BUSM131: THE MASTERCLASS IN BUSINESS ANALYTICS

How Big Data Is used In The Media – BBC

/

Adjusting the Industry Policy by Analysing the U.S. EPA Toxic Release Inventory Data in Artificial Neural Networks

Name : Jui-Ting Hu (Calista)

Student ID: 170376051

MSc Business Analytics

Content

Adjusting the Industry Policy by Analysing the U.S. EPA Toxic Release Inventory Data in Artificial Neural Networks

1. Introduction

In recent year, a large wave of artificial neural networks has been sweeping through machine learning. They are widely used in a variety of business fields such as Google, Apple and YouTube. In this project, it would stand on artificial neural networks to establish a business case. This case is devoted to U.S Environmental Protection Agency Toxic Release Inventory data, which records every year. We assume that although this organisation specialise in setting out the rules on toxic release, with the industries change, it seems that the rules might not inapplicability. In overview, we expect that this project can be used to support the organisation to modify the environmental policies on the industry sector.

The paper is structured as follows. In the purpose of the project section, we will introduce the choice of dataset and the concept of the business case. In the process of the project, the work is divided into two parts which are cleaning the dataset and building the model. In the result of the project, after running the model, we obtain and investigate the result to try interpreting.

2. Purpose of the Project

In this project, there are two primary purposes. One is that using the dataset which is chosen to achieve artificial neural networks. Another is that giving a meaningful business case and business insights based on this model. However, before discussing these primary points in detail, artificial neural networks which are the main machine learning tool should be introduced and understood. Artificial neural networks (ANNs), as the core of deep learning, which are the adaptive, versatile, powerful and flexible model, usually are applied in prediction, classification or clustering. The structure of ANNs consists of an input layer, multiple hidden layers, an output layer and activation functions such as ReLU and Softmax (Figure1). In general, ANNs are particularly useful for dealing with large and complicated machine learning tasks. For example, Google Images uses it to classify billions of image (Géron, 2017).

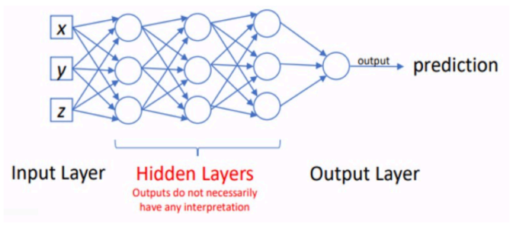


Figure1: The structure of ANNs

2.1 Identifying the dataset

Therefore, in order to achieve two purposes which mentioned above, firstly, there is no doubt that choosing a suitable dataset is the primary goal. Due to the features of ANNs, this project needs huge enough dataset to demonstrate their unique advantages. We were looking for a dataset at least greater than 10,000 data entries and more than 20 variables. This range narrowed our search. There are three datasets which met the requirements from Kaggle: National Football League (NFL) Play-by-Play data, Lending Club Loan data and U.S. Environmental Protection Agency (EPA) Toxic Release Inventory data. Although NFL data includes great detail information of play such as situation, players, results and win probability, it is more difficult to make a good business case out of it. And then, it seems that loan data which includes the current loan status and latest payment information can be used. However, we considered the U.S. EPA Toxic Release Inventory data has better usability, which records over 53,000 facilities' toxic release information from 1987 to 2016, including the facilities' location, the categories and quantities of chemicals released, the approaches of chemicals managed. There are several reasons for this choice. According to Kaggle.com (2017), this dataset which has 110 variables and around 5 million data entries definitely satisfies the basic requirement. Because it compiled by the U.S. EPA, which is an independent agency under the United States federal government, compared with other datasets, it is relatively reliable, robust and accurate. Noteworthy, about data file format and content such as the definition of variables, zeroes in the data and so on, it provides the extremely clear and complete description in the appendix documentation and the official website. Moreover, in our group, project partners who have chemistry and biology background can apply their knowledge and get more insight into this topic. These make the Toxic Release Inventory data become the best option for the first purpose.

2.2 Developing the business case

Secondly, after determining the dataset, the business case should be built for the second purpose of the study. The EPA, as an independent government organisation, is established in December 1970 in order to “protect human health and the environment”. It has the authority to set national standards on a wide range of environmental issues by formulating and enforcing regulations. If tribes or companies fail to reach the standards, the EPA will assist them (U.S. EPA, n.d.). U.S. EPA (2015) claims that the toxic release inventory program is meant to collect, manage and promote this dataset and provides a resource for the citizens. Through analysing this dataset, it helps track the trend of annual toxic released; compare the number of toxic released among industry sectors; identify priorities for facilities. Based on this information about the EPA, our key idea of this project is that following the goal of EPA’s toxic release inventory program to help EPA to adjust the regulation of industry sector through using ANNs machine learning model to analyse the toxic release inventory data.

We assume that the same industry sector is usually doing similar business activities. This leads to they would have many similarities like the location of factories, the number of toxic emissions, the category of release, the method of waste management. However, as time goes on, every industry or factory will appear some slight or dramatic changes. For example, a factory or industry, it initially emitted Dioxin and only did little treatment on wastewater. After a few years, because it might not manufacture the product which releases Dioxin or improve the appliance, it reduces the emission and turns into more friendly to the environment. Conversely, it is possible that the factory was harmless formerly, then it releases numerous toxic afterwards. Besides, it is evident that the EPA adopt different policies on a sector-by-sector basis (U.S. EPA, n.d.). Interestingly, If this factory or industry changes significantly, whether it is getting better or worse, it might be argued that the original policies which are established by EPA are inappropriate.

Hence, we believe that we can use this historical toxic release dataset which is sorted by year to create an industry classification model. We would use the first 80 per cent of data to train and the last 20 per cent of data to test. This means that those past data would build ANNs model with input, toxic release inventory of each facility, and output, the industry sector of each facility. And then, those recent years’ data is used to test on this model whether their outputs and industry labels are the same. Hence, we believe that we can use this historical toxic release dataset which is sorted by year to create an industry classification model. We would use the first 80 per cent of data to train and the last 20 per cent of data to test. This means that those past data would build ANNs model with input, toxic release inventory of each facility, and output, the industry sector of each facility. And then, those recent years data is used to test on this

model whether their outputs and industry labels are the same. According to this result from the model, we expected that if output and industry label of the facility is the same, it means that it is not much different from the past facilities which come from the same industry. In contrast, if it shows a different outcome, it has different from past facilities. The EPA should consider that does the difference only take place on one facility or the whole industry, and decide to put more resources or even adjust the policies for it.

