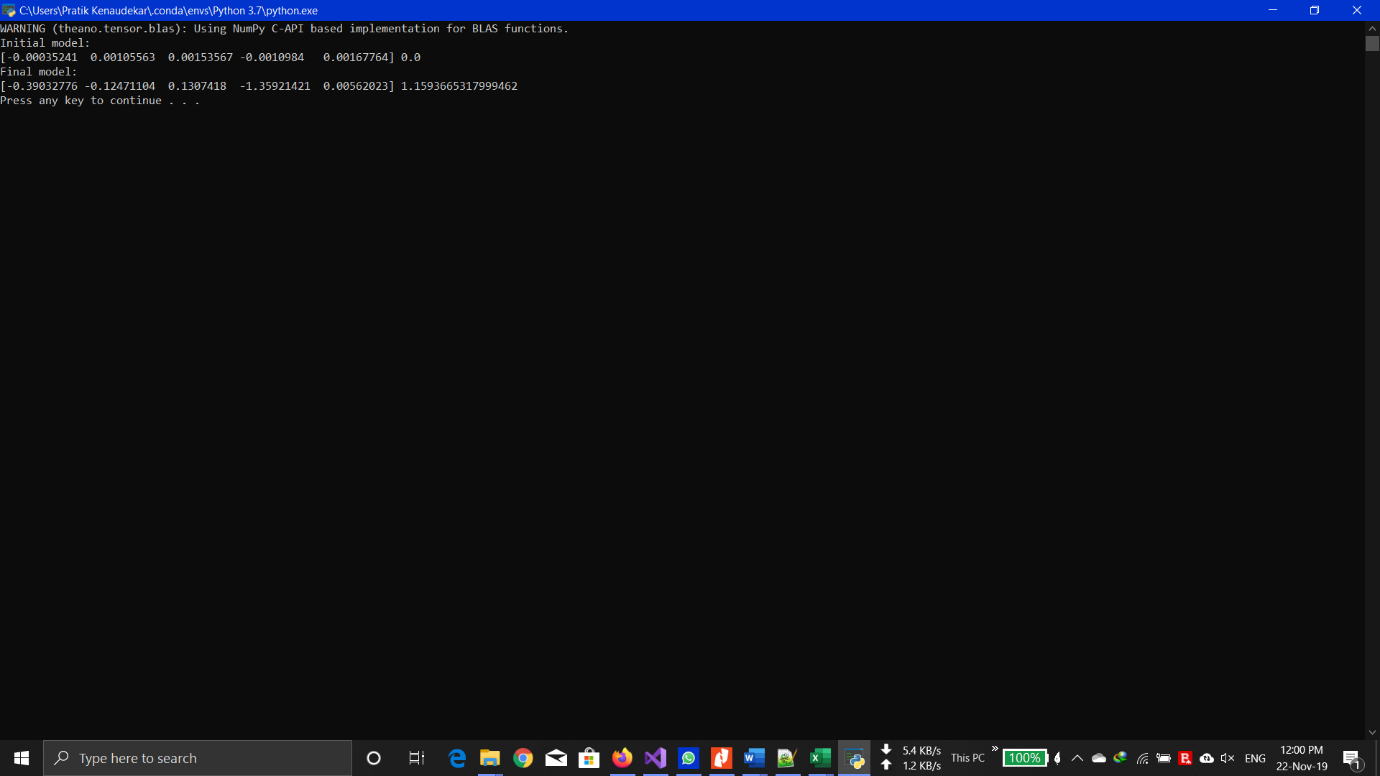
**Name: Pratik G Kenaudekar**

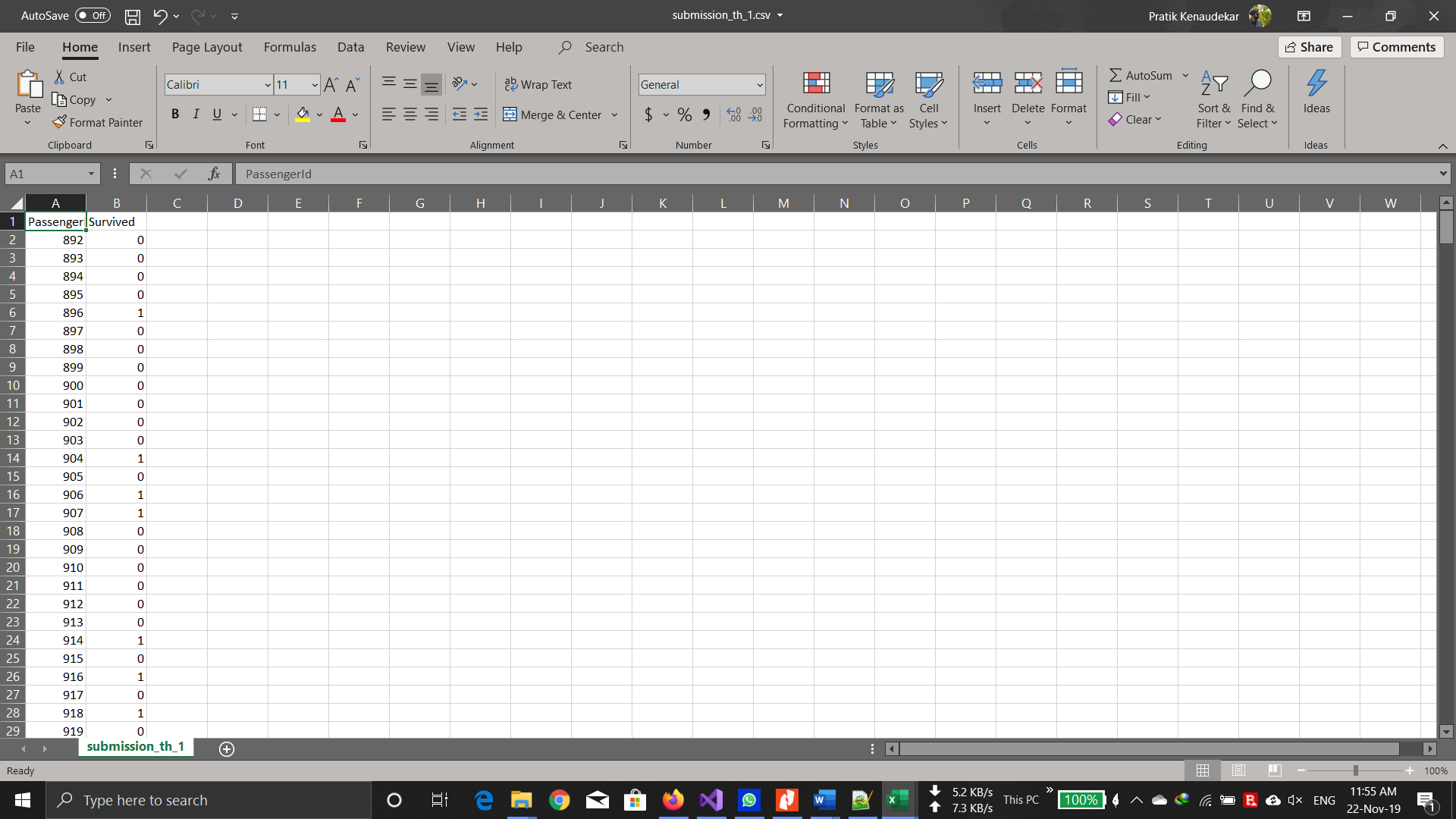
**Roll No: 1711**

1. **Data source and justification for the use of data for the model used.**

**Data source**: Kaggle Titanic Competition for MTLData - Oct.-22,2014

**Justification**: Since the model used in the program is based on Binary Logistical Regression, it has only two possible outcome and hence the input files are of **.npy** extension since these are a simple binary data format. It stores the type, shape and endianness information in a header, which is followed by a flat binary data field. This crate offers a simple, mostly type-safe way to read and write \*.npy files. Files are handled using iterators, so they don't need to fit in memory.

1. **Explanation of main functions used in case of python code or steps followed in case of weka tool.**
2. #Import required modules
3. import numpy
4. import theano
5. import theano.tensor as T
6. rng = numpy.random
7. #load trainData and targets, get Tuple D, define number of training steps
8. train\_data=numpy.load("titanic\_train.npy")
9. train\_supervision=numpy.load("train\_targets.npy")
10. D=(train\_data,train\_supervision)
11. ncols=train\_data.shape[1]
12. feats=ncols
13. training\_steps = 40000
14. # Declare Theano symbolic variables
15. x = T.matrix("x")
16. y = T.vector("y")
17. w = theano.shared(rng.randn(feats), name="w")
18. b = theano.shared(0., name="b")
19. scale = theano.function([], w, updates=[(w, w\*0.001)]) #theano.function(inputs=[x,y],list of input variables
20. # outputs=..., what values to be returned
