

Aishwarya Gautam (AXG210221)  
Nishtha Shukla (NXS220018)


## Homework 4 - Report Machine Learning

### LINK OF GOOGLE COLAB FILE:

<https://colab.research.google.com/drive/1JqXrnE8ncPouPQ787Wf-vUQcbV1h13dN?usp=sharing>

### OUR APPROACH:

- We started by reading the *.data* and *.uai* file to see its contents.
- Reading through the *.data* file, we learned about the *query*, *evidence* and *hidden* variables that it contains.
- As mentioned in the class we had to change the representation of the data to get a better representation in order to solve the task i.e. featurize Markov network and hence we updated the *.data* file with *evidence* and *hidden* variables.
- As the values of the *hidden* variables were unknown, we took all possible combinations of its assignment and computed newly generated input.
- For simplicity we initially took the value of *hidden* variables as -1 and later changed it to all possible combinations of values it can take.
- First, we created a 2-D matrix, named *matrix*, combining all the values of the *query*, *hidden* and *evidence* from the *.data* file. The matrix looks as shown in *Image 1.1*.

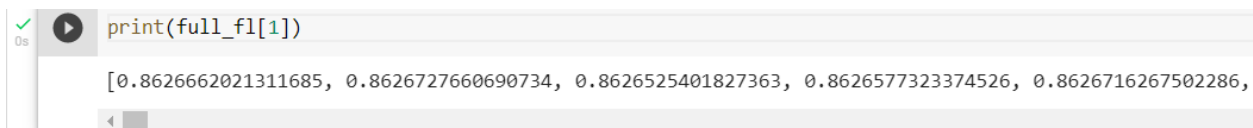


```
print(matrix[0])
```

```
[-1, 1, 1, 0, 0, 1, -1, 0, -1, -1, 1, 1, -1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, -1, -1, 1, 1, -1, 0, 0, 1, -1, 0, 0,
```

*Image 1.1*

- Now coming to the *.uai* file, we created another 2-D matrix, named *full\_fl*, containing the table values of the function table. The matrix looks as shown in *Image 1.2*.

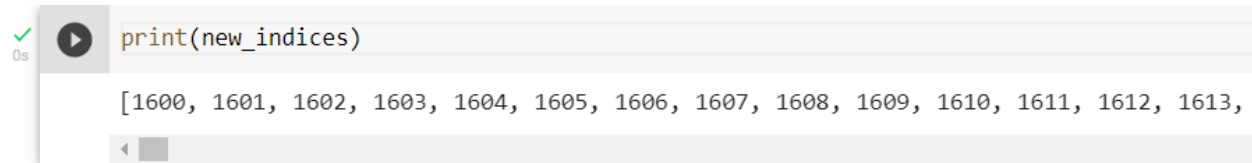


```
print(full_fl[1])
```

```
[0.8626662021311685, 0.8626727660690734, 0.8626525401827363, 0.8626577323374526, 0.8626716267502286,
```

*Image 1.2*

- Going to the function scope, we read the table values in the `.uai` file. We learnt the number of variables that function believed that and the index identity of each of the variables in the function's scope.
- Using the index identity and the function values, we computed the corresponding assignment to the *new evidence*. To access the function value, we used the function `binaryToDec`. This function helped us read the correct index position of the function value in the matrix.
- Next we calculated the indices of the *new evidence* and appended these to the new representation of the `.data` file. The new indices are as shown in *Image 1.3*.

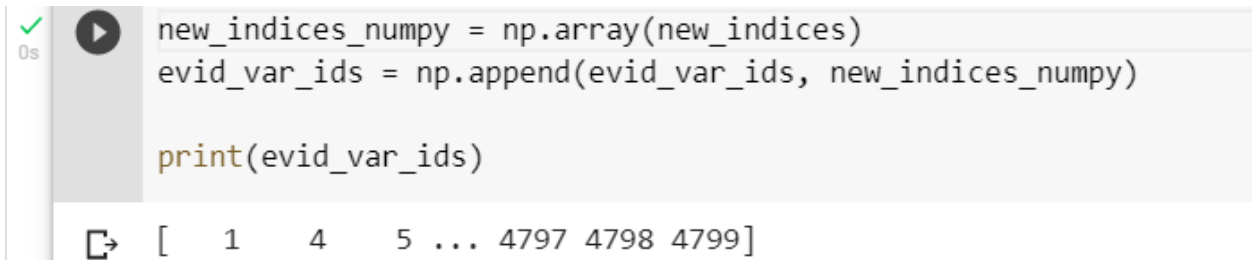


```
print(new_indices)
```

```
[1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613,
```

Image 1.3

- Now the `evid_var_ids` is appended with the `new_indices` as shown in *Image 1.4*.