3. Process of the Project

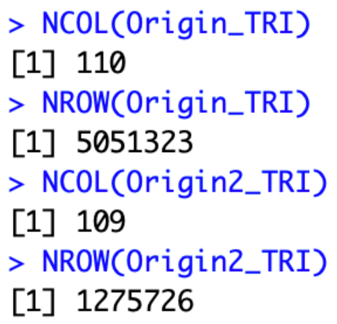
Following the business case that is proposed, there are two times modelling works. The first work is for the presentation which includes several errors. Three main issues are that deleting too much data causes the dataset we used not enough to represent initial data; creating the dummy variables way is misleading; including the INDUSTRY_SECTOR_CODE, which is the same as industry sector, predicts the model. After learning these, the second work is revised from the start. Thus, this project would emphasise on the process and result of the second work. In this part, it is divided into two parts: data description and modelling method.

3.1 Data Description

EPA.gov (2016) states that due to the toxic release program, the facilities should report relevant data to the EPA each year. These basic data files of the year can be easily accessed from the official webpage. The dataset which is used in this project is merged the toxic release inventory basic data files of the years between 1987 and 2016 into a single large CSV from Kaggle. All of the data types, definition, unit, content are illustrated in the official documentation.

In the beginning, although the content of this dataset is remarkably complete, it is so big that it cannot be opened and viewed in Excel. Unfortunately, when we load it in Python, it also cannot work. It continuously shows the error that says some rows which should have 110 columns have 111 columns. We try to use skiprows function while reading, but too many rows need to be handled. Therefore, we decide to load the data and address format errors in R. After importing the data, there are 110 variables and 5,051,323 observations. At the same time, it can be seen that some values of variables are discrepant. Take FEDERAL_FACILITY that is one of the variables as an example, it should only include Yes and No, but it contains the numerical values which look like the longitude before it in the variable fields. In our view, the extra column of the rows is the effect of the comma error in the data file.

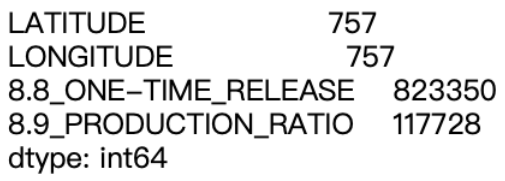
Therefore, we use FEDERAL_FACILITY as a filter to remove the format error rows which have an extra 111 column. Another format error is that the Unnamed:109 variable which does not exist in the official EPA data file should be removed. Now, it leaves with 109 columns and 1,275,726 rows which still a lot (Figure2).



```
> NCOL(Origin_TRI)
[1] 110
> NROW(Origin_TRI)
[1] 5051323
> NCOL(Origin2_TRI)
[1] 109
> NROW(Origin2_TRI)
[1] 1275726
```

Figure 2 : The number of columns and rows in dataset.

Next, we successfully re-import in Python and find a large number of missing values which are required to tackle in the dataset. Firstly, it is necessary to remove 1296 missing values in the INDUSTRY_SECTOR which is the output. Secondly, the variables that involve not related to the business case or excessive null values also would be deleted. The rest few of the missing values are that we need to fill in (Figure 3). Thirdly, because according to EPA.gov (2016), there are three reasons why the numeric data have missing values, including amounts are below the threshold, toxic is not possibly release or facilities do not report, we think that in terms of release fields should replace NA to 0. However, we use mean imputation in other fields which are not release variables.



```
LATITUDE                757
LONGITUDE                757
8.8_ONE-TIME_RELEASE    823350
8.9_PRODUCTION_RATIO    117728
dtype: int64
```

Figure 3 : The rest few of NA

Finally, we should create dummy variables for nominal data. There are two methods, binary variable and one-hot encoding. One the one hand, if the variable only has two classes, using binary dummies, 0 and 1, changes the values of classes. On the other hand, if the variable includes more than two categories, through one - hot encoding function, each category would be converted to new columns which all are binary (Hacker Noon, 2017).

