demand. Azure Traffic Manager can be used for load balancing and routing requests to different instances. **Endpoint Creation:** Create an API endpoint using Azure API Management or Azure Functions that can receive document inputs and return classification predictions. **Inference Pipeline:** The inference pipeline focuses on

performing document classification on new data efficiently. Here's a diagram showcasing the inference pipeline using Azure services: `rust Copy` `code` External Traffic -> Load Balancer -> Inference Server(s) -> Predictions

External Traffic: Incoming requests for document classification from external sources or client applications. Load Balancer: Utilize Azure Load Balancer or Azure Application Gateway for load balancing and distributing requests among multiple inference servers. Inference Server(s): Deploy multiple instances of the deployed model using Azure Kubernetes Service (AKS) or Azure Virtual Machines to handle inference requests. Predictions: Return the document classification predictions to the requesting client application or system. Remember to consider security measures, data privacy, and compliance requirements when designing the solution. These are high-level guidelines for designing an architecture for a Document Classification use case leveraging Azure Cloud services. Actual implementation details and specific services used may vary based on project requirements and constraints.

# Computer Vision

## Total Marks: 200

Each question 20 marks

### Question 1 -

Train a deep learning model which would classify the vegetables based on the images provided. The dataset can be accessed from the given link.

#### Link

<https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset>

#### Note -

1. Use PyTorch as the framework for training model
2. Use Distributed Parallel Training technique to optimize training time.
3. Achieve an accuracy of at least 85% on the validation dataset.
4. Use albumentations library for image transformation
5. Use TensorBoard logging for visualizing training performance
6. Use custom modular Python scripts to train model
7. Only Jupyter notebooks will not be allowed
8. Write code comments wherever needed for understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>



**Question 2 -**

From **Question 1**, you would get a trained model which would classify the vegetables based on the classes. You need to convert the trained model to ONNX format and achieve faster inference

**Note -**

1. There is no set inference time, but try to achieve as low an inference time as possible
2. Create a web app to interact with the model, where the user can upload the image and get predictions
3. Try to reduce the model size considerably so that inference time can be faster
4. Use modular Python scripts to train and infer the model
5. Only Jupyter notebooks will not be allowed
6. Write code comments whenever needed for understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 3 -**

Scrap the images from popular e-commerce websites for various product images sold on those websites. Your goal is to fetch the images from the website, create categories of different product classes and train a deep learning model to classify the same based on the user input.

**Note -**

1. You can use any framework of your choice like TensorFlow or PyTorch
2. You have to **not use any** pre-trained model, but instead create your own custom architecture and then train the model.
3. Write code comments wherever needed for understanding
4. Try to use little big dataset so that model can be generalized
5. Write modular Python scripts to train and infer the model
6. Only Jupyter Notebook will be not allowed
7. Write code comments wherever needed for code understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 4 -**

You have to train a custom YOLO V7 model on the dataset which is linked below. Your goal is to detect different products based on the given classes based on the user input

**Link -**

[https://drive.google.com/file/d/1MEgDYJwO\\_PVVfAbyfjaRHxt7qoiBBHYt/view?usp=share\\_link](https://drive.google.com/file/d/1MEgDYJwO_PVVfAbyfjaRHxt7qoiBBHYt/view?usp=share_link)

**Note -**

1. You have to use PyTorch implementation of YOLO V7
  2. The dataset consists of 102 classes with train, validation, and test images already in the respective folders.
  3. Labeling is already done, given with the dataset, so need for annotation
  4. Since the dataset is small, try to achieve at least an mAP of 85
  5. Write modular Python scripts to train the model
  6. Write code comments wherever needed for understanding
- Computer Vision Assessment iNeuron 3
7. Only Jupyter Notebook will not be allowed

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 5 -**

From **Question 4**, you would have a custom-trained YOLO model. Your goal is to need to convert the model to ONNX format and reduce the inference time.

**Note -**

1. Reduce the inference time to as much as possible
2. Try to reduce the model size by using techniques like Quantization, etc
3. Create a web app for users to interact with your model where users can upload images and get predictions.
4. Write modular Python scripts to infer the model.
5. Only Jupyter notebooks are not allowed.
6. Write code comments wherever needed for code understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 6 -**

You have to train a custom segmentation model based on Detectron 2 framework. Your goal is to segment the given images based on the user input into the different classes

**Link -**

<https://www.kaggle.com/competitions/open-images-2019-instance-segmentation/data>

**Note -**

1. For this, only the Jupyter Notebook is fine
2. Labels are in COCO format.
3. Write code comments wherever needed for understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 7 -**

From **Question 6**, you would have custom trained segmentation model. Your goal is to reduce the model inference time

**Note -**

1. Reduce inference time to as much as possible
2. Create a web app for users to interact with your model where they can upload images and get predictions
3. Write code comments wherever needed for code understanding.