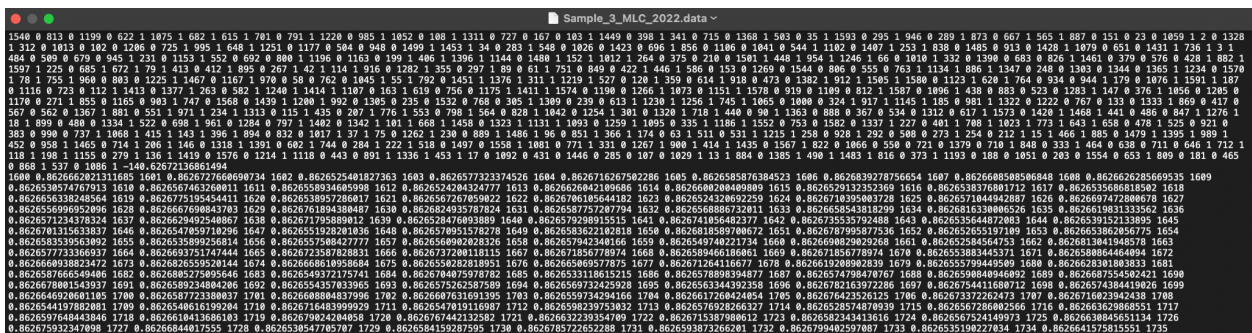
```
new_indices_numpy = np.array(new_indices)
evid_var_ids = np.append(evid_var_ids, new_indices_numpy)

print(evid_var_ids)
```

```
[ 1 4 5 ... 4797 4798 4799]
```

