Senior DataScien

Client Name
EY

Employee Name
Ani

Date of Attempt
07-Dec-2021

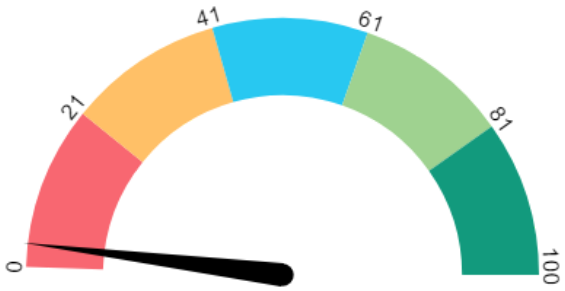
Employee ID
-

Index

Score Analysis

Score: 3/ 107 **Time Taken: 1 min 59 sec / 194 min**

No Knowledge (3%)



No Knowledge	Beginner	Intermediate	Knowledgeable	Expert
(0% - 20%)	(21% - 40%)	(41% - 60%)	(61% - 80%)	(81% - 100%)

Ani scored **3%** and completed assessment in **1%** of the alloted time

Section Score Analysis

Section	Percentage
Python_Coding	0/20 (0%)
Python	0/9 (0%)
Machine Learning_LogicBox	0/20 (0%)
Machine Learning_Coding	0/20 (0%)
Machine Learning	2/8 (25%)
SAS	1/10 (10%)
Data Visualisation & Reporting	0/10 (0%)
Statistics	0/10 (0%)

Section Skill Analysis

Section 1: Python_Coding

Total Score: **0/ 20** Negative Points: **0** Time Taken: **51 sec/30 min**

Question Analysis:

Total Question: **1** Correct: **0** Wrong: **1** Skipped: **0** Not Answered: **0**

Skills	#Questions	Skill Score
Coding - High	**1**	**0/20**

Section 2: Python

Total Score: **0/ 9** Negative Points: **0** Time Taken: **18 sec/18 min**

Question Analysis:

Total Question: **9** Correct: **0** Wrong: **9** Skipped: **0** Not Answered: **0**

Skills	#Questions	Skill Score
Python	**9**	**0/9**

Section 3: Machine Learning_LogicBox

Total Score: **0/ 20** Negative Points: **0** Time Taken: **15 sec/40 min**

Question Analysis:

Total Question: **4** Correct: **0** Wrong: **4** Skipped: **0** Not Answered: **4**

Skills	#Questions	Skill Score
Machine Learning AI-LogicBox	**4**	**0/20**

Section 4: Machine Learning_Coding

Total Score: **0/ 20** Negative Points: **0** Time Taken: **12 sec/30 min**

Question Analysis:

Total Question: **2** Correct: **0** Wrong: **2** Skipped: **0** Not Answered: **2**

Skills	#Questions	Skill Score
Machine Learning Coding	**2**	**0/20**

Section 5: Machine Learning

Total Score: **2/ 8** Negative Points: **0** Time Taken: **11 sec/16 min**

Question Analysis:

Total Question: **8** Correct: **2** Wrong: **3** Skipped: **0** Not Answered: **3**

Skills	#Questions	Skill Score
Machine Learning (Advanced)	4	1/4
Machine Learning (Intermediate)	4	1/4

Section 6: SAS

Total Score: **1/ 10** Negative Points: **0** Time Taken: **2 sec/20 min**

Question Analysis:

Total Question: **10** Correct: **1** Wrong: **0** Skipped: **0** Not Answered: **9**

Skills	#Questions	Skill Score
BASE SAS	10	1/10

Section 7: Data Visualisation & Reporting

Total Score: **0/ 10** Negative Points: **0** Time Taken: **6 sec/20 min**

Question Analysis:

Total Question: **10** Correct: **0** Wrong: **1** Skipped: **0** Not Answered: **9**

Skills	#Questions	Skill Score
Data Visualisation	10	0/10

Section 8: Statistics

Total Score: **0/ 10** Negative Points: **0** Time Taken: **4 sec/20 min**

Question Analysis:

Total Question: **10** Correct: **0** Wrong: **1** Skipped: **0** Not Answered: **9**

Skills	#Questions	Skill Score

Skills	#Questions	Skill Score
Applied Statistics	2	0/2
Statistics & Probability	3	0/3
Statistics for Machine Learning	3	0/3
Statistical Modeling	2	0/2

Identification of strengths and skill improvement needs

Strength

Congratulations! We have identified as your strengths.