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 8 -**

You have to train a custom object detection model based on DETR (Detection Transformer)

**Link -** <https://www.kaggle.com/datasets/andrewmvd/helmet-detection>

**Note -**

1. You need to use HuggingFace PyTorch as the framework
2. The dataset is about detecting football players from the images provided
3. Data Annotations are already in COCO format.
4. Write custom Python scripts for training.
5. Write code comments wherever needed for code understanding
6. Only Jupyter Notebooks are not allowed

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 9 -**

From **Question 8**, you would have a custom object detection model **Note -**

1. Try to reduce the model size using quantization
2. Create a web app where the users can interact with your model
3. Write modular Python script for model inference
4. Only Jupyter Notebooks are not allowed
5. Write code comments wherever needed for code understanding

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

**Question 10 -**

From all the questions from 1 to 9, take any image classification model, object model detection model, and image segmentation model and deploy it in the cloud

**Note -** 1. Deployment of all 3 different models should be AWS, Azure, and GCP 2. A

video demo of the application working in the cloud should be good enough 3.  
Containerization of all 3 applications is important and should be pushed to Docker Hub

Computer Vision Assessment iNeuron 5

4. CI-CD pipelines using GitHub actions that would deploy the models in all 3 clouds are mandatory.

ANS- <https://syedbasar14-zany-broccoli-gjr6rjrqvvh9vjw.github.dev/>

# Natural Language Processing

**Total Marks: 200**

**Each question 20 marks**

**Q-1.** Take any YouTube videos link and your task is to extract the comments from that videos and store it in a csv file and then you need define what is most demanding topic in that videos comment section.

To extract comments from a YouTube video and determine the most demanding topic in the comment section, we can use Natural Language Processing techniques.

Here's a step-by-step approach:

Obtain the YouTube video link:

Please provide me with the YouTube video link you would like to analyze.

Extract comments: We can use the YouTube Data API or web scraping techniques to retrieve the comments from the video. There are Python libraries like `google-api-python-client` or `pytube` that can assist with this task.

Preprocess the comments: Apply NLP techniques to preprocess the comments. This typically involves removing punctuation, converting text to lowercase, removing stopwords (common words like "the," "is," etc.), and performing tokenization (splitting the text into individual words).

Analyze the topics: To determine the most demanding topic in the comment section, we can use topic modeling algorithms such as Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF). These algorithms can identify latent topics within the comments and assign them probabilities.

Assign topics to comments: Once we have the topics, we can assign each comment to the most relevant topic based on the highest probability assigned by the topic modeling algorithm.

Aggregate topic frequencies: Count the frequencies of each topic across all the comments to determine the most demanding topic. The topic with the highest frequency can be considered the most demanding topic in the comment section.

Store the results: Finally, store the comments along with their assigned topics

and the most demanding topic in a CSV file for further analysis or reporting.

**Q-2.** Take any pdf and your task is to extract the text from that pdf and store it in a csv file and then you need to find the most repeated word in that pdf.

To extract text from a PDF, store it in a CSV file, and find the most repeated word, we can utilize Natural Language Processing techniques. Here's a step-by-step approach: Obtain the PDF file: Please provide me with the PDF file you would like to analyze. Extract text from the PDF: We can use Python libraries such as PyPDF2 or pdfplumber to extract the text content from the PDF file. Preprocess the text: Apply NLP techniques to preprocess the extracted text. This typically involves removing punctuation, converting text to lowercase, and performing tokenization (splitting the text into individual words). Count word frequencies: Count the frequencies of each word in the preprocessed text. You can use Python's collections.Counter or similar methods to accomplish this. Find the most repeated word: Identify the word with the highest frequency count, which indicates the most repeated word in the PDF. Store the results: Store the extracted text and the most repeated word in a CSV file for further analysis or reporting.

**Q-3.** from question 2, As you got the CSV and now you need perform key word extraction from that csv file and do the Topic modeling

Certainly! To perform keyword extraction and topic modeling on the CSV file obtained from the previous question, we can follow these steps:

Load the CSV file: Read the CSV file containing the extracted text into a suitable data structure, such as a pandas DataFrame. Preprocess the text: Apply NLP techniques to preprocess the text, similar to what we discussed earlier. This may include removing punctuation, converting text to lowercase, removing stopwords, and performing tokenization. Keyword extraction: Utilize techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or TextRank to extract keywords from the preprocessed text. These algorithms can identify important words or phrases that are most representative of the content. Topic modeling: Apply topic modeling algorithms such as Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF) to identify latent topics within the text. These algorithms assign probabilities to topics and words, allowing us to uncover underlying themes or subjects. Assign keywords to topics: Once topics are identified, assign the extracted keywords to the most relevant topic based on their occurrence and association within the topic model. Analyze the results: Review the extracted keywords and their assigned topics to gain insights into the main themes or subjects discussed in the text.