21. # updates=..., “state” values to be modified
22. # givens=..., substitutions to the graph)
23. print ("Initial model:")
24. scale()
25. print (w.get\_value(), b.get\_value())
26. # Construct Theano expression graph
27. p\_1 = 1 / (1 + T.exp(-T.dot(x, w) - b)) # Probability that target = 1
28. prediction = p\_1 > 0.5 # The prediction thresholded
29. xent = -y \* T.log(p\_1) - (1-y) \* T.log(1-p\_1) # Cross-entropy loss function
30. cost = xent.mean() + 0.01 \* (w \*\* 2).sum() # The cost to minimize
31. gw, gb = T.grad(cost, [w, b]) # Compute the gradient of the cost
33. # Compile
34. train = theano.function(inputs=[x,y],outputs=[prediction, xent],updates=((w, w - 0.001 \* gw), (b, b - 0.001 \* gb)))
35. predict = theano.function(inputs=[x], outputs=prediction)
36. # Train
37. for i in range(training\_steps):
38. pred, err = train(D[0], D[1])
39. print ("Final model:")
40. print (w.get\_value(), b.get\_value())
41. #Load Test Data
42. testData=numpy.load("titanic\_test.npy")
43. #Predict Survival
44. testPassengerStartIndex=892
45. outputs = [[x[0]+testPassengerStartIndex,x[1]] for x in enumerate(predict(testData))]
46. #Prepare file for submission
47. numpy.savetxt("submission\_th\_1.csv", outputs, delimiter=',', fmt='%d,%d', header='PassengerId,Survived', comments = '')
48. **Screen shot of output**



As the model used in the program is a Binary Logistical Regression in which the output gives you the probability. In our case we are trying to find if the passenger had survived or not. This is given by probability ranging between 0 and 1. Here a 0 means didn’t survive and 1 means survived.