Sample Candidate

Test ID: 325960026219455 | 0123456789 | sample@email.com

Test Date: December 17, 2021

Automata Data Science Pro

100 /100

Automata Data Science Pro

100 / 100

Data Wrangling and Munging	ML Workflow Implementation	Library Usage and Implementation
100 / 100	**100** / 100	**100** / 100

1 | Introduction

About the Report

This report provides a detailed analysis of the candidate's performance on different assessments. The tests for this job role were decided based on job analysis, O*Net taxonomy mapping and/or criterion validity studies. The candidate’s responses to these tests help construct a profile that reflects her/his likely performance level and achievement potential in the job role

This report has the following sections:

The **Summary** section provides an overall snapshot of the candidate’s performance. It includes a graphical representation of the test scores and the subsection scores.

The **Response** section captures the response provided by the candidate. This section includes only those tests that require a subjective input from the candidate and are scored based on artificial intelligence and machine learning.

The **Proctoring** section captures the output of the different proctoring features used during the test.

Score Interpretation

All the test scores are on a scale of 0-100. All the tests except personality and behavioural evaluation provide absolute scores. The personality and behavioural tests provide a norm-referenced score and hence, are percentile scores. Throughout the report, the colour codes used are as follows:

- Scores between 67 and 100
- Scores between 33 and 67
- Scores between 0 and 33

2 | Response

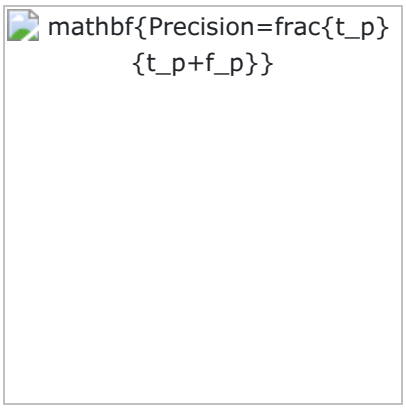
Automata Data Science Pro

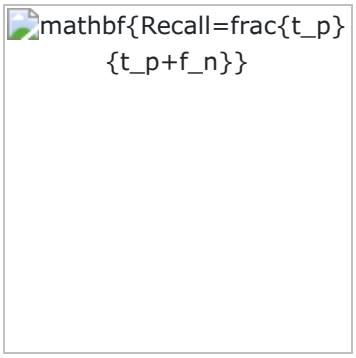
100 / 100

Question 1 (Language: Python3)

Emma, a data scientist, wishes to build a spam classifier that can classify emails as spam or non-spam. When given an input email as a vector, the classifier returns the probability of an email being spam. Emma decides to use a threshold on probability values according to which the classifier will decide whether the email is spam. She makes a set of threshold values and calculates precision and recall for each, in order to gauge the performance at various thresholds.

The formula for precision and recall is given by:





where

- t_p implies true positive
- f_p implies false positive
- f_n implies false negative

Given threshold values in the list **Thres**, compute corresponding precision and recall values.The class spam is positive in this case.

The input to the function **computePrecisionRecall** shall be column matrices (**y** and **y_pred**) and list **Thres**. **y** stores actual class labels and **y_pred** stores predicted probability. **Thres** stores the threshold values at which the predicted

actual class labels and **y_pred** stores predicted probability. **Thres** stores the threshold values at which the predicted class labels are to be computed.

The function should return a matrix. The first row of the matrix should contain precision values in the order of the corresponding threshold values. The second row should contain recall values in the same order.

The test cases tab illustrates some examples.

Final Code Submitted

```
# IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
# SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
# DEFINE ANY CLASS AND METHOD NEEDED
# THIS FUNCTION IS REQUIRED
#
# Parameters:  y: ndarray, shape (n_samples,1)
#           y_pred: ndarray, shape (n_samples,1)
#            Thres: list
#
# Returns:     score: list
#
def computePrecisionRecall(y,y_pred,Thres):

    # INSERT YOUR CODE HERE
    import numpy as np
    res = np.zeros((len(Thres),2))

    for i, thres in enumerate(Thres):
        y_pred_ = y_pred.copy()
        y_pred_[y_pred>=thres] = 1
        y_pred_[y_pred<thres] = 0
        tp = np.sum((y==1)&(y_pred_==1))
        fp = np.sum((y==0)&(y_pred_==1))
        fn = np.sum((y==1)&(y_pred_==0))
        res[i,0] = tp/(tp+fp)
        res[i,1] = tp/(tp+fn)
    return res
```

Test Case Execution

Total score

100%

Warnings

There are no errors in the candidate's code.

Compilation Statistics

33	23	0	3
Total attempts	Successful	Timed out	Runtime errors

Response time:	**00:29:20**
Average time taken between two compile attempts:	**00:00:53**
Average test case pass percentage per compile:	**6.7%**

Question 2 (Language: Python3)

Glen is a data science engineer at an online assessment company. The data from a particular test consists of various competency scores for candidates. His manager asks him to build a personality score prediction model based on the competency scores present in the input score data. He loads the input score data in a matrix and finds that some of the scores are missing. The missing scores are represented as **NaN**s. He decides to replace each **NaN** value with an average value of that feature. He then builds a model using linear regression to predict the values of personality scores. He evaluates his model by computing R-Squared statistic between actual and predicted personality scores.

In this question, you must replicate Glen's work.