Image 1.4

- The new indices and the *new evidence* were added to the `.data` file. It is as shown in *Image 1.5*. Thus we achieved a new and better representation of the `.data` file.



```
1540 0.813 0.150 0.622 1.075 1.682 1.615 1.701 0.791 1.220 0.985 1.052 0.108 1.131 0.727 0.167 0.183 1.449 0.398 1.341 0.715 0.136 0.158 0.583 0.35 1.593 0.295 1.946 0.289 1.873 0.667 1.565 1.887 0.151 0.23 0.1059 1.2 0.1328
1.312 0.1013 0.182 0.1206 0.725 1.995 1.648 1.1251 0.1177 0.584 0.948 0.1499 1.1453 1.34 0.283 1.548 0.1026 0.1423 0.696 1.856 0.1106 0.1041 0.544 1.1102 0.1407 1.253 1.838 0.1485 0.913 0.1428 1.1079 0.651 0.1431 1.736 1.3 1
484 0.589 0.679 0.945 1.231 0.1153 1.552 0.692 0.886 1.1196 0.1163 0.199 1.406 1.1396 1.1144 0.1488 1.152 1.1012 1.264 0.175 0.210 0.581 1.440 1.954 1.1246 1.66 0.1018 1.132 0.1390 0.683 0.826 1.1461 0.179 0.576 0.428 1.882 1
1597 1.225 0.685 1.672 1.79 1.413 0.412 1.095 0.267 1.42 1.114 1.916 0.1282 1.355 0.297 1.89 0.61 1.751 0.849 0.22 1.446 1.986 0.153 0.1269 0.1544 0.006 0.355 0.763 1.114 1.886 1.1347 0.248 0.1383 0.1344 0.1365 1.1234 0.1570
1.78 1.755 1.968 0.883 0.1225 1.1467 0.1167 1.978 0.58 0.762 0.1845 1.55 1.792 0.1451 1.1376 1.311 1.1219 1.527 0.120 1.359 0.614 1.918 0.473 0.1382 1.912 1.1505 1.1588 0.1123 1.620 1.764 0.934 0.944 1.179 0.1076 1.1591 1.187
0.1116 0.723 0.112 1.1413 0.1377 1.263 0.582 1.1248 1.1414 1.1107 0.163 1.619 0.756 0.1175 1.1411 1.1574 0.1150 0.1266 1.1075 0.1151 1.1578 0.919 0.1109 0.812 1.1587 0.1096 1.438 0.883 0.523 0.1283 1.147 0.176 1.1056 0.1205 0
1170 0.271 1.555 0.1165 0.943 1.747 0.1568 0.1439 1.1280 1.992 0.1393 0.235 0.1532 0.768 0.365 1.1389 0.239 0.612 1.1220 1.1256 1.745 1.1065 0.1000 0.224 1.917 1.1145 1.185 0.981 1.1322 0.1222 0.767 0.333 0.1333 1.869 0.417 0
567 0.562 0.1367 1.881 0.551 1.971 1.234 1.1313 0.115 1.435 0.207 1.776 1.553 0.798 1.564 0.828 1.1042 0.1254 1.301 0.1320 1.718 1.440 0.90 1.1363 0.888 0.367 0.534 0.1312 0.617 1.1573 0.1420 1.1468 1.441 0.486 0.847 1.1276 1
18 1.899 0.489 0.1334 1.522 0.698 1.961 0.1284 0.797 1.1482 0.1342 1.101 1.668 1.1458 0.1523 1.1131 1.1093 0.1259 1.1095 0.335 1.1186 1.1552 0.753 0.1582 0.1337 1.227 0.401 1.788 1.1023 1.773 1.643 1.658 0.478 1.525 0.921 0
383 0.900 0.737 1.1068 1.415 1.142 1.396 1.894 0.832 0.1017 1.37 1.75 0.1262 1.220 0.889 1.1486 1.86 0.851 1.366 1.174 0.63 1.511 0.531 1.1215 1.258 0.928 1.292 0.588 0.272 1.254 0.212 1.15 1.466 1.885 0.1479 1.1395 1.989 1
452 0.958 1.1465 0.714 1.206 1.146 0.1318 1.1391 0.682 1.744 0.284 1.222 1.518 0.1497 0.1558 1.1081 0.771 1.331 0.1267 1.908 1.414 1.1435 0.1567 1.822 0.1866 0.550 0.721 1.379 0.710 1.848 0.333 1.464 0.638 0.711 0.646 1.712 1
118 1.198 1.1155 0.279 1.136 1.1419 0.1576 0.1214 1.1118 0.443 0.891 1.1336 1.453 1.17 0.1092 0.431 0.1446 0.285 0.107 0.1029 1.13 1.884 0.1385 1.490 1.1483 1.816 0.373 1.1193 0.188 0.203 0.1554 0.653 1.809 0.181 0.465
0.668 1.537 0.1085 1.440 0.5672 1.0683 0.944
1600 0.8626626021311683 1601 0.8626727606989734 1602 0.86265254081827363 1603 0.8626577323374526 1604 0.8626716207562286 1605 0.8626585876384523 1606 0.8626839278756654 1607 0.862668508508506848 1608 0.8626626285669535 1609