Improvement area

Based on your score, are the identified areas of improvement.

A guide to get started on your Individual Development Plan (IDP) :

Difficulty Level Analysis

Level	Number of Questions	Correct Attempts	Correctness
Easy	0	0	0%
Medium	18	2	11.11%
Hard	36	1	2.78%

Proctoring Analysis

Image or Video Proctoring is not enabled for this test.

Note: The total violations are based on the custom violation settings for this test. The number of consecutive images considered as one violation is configured for all the categories (Unrecognized Face, Multiple Faces, No Face) and may differ from the default settings.

Window Violation: 0 **Time Violation: 0 min**

Time Flag: Early Finisher Flag

This candidate completed the test in 1 min 59 sec which is significantly faster than the average test time of 25 min 42 sec.

Test Log

No data available

Question Details

Question: **#1**	Type: **Coding**	Skill: **Coding - High**	Status: **Answered**
Result: **Wrong**	Level: **Hard**	Time Taken: **51 sec**	Average Time: **13 min 0 sec**
Score: **0 / 20**	Window Violation: **0 times**	Time Violation: **0 sec**	

Question #1

Data Structures: Stacks/Queues

A school has decided to supply packed **lunch boxes** to students.

- There are ***N*** lunch boxes placed on each other. The lunch boxes are either **circular or rectangular** in shape.
- Each student has his/her **own preference** for the **type of lunchbox.**

If the student finds that the tiffin at the top of the stack **is not as per his/her preference** (of shape), he/she will go back and rejoin the queue and the process will continue.

Estimate the **number of students who will not be able to get lunch**.

Function Description

In the provided code snippet, implement the provided `getLunch(...)` method using the variables to print or return the output. You can write your code in the space below the phrase **"`WRITE YOUR LOGIC HERE`"**.

There will be multiple test cases running, so the Input and Output should match exactly as provided.

The base output variable `result` is set to a default value of `-404`, which can be modified. Additionally, you can add or remove these output variables.

Input Format

Input1: **N**, denoting the number of children and lunchboxes.

Input2: An **array of N elements**, each element can be either 0 (rectangle) or 1 (circle) denoting the type of lunch box from top to bottom.

Input3: An **array of N elements**, each element denoting the preference 0 (rectangle) or 1 (Circle) of a student from the start till the end of the queue.

Sample Input 1

```
4          -- Number of children and lunchboxes
0 0 1 0    -- Types of lunch boxes
1 0 0 0    -- Preferences of respective students
```

Sample Input 2

```
6              -- Number of children and lunchboxes
0 1 1 0 1 0    -- Types of lunch boxes
1 1 1 0 1 0    -- Preferences of respective students
```

Output Format

For the given input, your code should output the number of students who will not be able to eat lunch.

Sample Output 1

```
0
```

Sample Output 2

```
1
```

Explanation 1

Here, we have 3 rectangular and 1 circular-shaped tiffin boxes.
We also have 3 students who prefer a rectangular tiffin box and 1 student who prefers a circular tiffin box.
Therefore, everyone will be able to eat lunch. Hence, the output is ***0***.

Explanation 2

Here, we have 3 rectangular and 3 circular-shaped tiffin boxes.
There are 4 students who prefer a rectangular tiffin box and 2 students who prefer a circular tiffin box.
However, as required by the problem statement, 1 student will keep rotating in the queue until only the boxes that are not of his choice remain in the stack.
Therefore, 1 student will not be able to eat lunch. Hence, the output is ***1***.

