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*.

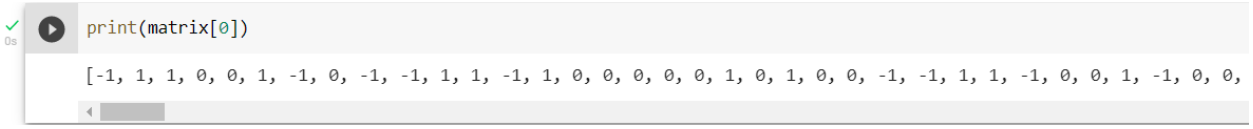

Image 1.1

- 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.*

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 belonged to 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*.

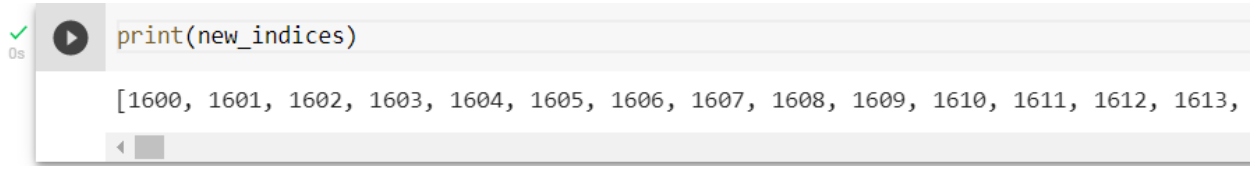

Image 1.3

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

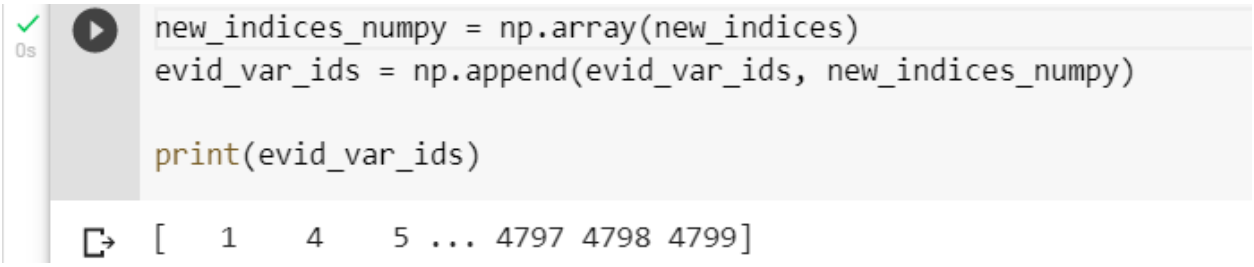

Image 1.4

- 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.