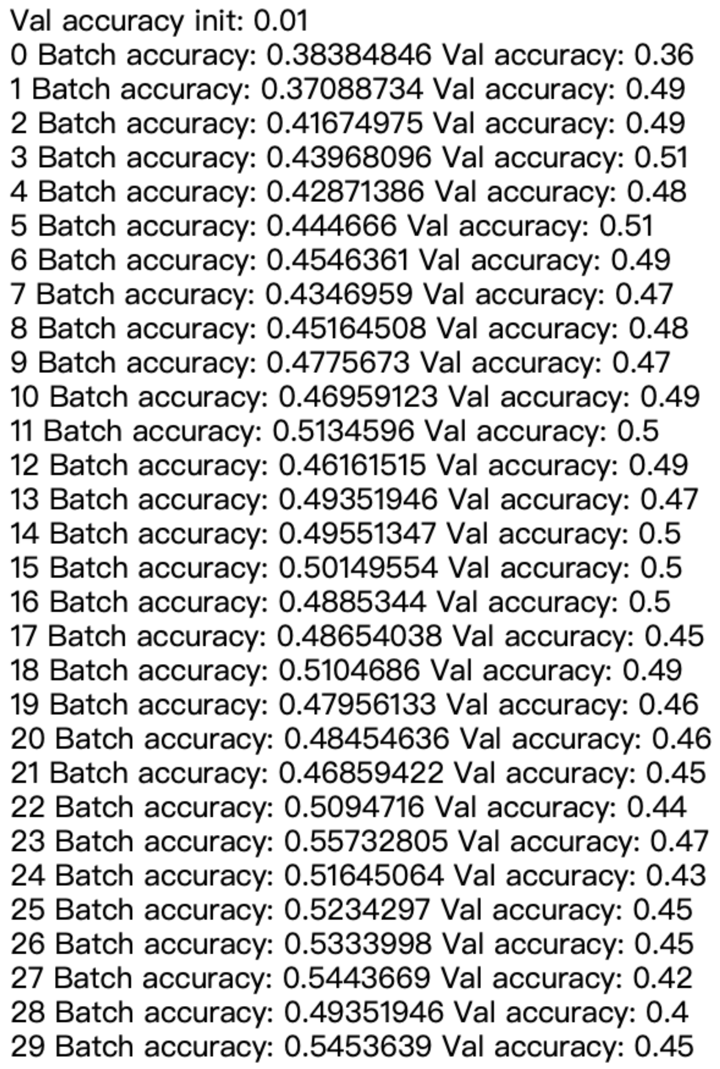
3.2 Modelling Method

Following the process mentioned above, it creates a new dataset with 1,274,430 rows and 764 columns. Before building the ANNs to train this dataset, we add the below functions in order to obtain a more reasonable result. Owing to the dataset too large to run, we use the random.permutation function to randomly generate a new dataset with only 20 per cent of data to continue the work. This makes the dataset reduces from 1,274,430 to 254,886 data entries which would be used to build the ANNs model. At the same time, the dataset should be divided into training and test set. Theoretically, a test set which is created usually randomly consists of 20 per cent of the dataset (Géron, 2017). However, based on our business case, before dividing, the data must be arranged in ascending order. In this case, when the dataset is split into the training set and test set, it would utilize the past data to train the model and the recent years' data to test the model as we wish. Now, the training set which has 80% of data is from 1993 to 2013 and the test set that has 20% of data is from 2013 to 2016. In addition, it can be argued that our features of the dataset are seriously not on the same scaling is because of different units, dummies variables or missing values filled. Medium (2018) points out that the particularly huge variable's variance might allocate the result, then causes other variables are ignored. Through using StandardScaler function which removes the mean and scales to unit variance to standardize features can help us avoid that happen (Scikit-learn.org, n.d.). Moreover, we add the more advanced optimizer function, AdagradOptimizer instead of GradientDescentOptimizer which can automatically adapt the learning rate to the parameters to minimize the cost function (Sebastian Ruder, 2016). The main advantage of it is that we almost do not need to manually adjust the learning rate which can be default as 0.01.

Finally, we can start to build the neural network model with these functions. We specify the number of 763 inputs which is the features; the number of 30 outputs which stands for the industry sector. Between these two layers, the number of hidden layers and neurons can be constantly revised to obtain a better result. As for activation function, we use ReLu in hidden layers and Softmax in output layers. As regards the number of epochs and the size of the batches, they also can be redefined when we run the model.

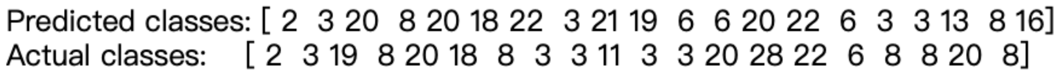
4. Result of the Project

After multiple trials and adjustments, we change the number of hidden layers from 3 to 5, the hidden of neurons, the number of epochs and, the size of batches in order to find the highest accuracy. In this model, in Figure 4, 45 per cent is the highest validation accuracy. We define the four hidden layers, 0.01 learning rate, 30 times of iterations and 1000 batch size. In Figure 5, we are able to achieve on average 7 out of 20 correctly predicted classes. Obviously, it is not a good result. In our view, the main reasons behind these are our computers cannot afford this complete dataset which contains 1,274,430 data entries, cannot try the higher hidden layers and so on. The improvement method will be discussed in the last section.



```
Val accuracy init: 0.01
0 Batch accuracy: 0.38384846 Val accuracy: 0.36
1 Batch accuracy: 0.37088734 Val accuracy: 0.49
2 Batch accuracy: 0.41674975 Val accuracy: 0.49
3 Batch accuracy: 0.43968096 Val accuracy: 0.51
4 Batch accuracy: 0.42871386 Val accuracy: 0.48
5 Batch accuracy: 0.444666 Val accuracy: 0.51
6 Batch accuracy: 0.4546361 Val accuracy: 0.49
7 Batch accuracy: 0.4346959 Val accuracy: 0.47
8 Batch accuracy: 0.45164508 Val accuracy: 0.48
9 Batch accuracy: 0.4775673 Val accuracy: 0.47
10 Batch accuracy: 0.46959123 Val accuracy: 0.49
11 Batch accuracy: 0.5134596 Val accuracy: 0.5
12 Batch accuracy: 0.46161515 Val accuracy: 0.49
13 Batch accuracy: 0.49351946 Val accuracy: 0.47
14 Batch accuracy: 0.49551347 Val accuracy: 0.5
15 Batch accuracy: 0.50149554 Val accuracy: 0.5
16 Batch accuracy: 0.4885344 Val accuracy: 0.5
17 Batch accuracy: 0.48654038 Val accuracy: 0.45
18 Batch accuracy: 0.5104686 Val accuracy: 0.49
19 Batch accuracy: 0.47956133 Val accuracy: 0.46
20 Batch accuracy: 0.48454636 Val accuracy: 0.46
21 Batch accuracy: 0.46859422 Val accuracy: 0.45
22 Batch accuracy: 0.5094716 Val accuracy: 0.44
23 Batch accuracy: 0.55732805 Val accuracy: 0.47
24 Batch accuracy: 0.51645064 Val accuracy: 0.43
25 Batch accuracy: 0.5234297 Val accuracy: 0.45
26 Batch accuracy: 0.5333998 Val accuracy: 0.45
27 Batch accuracy: 0.5443669 Val accuracy: 0.42
28 Batch accuracy: 0.49351946 Val accuracy: 0.4
29 Batch accuracy: 0.5453639 Val accuracy: 0.45
```