Answer:

Coding Language: Python

Candidate Code:

```
def getLunch(N, LunchBoxes, preference):

    #this is default OUTPUT. You can change it.
    result = -404

    # write your Logic here:


    return result


# INPUT [uncomment & modify if required]
N = int(raw_input())

LunchBoxes = ['x']
preference = ['y']

LunchBoxes = list(map(int, raw_input().split()))

preference = list(map(int, raw_input().split()))

# OUTPUT [uncomment & modify if required]
print(getLunch(N, LunchBoxes, preference))
```

Compilation Summary:

Compilation Status: **Compile Successfully**	No Of Compilations: **1**
Defualt Input:	Candidate Output:
4 0 1 1 0 1 0 0 1	-404

Test Case Summary:

Test Case: **1** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
40 0 1 01 0 0 0	0	-404

Test Case: **2** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
41 0 1 00 0 0 0	4	-404

Test Case: **3** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
80 0 1 1 1 0 1 11 1 0 0 0 0 0 0	4	-404

Test Case: **4** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
111 0 0 1 0 1 1 0 0 1 00 0 1 0 0 1 0 1 0 0 1	2	-404

Test Case: **5** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
40 1 0 00 0 0 0	3	-404

Test Case: **6** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
40 1 1 01 0 0 1	0	-404

Question: **#2**	Type: **Coding**	Skill: **Machine Learning Coding**	Status: **Not Answered**
Result: **Wrong**	Level: **Medium**	Time Taken: **12 sec**	Average Time: **2 min 43 sec**
Score: **0 / 10**	Window Violation: **0 times**	Time Violation: **0 sec**	

Question #2

Data Modeling: Linear Regression

You are given 2 features and a continuous decision variable.

You decide to model the decision variable using Linear Regression.

Since the dataset is very small, in order to evaluate your model performance, you apply 2-fold Cross-Validation to find the mean R-squared score. Therefore, for a given dataset, ***print the mean R-squared score after applying 2-fold Cross-Validation to the best-fit regression line, rounded up to 2 decimal places.***

Note

Print the output up to 2 decimal places. Example: 2.33, 4.66

Input Format

The first line contains an integer ***n*** denoting the number of ***data points***.

The next ***n*** lines contain a space-separated list of floating-point numbers denoting the ***feature values*** for each data point.

The next line contains a space-separated list of floating-point numbers denoting the ***decision variable***.

Output Format

Print a single floating point number denoting the mean R-squared score after applying 2-fold Cross-Validation to the best-fit regression line rounded up to 2 decimal places.

Sample Input

```
10
-2.5655825997563415 3.067268170254235
0.10502357218287406 2.1887071607233053
-0.13473353510349195 -1.4240384829872026
6.009052134894533 5.261420384218355
1.444707982235458 8.552440592156758
2.942353138040424 6.4114292007033775
1.647594582415084 4.0445858358965
2.54052599648529 -0.21402071588268434
6.306562749771207 0.8674683637535625
0.07694097386227527 -1.5881119118493707
-39.09780627250363 1.5870060861178077 0.1151122469899191 106.04026203669515
24.0123498945421 35.016517665991415 11.816766735928333 -2.976933513882667
63.03936700420861 -0.03311547701339884
```

Sample Output

```
0.04
```

Explanation

The input is first taken as it is and split into the given number of folds denoting training and test sets. Linear Regression is then fit to the training set and the R2 Score is recorded on the test set. Finally, the mean of all such R2 scores is ***0.04.***

Answer:

Coding Language: Python with ML

Candidate Code:

Compilation Summary:

Compilation Status:	No Of Compilations: **0**
Defualt Input:	Candidate Output:

Test Case Summary:

Test Case: **1** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
10-2.5655825997563415 3.0672681702542350.105 02357218287406 2.1887071607233053-0.13473353 510349195 -1.42403848298720266.009052134894 533 5.2614203842183551.444707982235458 8.552 4405921567582.942353138040424 6.41142920070 337751.647594582415084 4.04458583589652.54 052599648529 -0.21402071588268434 6.3065627 49771207 0.86746836375356250.0769409738622 7527 -1.5881119118493707-39.09780627250363 1.5 870060861178077 0.1151122469899191 106.0402620 3669515 24.0123498945421 35.016517665991415 1 1.816766735928333 -2.976933513882667 63.0393 6700420861 -0.03311547701339884	0.04	