0.8626538574767913 1610 0.8626567463268011 1611 0.862658934685998 1612 0.8626524284324777 1613 0.8626626042109686 1614 0.86266484204409809 1615 0.8626529132352369 1616 0.8626533686818582 1618
0.8626663102484584 1619 0.862677395454411 1620 0.8626538957286017 1621 0.8626367210785022 1622 0.862678618644182 1623 0.862654328602259 1624 0.862671839583728 1625 0.86265718444042087 1626 0.862669747080670 1627
0.862656996952066 1628 0.8626667098843703 1629 0.8626761894388487 1630 0.8626824935787824 1631 0.8626587757287794 1632 0.8626588886732811 1633 0.8626658543818299 1634 0.8626816338086626 1635 0.8626619881333562 1636
0.8626571234378324 1637 0.8626629492540867 1638 0.862671795889812 1639 0.8626528476893889 1640 0.8626579289915515 1641 0.8626741856482377 1642 0.8626735535792488 1643 0.8626535644872883 1644 0.8626539152133895 1645
0.8626781315633807 1646 0.8626547859710296 1647 0.8626551928291836 1648 0.862673095378278 1649 0.8626581622182810 1650 0.8626818589780672 1651 0.862678795877836 1652 0.862652655197189 1653 0.8626653362865675 1654
0.862653359583692 1655 0.8626535899256814 1656 0.8626557508427777 1657 0.86265689902828326 1658 0.86265742348166 1659 0.8626549740221734 1660 0.8626698829029268 1661 0.862652584564753 1662 0.8626813841480578 1663
0.86265773336937 1664 0.8626693751747444 1665 0.8626723587828831 1666 0.8626737280118115 1667 0.862671856778974 1668 0.8626589466186061 1669 0.862671856778974 1670 0.8626553883443371 1671 0.862658864464894 1672
0.862666938822472 1673 0.862682655920144 1674 0.8626668818058684 1675 0.8626550232818931 1676 0.862663809577875 1677 0.862671264116577 1678 0.862651928092839 1679 0.8626553790440590 1680 0.862668381883833 1681
0.86265957665649486 1682 0.862688575995646 1683 0.8626549372175741 1684 0.8626784075978782 1685 0.8626533118615215 1686 0.8626578898394877 1687 0.8626574798470767 1688 0.862659804094692 1689 0.862687554582421 1690
0.8626678801543937 1691 0.8626589234884206 1692 0.8626554357833965 1693 0.8626575262587589 1694 0.8626569732425928 1695 0.8626563344392358 1696 0.8626782163972286 1697 0.8626754411680712 1698 0.8626574384418026 1699
0.86266469286861185 1700 0.8626567723380837 1701 0.8626568884833996 1702 0.8626697631691392 1703 0.8626559734284166 1704 0.8626617280424864 1705 0.862676423526125 1706 0.86267372262473 1707 0.8626718823942438 1708
0.862654197882881 1709 0.8626540616199284 1710 0.8626716483999929 1711 0.8626547910116897 1712 0.8626598239753832 1713 0.8626576928266327 1714 0.8626528574878939 1715 0.8626567286002566 1716 0.8626638845651134 1717
0.8626597484443846 1718 0.8626610413868103 1719 0.862679824240458 1720 0.862676744213282 1721 0.8626632239354709 1722 0.8626715387980612 1723 0.8626582343413616 1724 0.86265657524149973 1725 0.8626535198227034 1734 0.8626641575815511 1735
0.862675932347890 1727 0.86266848417555 1728 0.862653854785707 1729 0.8626584159287595 1730 0.862678572652288 1731 0.8626593873266201 1732 0.8626799402597087 1733 0.8626535198227034 1734 0.8626641575815511 1735
```