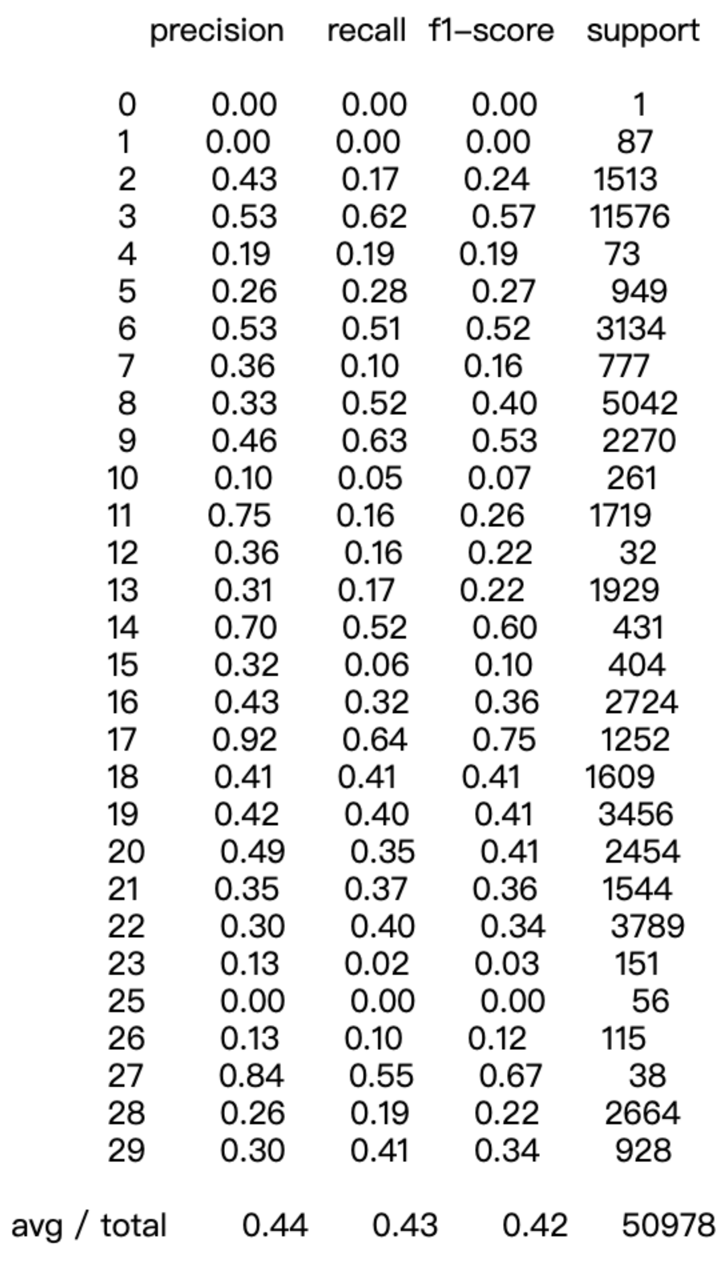
Figure 4 : The highest accuracy



```
Predicted classes: [ 2  3 20  8 20 18 22  3 21 19  6  6 20 22  6  3  3 13  8 16]
Actual classes:    [ 2  3 19  8 20 18  8  3  3 11  3  3 20 28 22  6  8  8 20  8]
```

Figure 5 : The predicted classes and actual classes

Nevertheless, depending on our assumption for the business case, we still can try to get some insight into this result. Interestingly, according to Koehrsen (2018), when the case is not a balance classification, accuracy is not a good measure of model performance. This point can support our model which has a seriously imbalanced dataset. For example, the number of facilities of chemical industry sector is much more than other industries. Under this situation, beyond accuracy, we can consider the values of recall or f1-score in each industry. Recall means the capability of the model to find all relevant instances within a dataset. In Figure 6, we believe that if the value of recall is up to 50%, it can be said that the amount of toxic release of facilities in this industry sector is relatively unchanged such as class 3, Chemicals, class 6, Electric utilities, class 8, Fabricated metals and so on. Therefore, the model can identify these facilities classification. In contrast, if the values if the value of recall is below 20%, it is possible that because the amount of toxic release of facilities in specific industries in recent years already significantly vary such as Chemical Wholesalers, Transportation Equipment. Depending on this concept and following our business case, the EPA might should pay more attention on these low recall and f1-score industry.

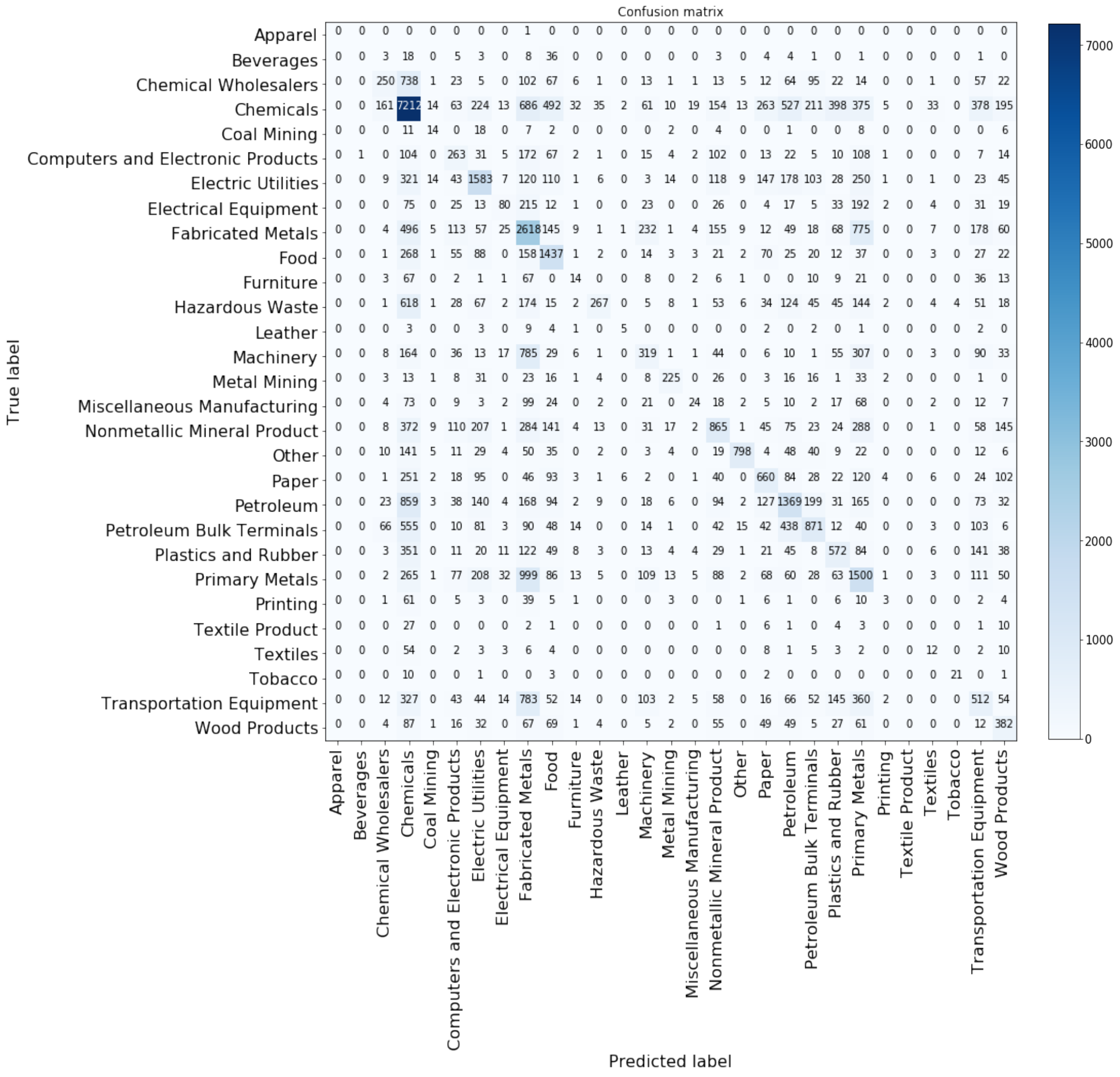


```
             precision    recall  f1-score   support

          0       0.00      0.00      0.00         1
          1       0.00      0.00      0.00        87
          2       0.43      0.17      0.24      1513
          3       0.53      0.62      0.57     11576
          4       0.19      0.19      0.19        73
          5       0.26      0.28      0.27       949
          6       0.53      0.51      0.52      3134
          7       0.36      0.10      0.16       777
          8       0.33      0.52      0.40      5042
          9       0.46      0.63      0.53      2270
         10       0.10      0.05      0.07       261
         11       0.75      0.16      0.26      1719
         12       0.36      0.16      0.22        32
         13       0.31      0.17      0.22      1929
         14       0.70      0.52      0.60       431
         15       0.32      0.06      0.10       404
         16       0.43      0.32      0.36      2724
         17       0.92      0.64      0.75      1252
         18       0.41      0.41      0.41      1609
         19       0.42      0.40      0.41      3456
         20       0.49      0.35      0.41      2454
         21       0.35      0.37      0.36      1544
         22       0.30      0.40      0.34      3789
         23       0.13      0.02      0.03       151
         25       0.00      0.00      0.00        56
         26       0.13      0.10      0.12       115
         27       0.84      0.55      0.67        38
         28       0.26      0.19      0.22      2664
         29       0.30      0.41      0.34       928

avg / total       0.44      0.43      0.42     50978
```