**Q-4.** Take any text file and now your task is to Text Summarization without using hugging transformer library

If you want to perform text summarization without using the Hugging Face Transformers library, you can use a traditional algorithmic approach. One such method is the extractive text summarization technique, which involves selecting and combining important sentences from the text to create a summary. Here's a simplified step-by-step process: Load the text file: Read the text file and store its content in a variable. Preprocess the text: Apply preprocessing steps such as removing special characters, punctuation, and extra whitespaces. You can also convert the text to lowercase for consistency. Sentence tokenization: Split the preprocessed text into individual sentences. You can use punctuation marks like periods, exclamation marks, and question marks as indicators to identify sentence boundaries. Sentence scoring: Assign a score to each sentence based on its importance or relevance to the overall text. There are multiple methods to calculate sentence scores, such as using sentence length, word frequency, or term weights like TF-IDF (Term Frequency-Inverse Document Frequency). Select top-scoring sentences: Determine a threshold or select a fixed number of sentences with the highest scores. These sentences will be the basis for the summary. Generate the summary: Combine the selected sentences to form a coherent summary. You can preserve the order of the sentences as they appear in the original text or rearrange them to enhance readability. Display or store the summary: Present the generated summary to the user or store it in a file for further use.

**Q-5.** Now you need build your own language detection with the fast Text model by Facebook and

To build your own language detection model using the FastText library by Facebook, you can follow these steps: Data collection and preprocessing: Gather a dataset of labeled text samples, where each sample is associated with its corresponding language. Preprocess the text by removing any unnecessary characters, normalizing the text, and splitting it into individual words or tokens. Data format: Convert your preprocessed data into the format required by FastText. The library expects a text file where each line contains a space-separated sequence of words, followed by the corresponding label or language identifier. Training the model: Use FastText to train a language detection model on your dataset. FastText provides command-line tools that you can use for this purpose. The training process involves specifying the input file, output file for the model, and parameters such as the learning rate, dimensionality of word embeddings, and number of training epochs. Model evaluation: Evaluate the performance of your trained model using a separate test dataset. This will give you insights into its accuracy and effectiveness in language detection. Prediction: Once the model is trained and evaluated, you can use it for language detection on new, unseen text samples. Provide the input text to the model, and it will predict the language based on its learned knowledge.

**Q-6.** Generate research papers titles using Bert model and containerize the application and push to public docker hub

To generate research paper titles using the BERT model and containerize

the application for deployment, you can follow these steps: Data collection and preprocessing: Gather a dataset of research paper titles that can be used to train the BERT model. Preprocess the titles by removing any unnecessary characters, normalizing the text, and ensuring the data is in a suitable format for training. Fine-tuning BERT: Use the pre-trained BERT model, such as bert-base-uncased, and fine-tune it on your research paper title dataset. Fine-tuning involves training the BERT model on your specific task, which in this case is generating research paper titles. Building the application: Develop an application that takes input prompts or keywords and generates research paper titles using the fine-tuned BERT model. You can use Python and frameworks like Flask or Django for this purpose. Containerization: Containerize your application using Docker. Create a Dockerfile that specifies the necessary dependencies, such as Python, required libraries, and the application code. Build a Docker image based on this Dockerfile. Testing and local deployment: Run the Docker image locally to ensure the application works as expected within the container environment. Test the title generation functionality and make any necessary adjustments to the application code. Docker Hub: Sign up for a Docker Hub account if you don't already have one. Create a repository on Docker Hub to store and publish your Docker image. Pushing the Docker image: Tag your locally built Docker image with the repository information, and then push it to your Docker Hub repository using the docker push command. This will make your application accessible to others who can pull the image from Docker Hub. Documentation: Provide clear instructions on how to use your containerized application and generate research paper titles using the BERT model. Include any relevant details such as input format, output format, and any additional configurations or considerations.

**Q-7.** Now you need to build your own chatbot using the seq2seq model of Amazon website by scrape the website and containerize the application and push to public docker hub