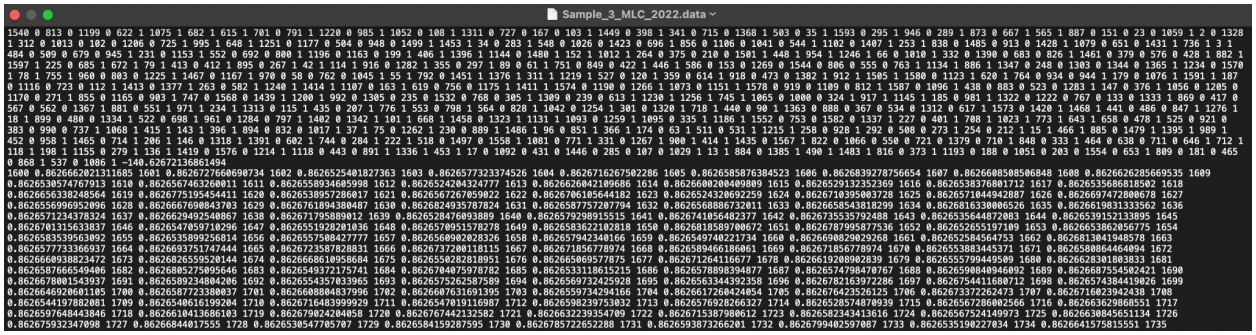

Image 1.5

- 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*

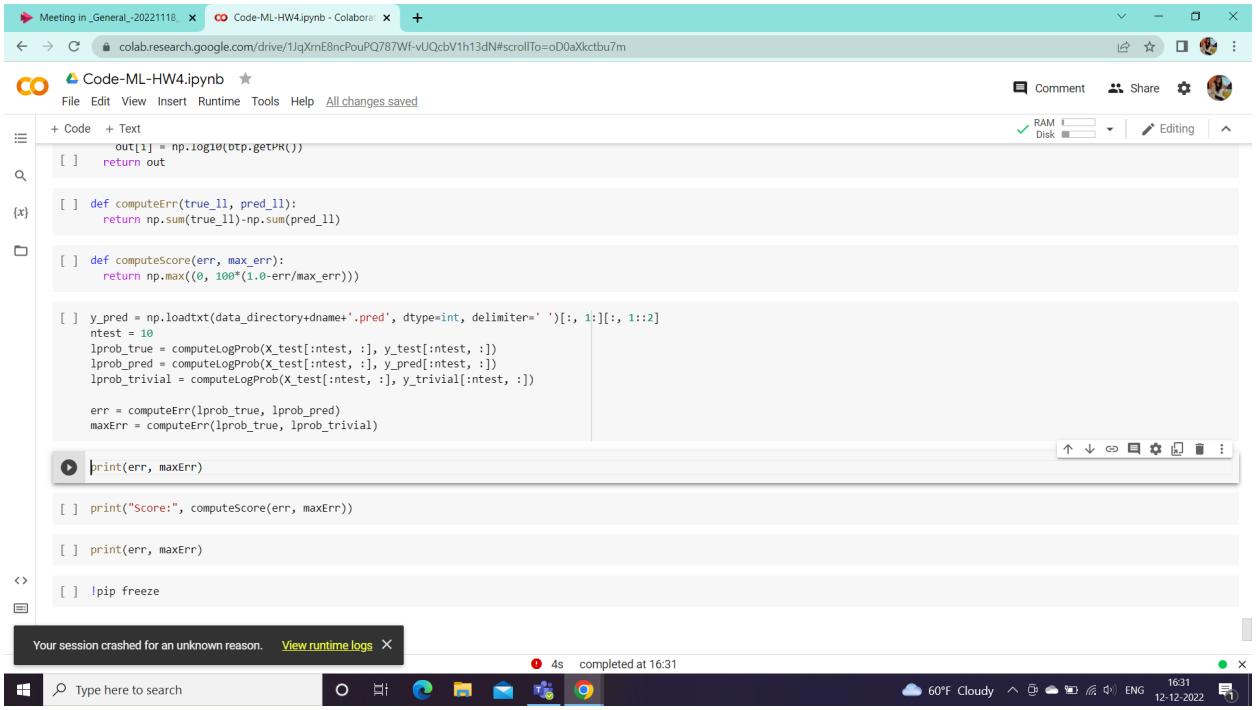

Image 1.7

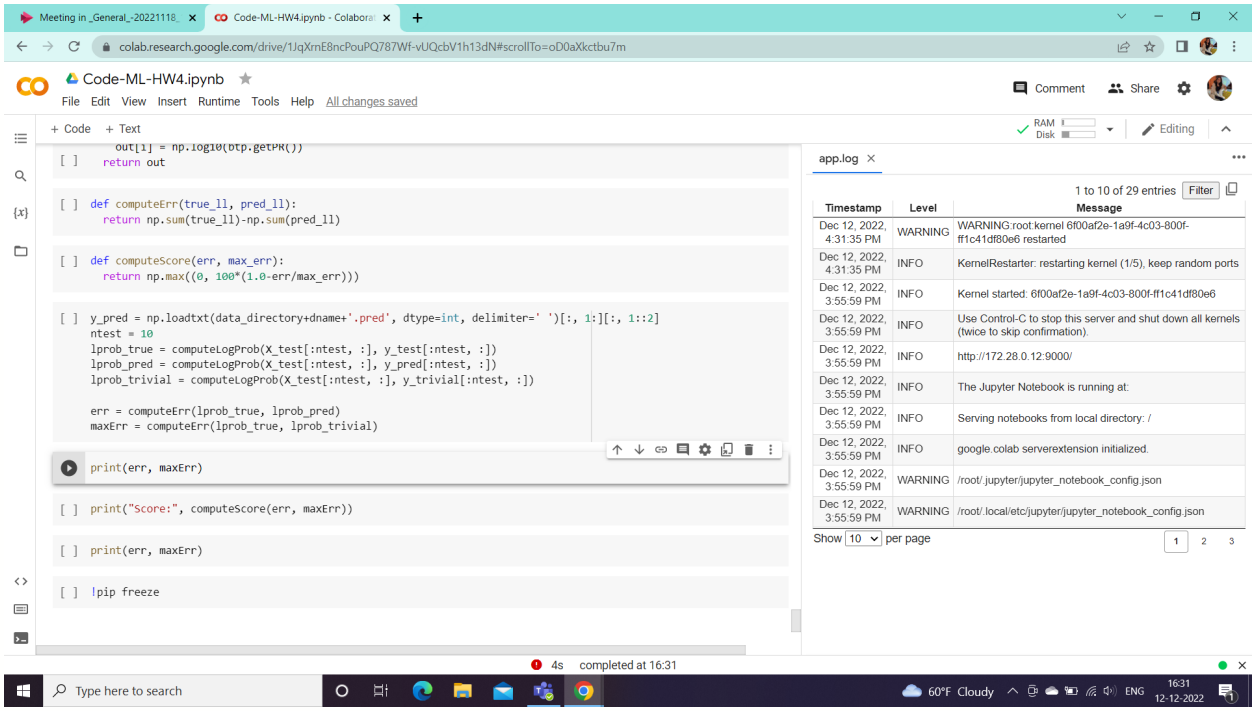

Image 1.8

- 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)
```

```
0.00026953961059916764 44.44903408518121
```

```
[64] print("Score:", computeScore(err, maxErr))
```

```
Score: 99.99939359849736
```

```
[65] print(err, maxErr)
```

```
0.00026953961059916764 44.44903408518121
```

Image 1.11

- 'Sample_2_MLC_2022'

```
[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'

```
[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.