Figure 6 : The precision, recall, f1-score and support of each industry



	Apparel	Beverages	Chemical Wholesalers	Chemicals	Coal Mining	Computers and Electronic Products	Electric Utilities	Electrical Equipment	Fabricated Metals	Food	Furniture	Hazardous Waste	Leather	Machinery	Metal Mining
Apparel	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Beverages	0	0	3	18	0	5	3	0	8	36	0	0	0	0	0
Chemical Wholesalers	0	0	250	738	1	23	5	0	102	67	6	1	0	13	1
Chemicals	0	0	161	7212	14	63	224	13	686	492	32	35	2	61	10
Coal Mining	0	0	0	11	14	0	18	0	7	2	0	0	0	0	2
Computers and Electronic Products	0	1	0	104	0	263	31	5	172	67	2	1	0	15	4
Electric Utilities	0	0	9	321	14	43	1583	7	120	110	1	6	0	3	14
Electrical Equipment	0	0	0	75	0	25	13	80	215	12	1	0	0	23	0
Fabricated Metals	0	0	4	496	5	113	57	25	2618	145	9	1	1	232	1
Food	0	0	1	268	1	55	88	0	158	1437	1	2	0	14	3
Furniture	0	0	3	67	0	2	1	1	67	0	14	0	0	8	0
Hazardous Waste	0	0	1	618	1	28	67	2	174	15	2	267	0	5	8
Leather	0	0	0	3	0	0	3	0	9	4	1	0	5	0	0
Machinery	0	0	8	164	0	36	13	17	785	29	6	1	0	319	1
Metal Mining	0	0	3	13	1	8	31	0	23	16	1	4	0	8	225
Miscellaneous Manufacturing	0	0	4	73	0	9	3	2	99	24	0	2	0	21	0
Nonmetallic Mineral Product	0	0	8	372	9	110	207	1	284	141	4	13	0	31	17
Other	0	0	10	141	5	11	29	4	50	35	0	2	0	3	4
Paper	0	0	1	251	2	18	95	0	46	93	3	1	6	2	0
Petroleum	0	0	23	859	3	38	140	4	168	94	2	9	0	18	6
Petroleum Bulk Terminals	0	0	66	555	0	10	81	3	90	48	14	0	0	14	1
Plastics and Rubber	0	0	3	351	0	11	20	11	122	49	8	3	0	13	4
Primary Metals	0	0	2	265	1	77	208	32	999	86	13	5	0	109	13
Printing	0	0	1	61	0	5	3	0	39	5	1	0	0	0	3
Textile Product	0	0	0	27	0	0	0	0	2	1	0	0	0	0	0
Textiles	0	0	0	54	0	2	3	3	6	4	0	0	0	0	0
Tobacco	0	0	0	10	0	0	1	0	0	3	0	0	0	0	0
Transportation Equipment	0	0	12	327	0	43	44	14	783	52	14	0	0	103	2
Wood Products	0	0	4	87	1	16	32	0	67	69	1	4	0	5	2