Test Case: **2** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
105.49046546189766 2.1254971818988586.20298 8122954237 2.0004763507361113.84730082877831 2 -2.233437495225969-1.3913117292962172 1.5630 5985400939523.8577492311763986 1.530696082 4422315.274939966951267 1.976753115626976-1.9 588996810491874 0.46693982483416613.17681981 98796993 -0.24611875327560506 2.190539484214 836 -0.76067614698216436.166628712396949 3.9 6714477487811750.97074679927426 73.047633198 87613 -15.665393993772422 -12.74402939856752 5 14.337332655120242 44.13198042643103 -14.909 479880997534 -2.5059901130391062 -5.8795236 49537885 96.50860764038917	0.84	

Test Case: **3** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output

Test Case Input	Expected Output	Actual Output
10-2.039391364189072 11.2643361540008263.999 298184485337 -4.1318558282609371.4187843495 093027 6.2043901693507152.867059130163137 4.7 091270039102292.44208953817117 4.1191726037112 74-0.7976773070001024 -0.113708783896563143. 5701835979454173 2.2497282520827531.2284604 005601794 2.7573909996138626.514085122815103 5 4.076225910129345-0.5671199484481324 1.9005 407493494915-61.72167843438855 -30.38248148 129769 16.941317432226146 25.18368827563920 6 15.750909662144501 -0.21013449019650277 18.0 44448667932514 5.480370878473196 111.13722313 165238 -4.781326986010032	-1.84	

Test Case: **4** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
102.6015229146349785 1.97668423919999265.520 156967322977 -2.79050293516763763.920670472 4782475 3.9663103987955255-4.205646092085 033 -2.221130498572429-0.527626430409960 3 0.108847786220596322.172852545597119 3.069 59520886581456.078121348798957 2.523229842 0320041.3399286239672277 1.552230029396683 52.4846892108427703 -1.12843795413151154.3573 59808568527 5.4976569405601817.88081663369 9281 -3.086669783238414 31.878294189341965 - 58.608995787056905 -1.741204394687984 11.299 647032992024 74.72753251959006 2.7835297503 52657 -7.122409160314122 54.980277629945846	0.20	

Test Case Input	Expected Output	Actual Output

Test Case: **5** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
107.791722348745398 2.4580205030758124-1.1157 343161243367 6.9063056517629932.25593698914 04693 0.8070534940861040.386750053448843 7 2.46267517338051262.8497900178902174 1.19519 62790427062-0.5234523943584879 2.59368269 03921982.840993053746378 3.207836014982263 -5.194453133662217 -2.06342647811601142.05250 9040182347 1.62428862385388120.804169402935 5877 0.8948067870159655161.62922310953383 -1 9.855135504948286 1.7480004496231065 0.76513 92026309045 4.874759545433907 -3.663484567 1671417 14.82828530608101 -109.2858565722170 3 3.681689776980598 0.9760855308836365	0.19	

Test Case: **6** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
104.8442786904319135 6.218387669962065-1.605 208705835599 1.89715678465824643.1249669261 52833 2.8951826621091103-1.596240021820904 - 3.5567435609901317-0.16266499586600736 -0.0 61998524434338622.6350705410895037 6.30463 70290170843.9391880396366474 8.09343445293 23820.3040977723648419 1.8860141545581643-2. 1040498137342203 2.6049352259896663-0.8733 063777741155 -4.0229908267023674.3550461415 0739 -14.840104866530416 16.38253377603788 2. 1360181808748857 0.4210493393857509 29.41939 3619734073 67.15191701281977 1.262280492671909 8 -25.256881919136752 4.236481297298631	0.78	

Question: **#3**	Type: **Coding**	Skill: **Machine Learning Coding**	Status: **Not Answered**
Result: **Wrong**	Level: **Medium**	Time Taken: **0 sec**	Average Time: **34 sec**
Score: **0 / 10**	Window Violation: **0 times**	Time Violation: **0 sec**	

Question #3

Find the maximum R-squared score

You are working on a dataset that has 2 features and 1 continuous decision variable.

You decide to apply linear regression to predict the decision variable.