The input to the function **linearRegressionWithMissingData** shall be two matrices, **X** and **y**. **X** represents the input score data matrix and **y** represents the column matrix for personality scores. The function must return the computed *R-Squared* statistic between actual and predicted personality scores.

The test cases tab illustrates some examples.

Final Code Submitted

```
1  # IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
2  # SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
3  # DEFINE ANY CLASS AND METHOD NEEDED
4  # THIS FUNCTION IS REQUIRED
5  #
6  # Parameters:  X: ndarray, shape (n_samples,n_features)
7  #              y: ndarray, shape (n_samples,1)
8  #
9  # Returns:    score: float
10 #
11 def linearRegressionWithMissingData(X,y):
12     # INSERT YOUR CODE HERE
13     import numpy as np
14     from sklearn.linear_model import LinearRegression
15     x_mean = np.nanmean(X,axis=0)
16     for i in range(X.shape[1]):
```

Test Case Execution

Total score

100%

Warnings

There are no errors in the candidate's code.

```
17          X[np.isnan(X[:,i]),i] = x_mean[i]
18      LR = LinearRegression().fit(X,y)
19      res = LR.score(X,y)
20      return res
```

Compilation Statistics

12	12	0	1
Total attempts	Successful	Timed out	Runtime errors

Response time:	**00:11:42**
Average time taken between two compile attempts:	**00:00:59**
Average test case pass percentage per compile:	**18.2%**

Question 3 (Language: Python3)

When building a predictive model, feature selection is an important step. You wish to find the best set of features in a data set. To know whether a feature is informative or not, you need to build a linear regression model using one feature at a time and compare its F-statistic with the rest of the features.

Given a number **k**, identify the **k** best features in the data set provided to you.

The inputs to the function **fstatFeatures** shall be the input matrix **X**, response matrix **y**, and an integer **k**. The function must return an array of booleans wherein the i^{th} element is 1 if the i^{th} feature is among the top **k** features. Otherwise the ith element is 0.

The test cases tab illustrate some examples.

Final Code Submitted

```
1  # IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
2  # SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
3  # DEFINE ANY CLASS AND METHOD NEEDED
4  # THIS FUNCTION IS REQUIRED
5  #
6  # Parameters:  X: ndarray shape (n_samples,n_features)
7  #              y: ndarray shape(n_samples,1)
8  #              k: integer
9  #
10 # Returns:     X_new: ndarray shape (n_samples,k)
11 #
12 def fstatFeatures(X,y,k):
13      #INSERT YOUR CODE HERE
14      from sklearn.feature_selection import f_regression
```

Test Case Execution

Total score

100%

Warnings

There are no errors in the candidate's code.

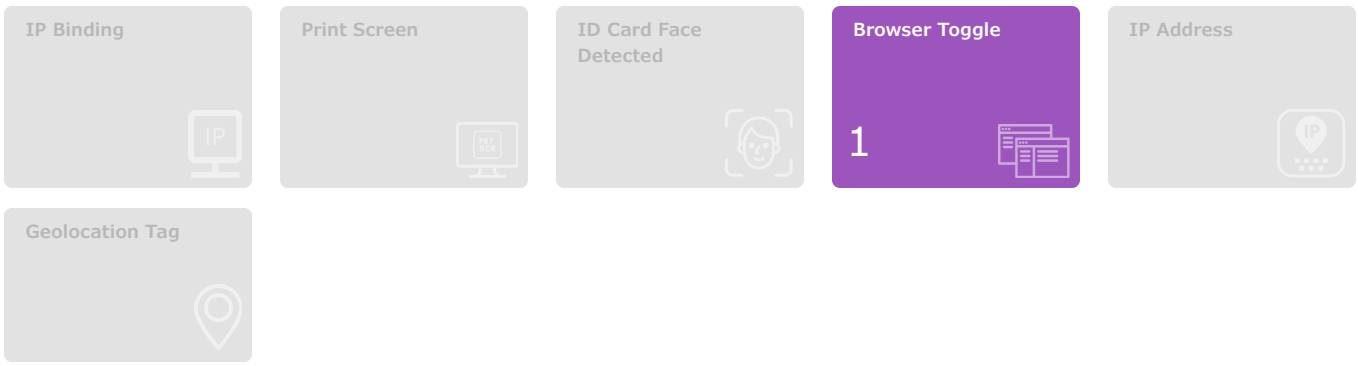
```
15      import numpy as np
16      f_statistic, p = f_regression(X,y)
17      n = (-f_statistic).argsort()[:k]
18      res = np.array([0]*X.shape[1])
19      for i in n:
20          res[i] = 1
21      return res
```

Compilation Statistics

18	17	0	2
Total attempts	Successful	Timed out	Runtime errors

Response time:	**00:17:38**
Average time taken between two compile attempts:	**00:00:59**
Average test case pass percentage per compile:	**13.8%**

3 | Proctoring



AI Proctoring Information

Print Screen: The number of times the candidate attempted to take a screenshot of the assessment screen using the “print screen” function on their device. Note: This impacts proctoring index.

ID Card Face Detected: Looks at the candidate images captured during the assessment and flags anywhere different people appear to be present. Snapshots are included in the report.

Browser Toggle: Either the proportion of time the candidate spent focused on a tab/window other than that of assessment screen (%), or the number of times the candidate toggled to another tab/window (count). Note: This impacts proctoring index.

IP Address: Confirms that the candidate took the assessment from the specified IP address(s).

Geolocation Tag: Detects whether the candidate attempted the assessment from a location beyond the distance set by the administrator.