Image 1.5

- The 2-D matrix *evid\_assignments* is updated subsequently with the new values. This is as shown in *Image 1.6*.

```
print(evid_assignments)

[[1.         0.         1.         ... 0.86264782 0.86264782 0.86264782]
 [0.         0.         0.         ... 0.86264782 0.86264782 0.86264782]
 [1.         0.         0.         ... 0.86264782 0.86264782 0.86264782]
 ...
 [1.         0.         0.         ... 0.86264782 0.86264782 0.86264782]
 [0.         0.         1.         ... 0.86264782 0.86264782 0.86264782]
 [1.         0.         0.         ... 0.86264782 0.86264782 0.86264782]]
```

*Image 1.6*

- The size of the input (i.e. *evidence* variables) is increased as we added new function values using the *Markov Network* and enhanced the representation of the file.
- Later this new representation is split into test and train dataset and sent to the classifier for the task of *Multi-Label Classification*.

## **EXPERIMENTAL RESULTS:**

- As we proceeded to implement the cell [75] which includes the calculation of *lprob\_true*, *lprob\_pred* and *lprob\_trivial*, the session on colab crashes for some unknown reason, as shown in *Image 1.7*, and its corresponding logs as shown in *Image 1.8*

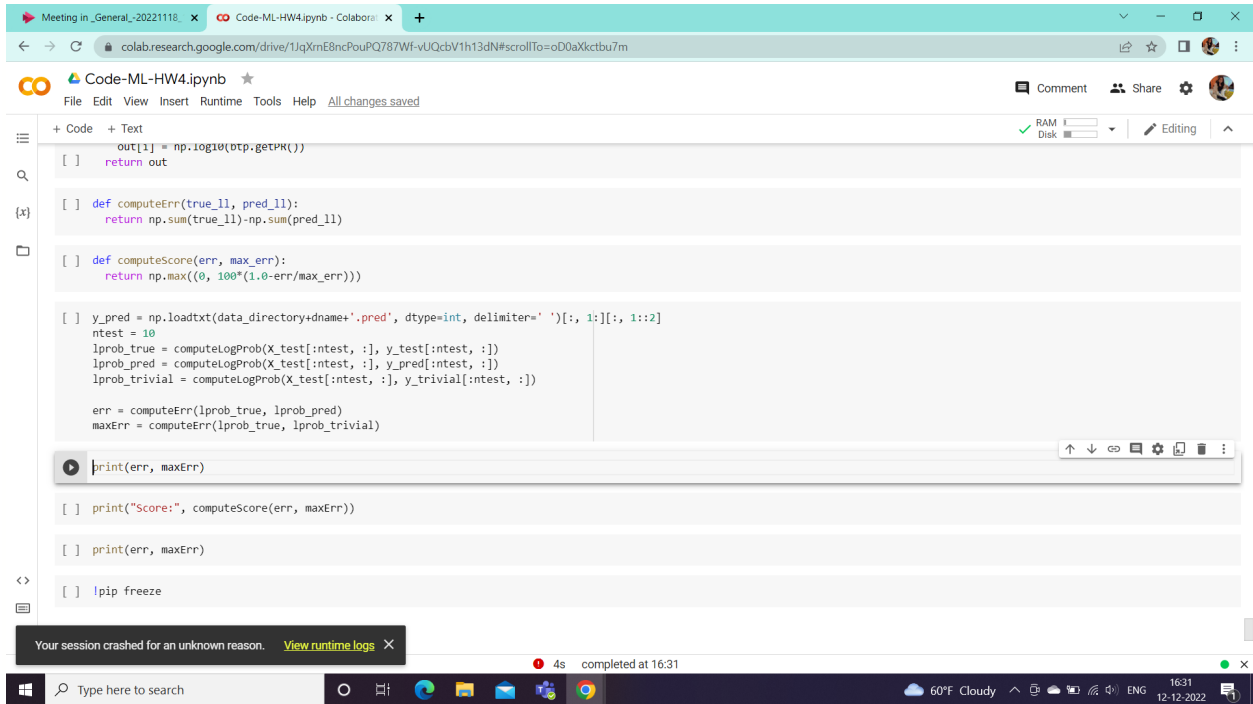


Image 1.7

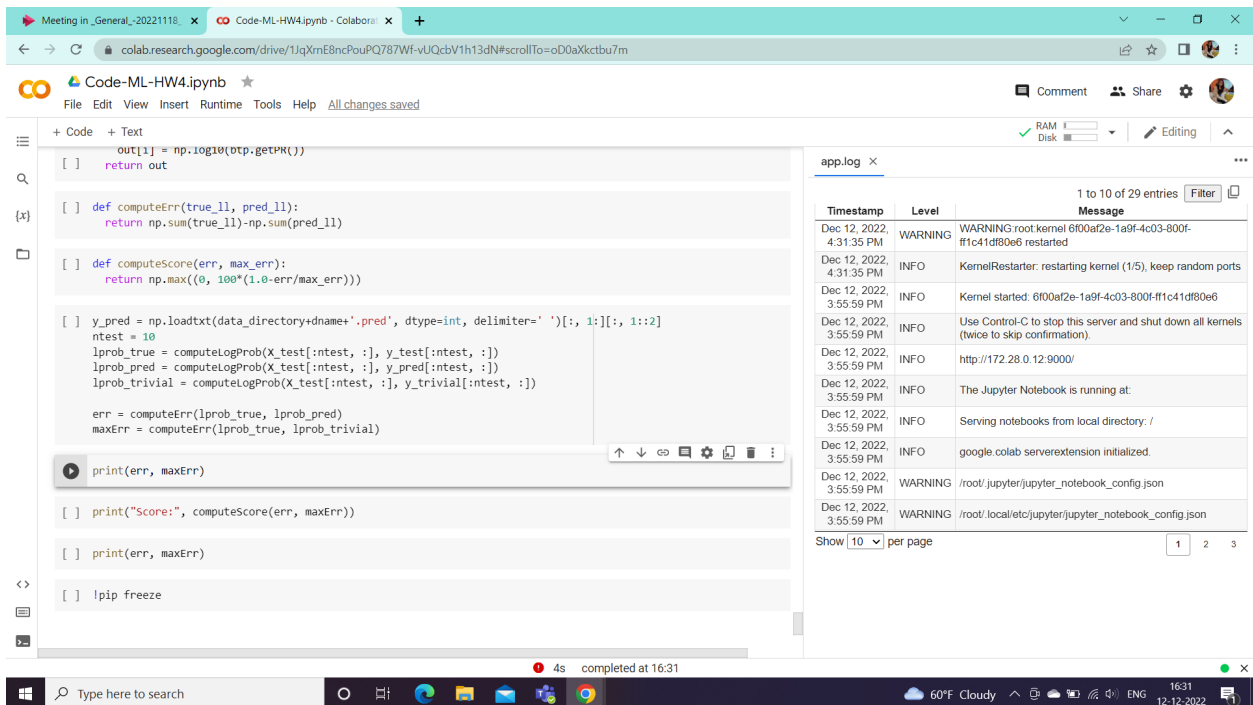


Image 1.8