You think that adding more features which are a combination of existing features might lead to better best-fit lines.

Therefore, for the dataset at hand, you should do the following:

- One by one, try adding permutations of products of the given features to the dataset. For Example: If the dataset has 2 features, try adding $F1^2$, $F2^2$, and F1*F2 one by one.
- Note the average R-squared score after each feature added to the dataset on 2-fold cross-validation of the best-fit line.

Finally, you need to ***print the maximum R-squared score rounded up to 2 decimal places after doing the above-mentioned procedure.***

Input Format

The input contains an integer ***n*** denoting the number of ***data points***.
The next ***n*** lines contain a space-separated list of floating-point numbers denoting

the ***feature values for each data point.***
The next line contains a space-separated list of floating-point numbers denoting the ***decision variable***.

Output Format

Print a single floating point number denoting the maximum R2 Score of the 2-Fold Cross Validated Linear Regression Model rounded up to 2 decimal places.

Sample Input

```
10
4.668967417885997 0.9680039617262333
0.04070926923867635 6.235601342719975
2.4607406247418284 6.050199862052582
-0.19853954998217382 0.7316541816343052
1.4453891247828148 4.833223978549363
2.1725241540313593 -1.9287257992349778
5.9729160436607245 0.8620262985894125
2.5908484598707666 3.3706861607844947
-2.0331978641548023 -0.21869962231003104
-0.5702875486463874 1.3118930636715922
22.858095882758263 0.815236033656948 28.958673507372826
-2.175381657182461 12.39361138617021 -11.486876427812753 52.72269638359712
14.932908000511276 -14.331509366112709 -2.1630639847347837
```

Sample Output

```
0.72
```

Explanation

The input is first taken as it is and one by one each feature permutation is added to the data to make a new dataset. The new dataset is then split into the given number of folds denoting training and test sets. Linear Regression is then fit to the training set and the R2 Score is recorded on the test set. The mean of R2 scores on each fold is recorded and denoted as the final mean R2 score for that feature permutation. The maximum of all such R2 Scores is ***0.72***.

Answer:

Coding Language:

Candidate Code:

Compilation Summary:

Compilation Status:	No Of Compilations: **0**
Defualt Input:	Candidate Output:

Test Case Summary:

Test Case: **1** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
104.668967417885997 0.96800396172623330.040 70926923867635 6.2356013427199752.46074062 47418284 6.050199862052582-0.19853954998217 382 0.73165418163430521.4453891247828148 4.83 32239785493632.1725241540313593 -1.92872579 923497785.9729160436607245 0.8620262985894 1252.5908484598707666 3.3706861607844947-2. 0331978641548023 -0.21869962231003104-0.5702 875486463874 1.311893063671592222.858095882 758263 0.815236033656948 28.95867350737282 6 -2.175381657182461 12.39361138617021 -11.48687 6427812753 52.72269638359712 14.932908000511 276 -14.331509366112709 -2.1630639847347837	`0.72`	

Test Case: **2** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
10-0.7979127163604423 0.9701170110050863-0.62 03049202529076 4.8960347507012952.0506696 908757984 -0.21458022386569821.03553446389 3462 3.58248140153313830.5182355692465237 5.1 9087314627644-0.4105736530154429 -1.18594281 644962680.8310735104846247 -1.39810711725233 15.156818068122139 -0.07073285661969919-2.297 61063471405 4.9434033048948130.184188705958 1251 6.31175736596554-3.9073198942851484 -7.4 089260557965355 -2.086662654681817 9.197081 008133974 3.5579047607461525 0.569199501558 4507 -3.084520575969758 18.93448174886627 - 40.448897743847034 1.0833123080010574	`0.53`	

Test Case: **3** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output

Test Case Input	Expected Output	Actual Output
10-0.24411984980901202 -3.82884770809453024. 0870660416828 4.4913222138551667.2822768968 3691 1.05432330113126631.4308437908266745 8.5 614544961687072.765845388362262 -0.1856645 37967291652.8066557168582276 3.568478221877 84921.7127486072082783 1.9661092221129727.974 931588996656 -2.01717701062718565.1642985383 13936 7.2160775624742654.021245245807713 -0.8 7166301237555421.5115889891408996 42.4873165 8130496 110.65803351176889 24.2427315647086 3 -2.493141827437184 18.443415387091157 3.6600 5306969779 102.82215728564056 94.2845369911 7279 -4.408465074693566	0.70	