	Miscellaneous Manufacturing	Nonmetallic Mineral Product	Other	Paper	Petroleum	Petroleum Bulk Terminals	Plastics and Rubber	Primary Metals	Printing	Textile Product	Textiles	Tobacco	Transportation Equipment	Wood Products
Apparel	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Beverages	0	3	0	4	4	1	0	1	0	0	0	0	1	0
Chemical Wholesalers	1	13	5	12	64	95	22	14	0	0	1	0	57	22
Chemicals	19	154	13	263	527	211	398	375	5	0	33	0	378	195
Coal Mining	0	4	0	0	1	0	0	8	0	0	0	0	0	6
Computers and Electronic Products	2	102	0	13	22	5	10	108	1	0	0	0	7	14
Electric Utilities	0	118	9	147	178	103	28	250	1	0	1	0	23	45
Electrical Equipment	0	26	0	4	17	5	33	192	2	0	4	0	31	19
Fabricated Metals	4	155	9	12	49	18	68	775	0	0	7	0	178	60
Food	3	21	2	70	25	20	12	37	0	0	3	0	27	22
Furniture	2	6	1	0	0	10	9	21	0	0	0	0	36	13
Hazardous Waste	1	53	6	34	124	45	45	144	2	0	4	4	51	18
Leather	0	0	0	2	0	2	0	1	0	0	0	0	2	0
Machinery	1	44	0	6	10	1	55	307	0	0	3	0	90	33
Metal Mining	0	26	0	3	16	16	1	33	2	0	0	0	1	0
Miscellaneous Manufacturing	24	18	2	5	10	2	17	68	0	0	2	0	12	7
Nonmetallic Mineral Product	2	865	1	45	75	23	24	288	0	0	1	0	58	145
Other	0	19	798	4	48	40	9	22	0	0	0	0	12	6
Paper	1	40	0	660	84	28	22	120	4	0	6	0	24	102
Petroleum	0	94	2	127	1369	199	31	165	0	0	0	0	73	32
Petroleum Bulk Terminals	0	42	15	42	438	871	12	40	0	0	3	0	103	6
Plastics and Rubber	4	29	1	21	45	8	572	84	0	0	6	0	141	38
Primary Metals	5	88	2	68	60	28	63	1500	1	0	3	0	111	50
Printing	0	0	1	6	1	0	6	10	3	0	0	0	2	4
Textile Product	0	1	0	6	1	0	4	3	0	0	0	0	1	10
Textiles	0	0	0	8	1	5	3	2	0	0	12	0	2	10
Tobacco	0	0	0	2	0	0	0	0	0	0	0	21	0	1
Transportation Equipment	5	58	0	16	66	52	145	360	2	0	0	0	512	54
Wood Products	0	55	0	49	49	5	27	61	0	0	0	0	12	382

Figure 7 : Confusion Matrix of the model

5. Conclusion

In conclusion, to achieve artificial neural networks, we try to find the more suitable dataset and build the practicable business case. The U.S. EPA toxic release inventory is a worth case which is an important issue. Although the result shows that the processes still need to be improved, it looks can be interpreted under our assumptions. As a result, we think that the organisation can use this model to adjust industry policies when the recall and f1-score under the specific percentage. By contrast, the industry sector which has high accuracy, recall and f1-score can be regarded that the policies should be maintained.

For future work on this study, we recommend that the dataset of each year should be re-loaded from the official site. Although this dataset which we used seems complete and convenient, it seems that it has some format error but too large to open in Excel to revise. If we download the data year by year, it will smaller and easier to figure out and amend in Excel. In addition, computer configuration is also a problem for this dataset. It will be better with GPU which can support to run the whole dataset and add more hidden layers.

6. Reference

EPA.gov. (2016). [online] Available at: https://www.epa.gov/sites/production/files/2016-11/documents/tri_basic_data_file_format_v15.pdf [Accessed 13 Jun. 2019].

Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media.

Hacker Noon. (2017). *What is One Hot Encoding? Why And When do you have to use it?*. [online] Available at: https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f [Accessed 13 Jun. 2019].

Kaggle.com. (2017). *Toxic Release Inventory*. [online] Available at: https://www.kaggle.com/epa/toxic-release-inventory [Accessed 11 Jun. 2019].

Koehrsen, W. (2019). *Beyond Accuracy: Precision and Recall*. [online] Towards Data Science. Available at: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c [Accessed 14 Jun. 2019].

Medium. (2018). *Why do we need feature scaling in Machine Learning and how to do it using SciKit Learn?*. [online] Available at: https://medium.com/@contactsunny/why-do-we-need-feature-scaling-in-machine-learning-and-how-to-do-it-using-scikit-learn-d8314206fe73 [Accessed 14 Jun. 2019].

Scikit-learn.org. (n.d.). *sklearn.preprocessing.StandardScaler — scikit-learn 0.21.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Accessed 14 Jun. 2019].

Sebastian Ruder. (2016). *An overview of gradient descent optimization algorithms*. [online] Available at: http://ruder.io/optimizing-gradient-descent/index.html#adagrad [Accessed 14 Jun. 2019].

U.S. EPA. (n.d.). *Our Mission and What We Do | US EPA*. [online] Available at: https://www.epa.gov/aboutepa/our-mission-and-what-we-do [Accessed 11 Jun. 2019].

U.S. EPA. (n.d.). *Regulatory Information By Sector | US EPA*. [online] Available at: https://www.epa.gov/regulatory-information-sector [Accessed 11 Jun. 2019].

U.S. EPA. (2015). *Factors to Consider When Using Toxics Release Inventory Data | US EPA*. [online] Available at: https://www.epa.gov/toxics-release-inventory-tri-program/factors-consider-when-using-toxics-release-inventory-data [Accessed 15 Jun. 2019].

7. Appendix

All codes are shared with Team7, Rachel, Zahra and Jui-Ting, and mainly based on the book, Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media.

1. Python code - fail to load original dataset

```
### import dataset but fail
import pandas as pd
df = pd.read_csv('basicTRI.csv', skiprows=[2548771])
# Code cited from thispointer.com : https://thispointer.com/pandas-skip-rows-while-reading-csv-file-to-a-dataframe-using-read_csv-in-python/
print(df.shape)
# ParserError: Error tokenizing data. C error: Expected 110 fields in line 2548773, saw 111
```

2. R code – address format errors

```
### Import data
basicTRI <- read_csv("~/Desktop/QM-BA/BUSM131-MASTERCLASS BA/EPA Project/basicTRI.csv⁩")
View(basicTRI)
Origin_TRI = basicTRI
### Dealing with the two format error:

# Remove the rows who have extra 111 column
# Column 14 FEDERAL_FACILITY - only should include "yes or NO"
Origin2_TRI = Origin_TRI[Origin_TRI[,14] == c("YES","NO"),]

NCOL(Origin_TRI)
NROW(Origin_TRI)
NCOL(Origin2_TRI)
NROW(Origin2_TRI)

# Remove 110 column - Unamed:109
# Because after checking with document, it does not exist in the original EPA dataset
Origin2_TRI = Origin2_TRI[ , -110]
NCOL(Origin_TRI)
```

```
NROW(Origin_TRI)
NCOL(Origin2_TRI)
NROW(Origin2_TRI)
# write a new CSV
write.csv(Origin2_TRI, file = "0612_EPA_Origin2Data.csv")
```