- To debug what is causing the crash, we thought about changing the model that was used for training. Instead of the *Logistic Regression*, we implemented the models *KNN*, and *Decision Tree*. It is as shown in *Image 1.9* and *Image 1.10*.

```

▶ param_grid = {
    'max_depth' : [2,4,6,8,10],
    'min_samples_leaf' : [4,8,16,32,64,128,256,512],
    'criterion' : ["gini", "entropy"]
}
clf = tree.DecisionTreeClassifier()
clf_RandomGridSearch = RandomizedSearchCV(estimator = clf, param_distributions = param_grid, cv =10, verbose =5 , n_jobs =2 )
clf_RandomGridSearch.fit(X_train, y_train)

↳ Fitting 10 folds for each of 10 candidates, totalling 100 fits
RandomizedSearchCV(cv=10, estimator=DecisionTreeClassifier(), n_jobs=2,
    param_distributions={'criterion': ['gini', 'entropy'],
        'max_depth': [2, 4, 6, 8, 10],
        'min_samples_leaf': [4, 8, 16, 32, 64,
            128, 256, 512]},
    verbose=5)

```

**Image 1.9**

```

[ ] clf_RandomGridSearch.best_params_

{'min_samples_leaf': 32, 'max_depth': 2, 'criterion': 'gini'}

```

**Image 1.10**

- Another approach we tried for debugging the cause of the session crash was reducing the size of the dataset. On doing this, the experimental results are as follows:

- 'Sample\_1\_MLC\_2022'

```

✓ [63] print(err, maxErr)
0s
0.00026953961059916764 44.44903408518121

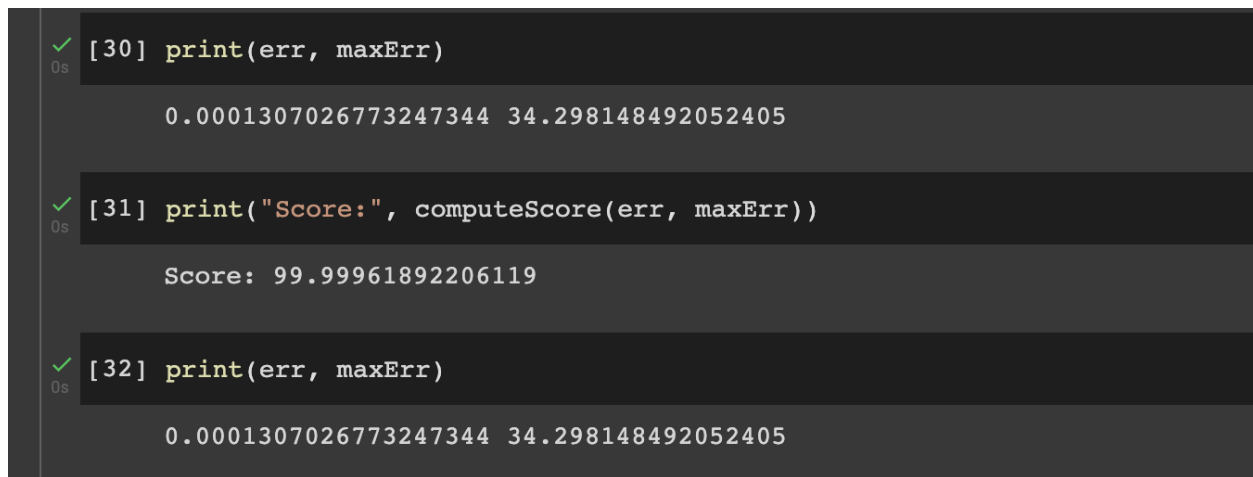
✓ [64] print("Score:", computeScore(err, maxErr))
0s
Score: 99.99939359849736

✓ [65] print(err, maxErr)
0s
0.00026953961059916764 44.44903408518121