Test Case: **4** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
101.6532300986253183 8.647059403814832.12800 12212277026 3.1232703583663715-3.61537077062 41462 1.02633826344432544.024214082202585 3. 33494462142684345.701699814673905 0.409553 5177881431-0.7778879416542774 2.6270761681611 4880.23543642949892352 4.133587481142497-1.8 489702201759428 3.618341612606793-1.25516060 4875508 0.3034976674990108 63.9104195238765	0.05	

Test Case Input	Expected Output	Actual Output
707 2.020658787273835527.527441869424358 10. 526063757482385 -61.43912796560078 30.87407 725280257 37.67690353568192 -4.754041153627 977 2.6282186981792623 -25.418314460027688 - 5.18536792545289 18.332962984075092		

Test Case: **5** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
105.233174182093446 0.351835347216331145.45611 0160373608 3.733271914514239-0.5554721823353 317 0.79618584618839133.0777990930660373 -0. 41269861688233480.02103479077107373 4.87534 8407143538-2.9868812444446693 -3.175656385 9890825-0.21232564357245343 0.250670489558 83475-0.29285407105398154 2.447201474595039 6.870514218819106 -1.040520496011244 41.7073521 109170426 4.76147185781012925.79861855485263 4 67.1583004430352 -1.7927994820126814 -4.239 241379242365 -0.09771834393031337 -14.323736 615931054 -0.7762771860585083 -1.88327725742 77619 60.506973437310265 14.621191891585536	0.52	

Test Case: **6** Status: **Fail** Score:**0**

Test Case Input	Expected Output	Actual Output
102.050443355836034 -0.33151456137535854.621 265005108338 5.72853880344015-0.3566047014 9463386 -0.67153999305325440.6385014049049 82 4.7554574133593075-0.08447081175943971 3. 7833259330987641.628560150742153 0.64474367 161505415.4800779278674145 5.55465337681353 4-1.298274865331099 4.5295915459865652.0714 56219398078 -0.66747442676105570.969824903 3934143 3.6433949990803236-3.6199337506305 2 62.59992013222866 0.9440953208120405 6.55 9400238265667 -0.5427076780933632 0.862794 4024832155 88.04077175652797 -17.4119119810601 54 -5.085141073584056 6.461477189885124	0.93	

Question: **#4**	Type: **AI-LogicBox**	Skill: **Machine Learning AI-LogicBox**	Status: **Not Answered**
Result: **Wrong**	Level: **Hard**	Time Taken: **2 sec**	Average Time: **2 sec**
Score: **0 / 5**	Window Violation: **0 times**	Time Violation: **0 sec**	No. of Runs & Validations: **0**

Question #4

You have a credit card fraud classification dataset at hand. For each credit card transaction, the dataset has several eight independent variables and a single dependent variable denoting whether the transaction was fraudulent or not. The independent variables include variables like the speed of transaction, amount, etc.

If the above dataset is modelled using a simple multi-layer perceptron with two hidden layers with 6 and 4 neurons respectively in Keras, fill in the following blanks as per the given instructions.

At Blank 1: Write the code to fill in the correct input shape.

At Blank 2: Write the code to fill in the correct layer that should be used to build the network.

At Blank 3: What should be the size of the output layer of the network?

At Blank 4: What should be the activation of the

Sample Script:

```
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense

X = data.features
y = data.target

inp_size = Blank 1: Write your code her
classifier = Sequential()
classifier.add(Blank 2: Write your code her(units=6,
kernel_initializer='uniform', activation='relu',
input_dim=inp_size))
classifier.add(Dense(units=4,
kernel_initializer='uniform',
activation='relu'))
classifier.add(Dense(units=""
Blank 3: Write your code her"",
kernel_initializer='uniform', activation ='
Blank 4: Write your code her'))
classifier.compile(optimizer='adam', loss='
```

output layer of the network?