Building a chatbot using the seq2seq model trained on Amazon website data, containerizing the application, and pushing it to the public Docker Hub involves several complex steps. Here's an overview of the process: Data collection: Scrape relevant data from the Amazon website, including product descriptions, reviews, or customer questions and answers. This data will be used to train the seq2seq model for the chatbot. Preprocessing: Clean and preprocess the scraped data. This involves removing any unwanted characters, normalizing the text, and splitting it into input-output pairs for training the seq2seq model. Seq2Seq model training: Use the preprocessed data to train a seq2seq model. Seq2seq models consist of an encoder-decoder architecture that maps input sequences to output sequences. The encoder processes the input, and the decoder generates the appropriate response. Building the chatbot application: Develop an application that takes user queries as input and generates responses using the trained seq2seq model. You can use Python and frameworks like Flask or Django to build the chatbot application. Containerization: Containerize the chatbot application using Docker. Create a Dockerfile that specifies the necessary dependencies,



including Python, required libraries, and the application code. Build a Docker image based on this Dockerfile. Local testing and deployment: Run the Docker image locally to ensure the chatbot application works as expected within the container environment. Test the chatbot's response generation capabilities and make any necessary adjustments to the application code. Docker Hub: Sign up for a Docker Hub account if you don't already have one. Create a repository on Docker Hub to store and publish your Docker image. Pushing the Docker image: Tag your locally built Docker image with the repository information and then push it to your Docker Hub repository using the `docker push` command. This will make your chatbot application accessible to others who can pull the image from Docker Hub. Documentation: Provide clear instructions on how to interact with your chatbot, including input format, expected responses, and any additional configurations or considerations. Explain how to use the chatbot effectively.

**Q-8.** Take a any own dataset and build a knowledge bot using Llama model.

**Q-9.** Using wisher you need transcribe any audio file and then you need to convert that audio file into text file and now convert that text file into audio file of different language.

To transcribe an audio file, convert it into a text file, and then convert the text file into an audio file in a different language, you can follow these steps: Choose a speech recognition service: Select a speech recognition service or library that can transcribe the audio file into text. Popular options include Google Cloud Speech-to-Text, IBM Watson Speech to Text, or Mozilla DeepSpeech. Transcribe the audio file: Use the chosen speech recognition service to transcribe the audio file into text. This typically involves sending the audio file to the service's API and receiving the transcribed text as a response. Save the transcribed text: Store the transcribed text in a file, such as a plain text file or a CSV file, for further processing. Language translation: Utilize a language translation service or library to convert the transcribed text into a different language. Options like Google Translate API, Microsoft Translator API, or libraries such as `translate` in Python can be used for this purpose. Specify the source language of the transcribed text and the target language for translation. Generate the translated text: Pass the transcribed text through the language translation service, which will provide the translated text as the output. Text-to-speech synthesis: Use a text-to-speech synthesis service or library to convert the translated text into an audio file. Services like Google Text-to-Speech, Amazon Polly, or libraries such as `pyttsx3` in Python can be used for this step. Specify the desired language and voice for the audio output. Save the audio file: Save the generated audio file in a suitable format, such as WAV or MP3.

**Q-10.** Build a whole End- End api and deploy it on Heroku /railways so the task is that you need build a Auto-Correction of text using NLP

Note: only Jupyter notebook is not allowed from 5th question

To build an end-to-end API for auto-correction of text using NLP and deploy it on a platform like Heroku or Railways, you can follow these steps: Data preprocessing: Prepare a dataset or gather text

data to train your auto-correction model. Clean and preprocess the data by removing any unwanted characters, normalizing the text, and splitting it into training and validation sets. Model development: Build a model for auto-correction using NLP techniques such as sequence-to-sequence models, language models, or transformer models. Train the model on your prepared dataset, adjusting hyperparameters and performing validation to optimize performance. API development: Create an API using a framework like Flask or FastAPI. Define appropriate endpoints for handling incoming requests and returning auto-corrected text responses. Implement the logic to preprocess incoming text and pass it through your trained model for correction. Model serialization: Serialize your trained model to a file format that can be loaded by the API. Common formats include pickle, HDF5, or SavedModel. API deployment: Choose a platform for deployment, such as Heroku or Railways. Create an account, set up the necessary project configurations, and deploy your API codebase to the chosen platform. Follow the platform's documentation for guidance on deployment procedures specific to the platform you've chosen. Testing and validation: Once the API is deployed, test it using sample inputs and verify that the auto-correction functionality is working as expected. Validate the corrections and make any necessary adjustments to improve the accuracy and performance of the auto-correction model. Documentation and usage instructions: Provide clear instructions on how to interact with your API, including the API endpoints, input format, expected response format, and any additional configurations or considerations. Explain how to utilize the auto-correction functionality effectively.

## ← Submission Process →

There are Two Types of Questions Theory based Question and Project-based (where you actually have to code)

First of all, You have to create an Google doc, where you will add answers of all the questions

If you are attempting a question in which you have to write code, so create a repo push your code to repo and copy the link of repo and add it into docs as shown below

*Eg. Answer. 6 Python - > GitHub repo link*

*Note:*

- *If you are building any End to end project try to write code in .py file*
- *If you are only analyzing or doing EDA use .ipynb file*

If you are attempting a theory-based question then you have to add the answer in the same google docs as it's

Then submit that final link (google doc link which has all the answers)