```

**Image 1.11**

- 'Sample\_2\_MLC\_2022'

A terminal window showing three code blocks. Each block starts with a green checkmark and '0s' in the left margin. The first block contains '[30] print(err, maxErr)' followed by the output '0.0001307026773247344 34.298148492052405'. The second block contains '[31] print("Score:", computeScore(err, maxErr))' followed by the output 'Score: 99.99961892206119'. The third block contains '[32] print(err, maxErr)' followed by the output '0.0001307026773247344 34.298148492052405'.

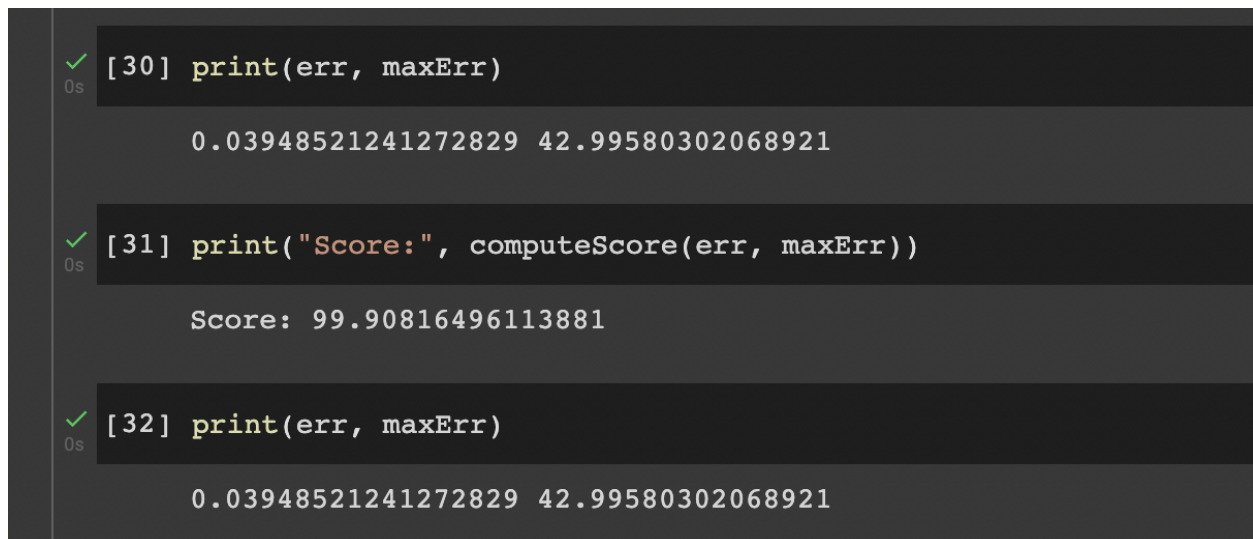
```
[30] print(err, maxErr)
0.0001307026773247344 34.298148492052405

[31] print("Score:", computeScore(err, maxErr))
Score: 99.99961892206119

[32] print(err, maxErr)
0.0001307026773247344 34.298148492052405
```

*Image 1.12*

- 'Sample\_3\_MLC\_2022'

A terminal window showing three code blocks. Each block starts with a green checkmark and '0s' in the left margin. The first block contains '[30] print(err, maxErr)' followed by the output '0.03948521241272829 42.99580302068921'. The second block contains '[31] print("Score:", computeScore(err, maxErr))' followed by the output 'Score: 99.90816496113881'. The third block contains '[32] print(err, maxErr)' followed by the output '0.03948521241272829 42.99580302068921'.

```
[30] print(err, maxErr)
0.03948521241272829 42.99580302068921

[31] print("Score:", computeScore(err, maxErr))
Score: 99.90816496113881

[32] print(err, maxErr)
0.03948521241272829 42.99580302068921
```

*Image 1.13*

- But this approach of reducing the size of the dataset did not work on 'Sample\_4\_MLC\_2022' and hence could not compute its results.