At Blank 5: What loss function should be used to train the network?

```
Blank 5: Write your code her', metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size =
32, epochs = 100)
```

Answer:

Question: **#5**	Type: **AI-LogicBox**	Skill: **Machine Learning AI-LogicBox**	Status: **Not Answered**
Result: **Wrong**	Level: **Hard**	Time Taken: **4 sec**	Average Time: **4 sec**
Score: **0 / 5**	Window Violation: **0 times**	Time Violation: **0 sec**	No. of Runs & Validations: **0**

Question #5

The scikit-learn library in Python is a well-known library used to solve modelling problems. You have the following dataset at hand:

Citrus Content (g/L)	Sugar Content (g/L)	Beverage Type (1/0 for Wine/Energy Drink)
2	1.54	0
4.5	4.6	1
1.8	10	0
6.6	12.21	1

In relation to the modelling of the above dataset (Citrus and Sugar Content being the independent variables, Beverage Typ being the dependent variable), fill in the blanks below:

At Blank 1: Fill in the blank regarding importing the relevant libraries to model the problem using a voting classifier that uses Logistic Regression, Decision Tree and Random Forest Classifiers as constituents.

At Blank 2: Write the code to define a Logistic Regression Model with a random state equal to 1.

At Blank 3: Write the code to define a Decision Tree Classifier Model with a random state equal to 1.

At Blank 4: Write the code to define a Random Forest Classifier Model with a random state equal to 1 and the number of estimators equal to 50.

At Blank 5: Write the code to declare and fit a Voting Classifier using the above three

Sample Script:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import
LogisticRegression
from sklearn.tree import
DecisionTreeClassifier
from sklearn.Blank 1: Write your code herimport
RandomForestClassifier, VotingClassifier

X = data.features
y = data.target

clf1 =
Blank 2: Write your code here
clf2 =
Blank 3: Write your code here
clf3 =
Blank 4: Write your code here

voting_clf =
Blank 5: Write your code here
```

classifiers on the given data. The names spaces for each of the classifiers declared in blanks 2, 3, and 4 should be 'lr', 'dtc', and 'rf', and the voting should be hard.

Answer:

Question: **#6**	Type: **AI-LogicBox**	Skill: **Machine Learning AI-LogicBox**	Status: **Not Answered**
Result: **Wrong**	Level: **Hard**	Time Taken: **2 sec**	Average Time: **2 sec**
Score: **0 / 5**	Window Violation: **0 times**	Time Violation: **0 sec**	No. of Runs & Validations: **0**

Question #6

The statsmodels library in Python is a well-known library used to solve time-series forecasting problems. You have a time-series of the past one year of daily electricity consumptions of a locality. Since electricity consumption has a strong seasonal nature, you want to predict the next-day electricity consumption using the given data by applying time-series forecasting.

In relation to the problem, as described above, fill in the blanks below:

At Blank 1: Write the code to import the relevant libraries required to model the problem using ARIMA

At Blank 2: Write the code to import the relevant libraries to perform the dickey-fuller stationarity tests on the time series.

At Blank 3: According to the nature of the problem, the original time series has seasonality as well as trend. Hence it must not be stationary. Write the code to find the absolute difference between the Dickey-Fuller ADF Statistic and the 1% critical value.

At Blank 4: Write the code that outputs a boolean value denoting if the Dickey-Fuller p-value is above or below 0.05. Print true for below and false for above.

At Blank 5: Write the code to find the transformed stationary time series by finding the differenced time-series using the given

Sample Script:

```
from random import random
Blank 1: Write your code here

Blank 2: Write your code here

data = [x + random() for x in range(252)]

def difference(dataset):
   diff = list()
   for i in range(1, len(dataset)):
      value = dataset[i] - dataset[i - 1]
      diff.append(value)
    return numpy.array(diff)

abs_diff =
Blank 3: Write your code here

Blank 4: Write your code here

data_diff =
Blank 5: Write your code here
```

function.

Answer:

Question: **#7**	Type: **AI-LogicBox**	Skill: **Machine Learning AI-LogicBox**	Status: **Not Answered**
Result: **Wrong**	Level: **Hard**	Time Taken: **7 sec**	Average Time: **7 sec**
Score: **0 / 5**	Window Violation: **0 times**	Time Violation: **0 sec**	No. of Runs & Validations: **0**

Question #7

You need to perform text classification on a dataset of essays on a particular domain and provide a score in the range of 0 to 1. The data schema is a follows:

Essay	Score
The afternoon grew so glowering that in the sixth inning the arc lights were turned on--always a wan sight in the daytime, like the burning headlights of a funeral procession. Aided by the gloom, Fisher was slicing through the Sox rookies, and William Hildebrand come to bat in the seventh. He was second up in the eighth. This was almost certainly his last time to come to the plate in Fenway Park, and instead of merely cheering, as we had at his three previous appearances, we stood, all of us, and applauded.	0.8
Like his twisted feathers, his many scars, the reliable old owl chose the gnarled, weather-beaten, but solid branch often - it being a companion to the wise alone with the night and the last branch to creak in the heaviest wind. He often came to survey the fields and the clouds before his hunt, to listen to the steady sound of the stream passing through reeds under the bridge while combing his feathers for the unwanteds - whatever they might be.	0.32
Did you know that 7 out of 10 students have cheated at least once in the past year? Did you know that 50 % of those students have cheated more than twice? These shocking statistics are from a survey of 9,000 U.S. high school students. Incredibly, teachers may even be encouraging their students to cheat! Last year at a school in Detroit, teachers allegedly provided their students with answers to statewide standard tests.	0.97
1500	2

The essays are cleaned and tokenized and padded into fixed-length sequences of length 500. If the above dataset is modeled using an LSTM in Keras, fill the following blanks:

At Blank 1: Write the code to add the required layer according to the model specifications.

At Blank 2: The LSTM layer should have a hidden dimension of 300. Fill in the required value in the blank.

At Blank 3: Fill in the required value of the output neuron size in the blank.

Sample Script:

```
from sklearn.feature_extraction.text import CountVectorizer
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, LSTM, Embedding

vectorizer = CountVectorizer(binary=True, stop_words='english',
                             lowercase=True, min_df=3, max_df=0.9, max_features=5000)

vectorizer.fit(X_train)

word2idx = {word: idx for idx, word in enumerate(vectorizer.get_feature_names())}
tokenize = vectorizer.build_tokenizer()
preprocess = vectorizer.build_preprocessor()

def to_sequence(tokenizer, preprocessor, index, text):
    words = tokenizer(preprocessor(text))
    indexes = [index[word] for word in words if word in index]
    return indexes

X_train_sequences = [to_sequence(tokenize, preprocess, word2idx, x) for x in X_train]

MAX_SEQ_LENGHT = 500

N_FEATURES = len(vectorizer.get_feature_names())
```

At Blank 4: Fill in the type of loss that is required to fit the model.

At Blank 5: Write the code to fir the model on the given data with 512 batch size and 8 epochs.

```
X_train_sequences =
pad_sequences(X_train_sequences,
maxlen=MAX_SEQ_LENGHT,
value=N_FEATURES)


y_train = data.labels
model = Sequential()
model.add(Blank 1: Write your code her
(N_FEATURES + 1,
                300,

input_length=MAX_SEQ_LENGHT))
model.add(LSTM(Blank 2: Write your code her))
model.add(Dense(units=
Blank 3: Write your code her, activation='sigmoid'))

model.compile(loss='Blank 4: Write your code her',
optimizer='adam', metrics=['accuracy'])

print(model.summary())

Blank 5: Write your code here
```

Answer:

Disclaimer

This report is generated electronically based on the information provided by the assessment participants. It also includes flags, especially when proctoring services are utilized. However, it is important to emphasize that this report should not be the sole basis for making any business, selection, entrance, or employment-related decisions. iMocha assumes no responsibility for any consequences arising from the use of this report or any actions taken or refrained from as a result of it. This includes all business decisions made in reliance on the information, advice, or AI flags contained in this report, as well as any sources of information mentioned or referred to in the report.