3. Python code – final data cleaning

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

import numpy as np
import os

# to make this notebook's output stable across runs
def reset_graph(seed=42):
    tf.reset_default_graph()
    tf.set_random_seed(seed)
    np.random.seed(seed)

# import dataset after correcting format error
import pandas as pd
df = pd.read_csv('0612_EPA_Origin2Data.csv')

df.drop(['Unnamed: 0'], axis=1, inplace=True)
print(df.shape)

df.head()

### sort the data frame by year ascending
df.sort_values(by='YEAR', ascending=True, inplace=True)
# Code cited from Pandas.pydata.org: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html

# show unique value in YEAR column
df.YEAR.unique()

# start from 1993
```

```
df.head()

### deal with NA:

# check with NA
nulls = df.isnull().sum()
nulls[nulls > 0]

# remove NA in INDUSTRY_SECTOR
df = df[pd.notnull(df['INDUSTRY_SECTOR'])]
print(df.shape)
# Code cited from Stack Overflow:
https://stackoverflow.com/questions/13413590/how-to-drop-rows-of-pandas-dataframe-whose-value-in-certain-columns-is-nan

# check with NA after remove NDUSTRY_SECTOR's NA
nulls = df.isnull().sum()
nulls[nulls > 0]

# drop not relative with business case or too much NA variables
df.drop(['TRI_FACILITY_ID'], axis=1, inplace=True)
df.drop(['FRS_ID'], axis=1, inplace=True)
df.drop(['FACILITY_NAME'], axis=1, inplace=True)
df.drop(['STREET_ADDRESS'], axis=1, inplace=True)
df.drop(['CITY'], axis=1, inplace=True)
df.drop(['COUNTY'], axis=1, inplace=True)
df.drop(['ZIP'], axis=1, inplace=True)
df.drop(['BIA_CODE'], axis=1, inplace=True)
df.drop(['TRIBE'], axis=1, inplace=True)
df.drop(['INDUSTRY_SECTOR_CODE'], axis=1, inplace=True)
df.drop(['PRIMARY_SIC'], axis=1, inplace=True)
df.drop(['SIC_2'], axis=1, inplace=True)
df.drop(['SIC_3'], axis=1, inplace=True)
df.drop(['SIC_4'], axis=1, inplace=True)
df.drop(['SIC_5'], axis=1, inplace=True)
df.drop(['SIC_6'], axis=1, inplace=True)
df.drop(['PRIMARY_NAICS'], axis=1, inplace=True)
df.drop(['NAICS_2'], axis=1, inplace=True)
df.drop(['NAICS_3'], axis=1, inplace=True)
df.drop(['NAICS_4'], axis=1, inplace=True)
df.drop(['NAICS_5'], axis=1, inplace=True)
df.drop(['NAICS_6'], axis=1, inplace=True)
df.drop(['DOC_CTRL_NUM'], axis=1, inplace=True)
df.drop(['CAS_#/COMPOUND_ID'], axis=1, inplace=True)
df.drop(['SRS_ID'], axis=1, inplace=True)
df.drop(['PROD_RATIO_OR_ACTIVITY'], axis=1, inplace=True)
df.drop(['PARENT_COMPANY_NAME'], axis=1, inplace=True)
df.drop(['PARENT_COMPANY_DB_NUMBER'], axis=1, inplace=True)

print(df.shape)

# check what we need to do NA
nulls = df.isnull().sum()
nulls[nulls > 0]

# Replacing NAs with mean and 0
df['LATITUDE'].fillna((df['LATITUDE'].mean()), inplace=True)
df['LONGITUDE'].fillna((df['LONGITUDE'].mean()), inplace=True)
df['8.9_PRODUCTION_RATIO'].fillna((df['8.9_PRODUCTION_RATIO'].mean()),
inplace=True)
df['8.8_ONE-TIME_RELEASE'].fillna(0, inplace=True)
# Code cited form Stack Overflow:
https://stackoverflow.com/questions/18689823/pandas-dataframe-replace-nan-values-
with-average-of-columns

#check NA in datatset
nulls = df.isnull().sum()
nulls[nulls > 0]

### start to do dummy variables
# one-hot encoding YEAR
df2 = pd.get_dummies(df['YEAR'])
df = df.join(df2)
df.drop(['YEAR'], axis=1, inplace=True)
print(df.shape)
```

```
# Code cited form Towards Data Science: https://towardsdatascience.com/the-dummys-guide-to-creating-dummy-variables-f21faddb1d40
# one-hot encoding ST
df2 = pd.get_dummies(df['ST'])
df = df.join(df2)
df.drop(['ST'], axis=1, inplace=True)
print(df.shape)

#binary dummy FEDERAL_FACILITY
FEDERAL_FACILITY= {'NO':0,'YES':1,}
df.FEDERAL_FACILITY=df.FEDERAL_FACILITY.map(FEDERAL_FACILITY)
print(df.shape)

# How many output - 30
df.INDUSTRY_SECTOR.unique()

# create the labels for output
INDUSTRY_SECTOR= {'Apparel':0, 'Beverages':1,'Chemical Wholesalers':2,
'Chemicals':3, 'Coal Mining':4, 'Computers and Electronic Products':5, 'Electric
Utilities':6, 'Electrical Equipment':7, 'Fabricated Metals':8, 'Food':9, 'Furniture':10,
'Hazardous Waste':11, 'Leather':12, 'Machinery':13, 'Metal Mining':14, 'Miscellaneous
Manufacturing':15, 'Nonmetallic Mineral Product':16, 'Other':17, 'Paper':18,
'Petroleum':19, 'Petroleum Bulk Terminals':20, 'Plastics and Rubber':21, 'Primary
Metals':22, 'Printing':23, 'Publishing':24, 'Textile Product':25, 'Textiles':26,
'Tobacco':27, 'Transportation Equipment':28, 'Wood Products':29}
df.INDUSTRY_SECTOR=df.INDUSTRY_SECTOR.map(INDUSTRY_SECTOR)
print(df.shape)

# one-hot encoding CHEMICAL
df2 = pd.get_dummies(df['CHEMICAL'])
df = df.join(df2)
df.drop(['CHEMICAL'], axis=1, inplace=True)
print(df.shape)

#binary dummy CLEAR_AIR_ACT_CHEMICAL
CLEAR_AIR_ACT_CHEMICAL= {'NO':0,'YES':1,}
df.CLEAR_AIR_ACT_CHEMICAL=df.CLEAR_AIR_ACT_CHEMICAL.map(CLEAR_AIR_ACT_CHEMICAL)
```

```
print(df.shape)

# one-hot encoding CLASSIFICATION
df2 = pd.get_dummies(df['CLASSIFICATION'])
df = df.join(df2)
df.drop(['CLASSIFICATION'], axis=1, inplace=True)
print(df.shape)

#binary dummy METAL
METAL= {'NO':0,'YES':1,}
df.METAL=df.METAL.map(METAL)
print(df.shape)

# one-hot encoding METAL_CATEGORY
df2 = pd.get_dummies(df['METAL_CATEGORY'])
df = df.join(df2)
df.drop(['METAL_CATEGORY'], axis=1, inplace=True)
print(df.shape)

#binary dummy CARCINOGEN
CARCINOGEN= {'NO':0,'YES':1,}
df.CARCINOGEN=df.CARCINOGEN.map(CARCINOGEN)
print(df.shape)

# one-hot encoding FORM_TYPE
df2 = pd.get_dummies(df['FORM_TYPE'])
df = df.join(df2)
df.drop(['FORM_TYPE'], axis=1, inplace=True)
print(df.shape)

# one-hot encoding UNIT_OF_MEASURE
df2 = pd.get_dummies(df['UNIT_OF_MEASURE'])
df = df.join(df2)
df.drop(['UNIT_OF_MEASURE'], axis=1, inplace=True)
print(df.shape)

df.head()
# write the new CSV
```

```
df.to_csv('After_Datacleaning_EPA.csv')

# finished data cleaning
# now begin analysis (next script)
```

4. Python code – artificial neural networks for EPA TRI

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals
# Common imports
import numpy as np
import os
# to make this notebook's output stable across runs
def reset_graph(seed=42):
    tf.reset_default_graph()
    tf.set_random_seed(seed)
    np.random.seed(seed)

# import dataset
import pandas as pd
df = pd.read_csv('After_Datacleaning_EPA.csv')
print(df.shape)

df.head()

df.drop(['Unnamed: 0'], axis=1, inplace=True)
print(df.shape)

# set all variables to the same data type - float32
for col in df.columns :
    df[col].astype(np.float32)

# set y (output) = INDUSTRY_SECTOR
y = (df['INDUSTRY_SECTOR'])
y.head()

df.drop(['INDUSTRY_SECTOR'], axis=1, inplace=True)
```

```
# check y in the order of df.head()
y.head()

# check the dataset is sorted by year - start from 1993
df[['1993.0','1994.0','1995.0','1996.0','1997.0','1998.0','1999.0','2000.0','2001.0','2002.
0','2003.0','2004.0','2005.0','2006.0','2007.0','2008.0','2009.0','2010.0','2011.0','2012.0',
'2013.0','2014.0','2015.0','2016.0']].head()

# to know which year is 80%
# to know which year splits training and test set (2013)

df.iloc[int(len(df)*0.8)][['1993.0','1994.0','1995.0','1996.0','1997.0','1998.0','1999.0','2
000.0','2001.0','2002.0','2003.0','2004.0','2005.0','2006.0','2007.0','2008.0','2009.0','20
10.0','2011.0','2012.0','2013.0','2014.0','2015.0','2016.0']]

# training set is 80% from year 1993 - 2013
# test set is other 20% from 2013 – 2016

# randomly drop 20% of the dataset to split training and test set
from sklearn.model_selection import train_test_split
rnd_idx = np.random.permutation(len(df))
rnd_idx_first_part = rnd_idx[:int(len(df)*0.80)]    # number = %
rnd_idx_second_part = rnd_idx[int(len(df)*0.80):]
df.drop(rnd_idx_first_part, axis=0, inplace=True)
y = y[rnd_idx_second_part]
y = y.sort_index()

# split data into test and training set
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.2, shuffle=False)

df.head()    # new 20% dataset
print(df.shape) #new 20% dataset #old one is (1274430, 763)
y[:5] # printing first 5 rows of new 20% y

X_train = X_train.values
X_test = X_test.values
y_test = y_test.values
y_train = y_train.values
```

```
# feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test);
# Code cited from Linus

# print the shapes of the training and test sets for x and y
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

import tensorflow as tf

#define validation and training sets
y_train = y_train.astype(np.int32)
y_test = y_test.astype(np.int32)
X_valid, X_train = X_train[:100], X_train[100:]
y_valid, y_train = y_train[:100], y_train[100:]

# define how many neuron and layers
n_inputs = 763    # number of features
n_hidden1 = 1400
n_hidden2 = 700
n_hidden3 = 350
n_hidden4 = 150
#n_hidden5 = 150
n_outputs = 30 # number of outputs

tf.reset_default_graph()      # reset graph
X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int32, shape=(None), name="y")
a = tf.placeholder(tf.float32,shape=(None),name='a')
summary1 = tf.reduce_mean(a, name="summary1")

def neuron_layer(X, n_neurons, name, activation=None):
	with tf.name_scope(name):
        n_inputs = int(X.get_shape()[1])
        stddev = 2 / np.sqrt(n_inputs)
        init = tf.truncated_normal((n_inputs, n_neurons), stddev=stddev)
        W = tf.Variable(init, name="kernel")
        b = tf.Variable(tf.zeros([n_neurons]), name="bias")
        Z = tf.matmul(X, W) + b
        if activation is not None:
            return activation(Z)
        else:
            return Z

# define the activation function
with tf.name_scope("dnn"):
    hidden1 = neuron_layer(X, n_hidden1, name="hidden1", activation=tf.nn.relu)
    hidden2 = neuron_layer(hidden1, n_hidden2, name="hidden2",
activation=tf.nn.relu)
    hidden3 = neuron_layer(hidden2, n_hidden3, name="hidden3",
activation=tf.nn.relu)
    hidden4 = neuron_layer(hidden3, n_hidden4, name="hidden4",
activation=tf.nn.relu)
    #hidden5 = neuron_layer(hidden4, n_hidden5, name="hidden5",
activation=tf.nn.relu)
    logits    = neuron_layer(hidden4, n_outputs, name="outputs")

# define the loss function
with tf.name_scope("loss"):
    xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y,
logits=logits)
    loss = tf.reduce_mean(xentropy, name="loss")

#learning rate - change if have problems with convergence or other errors
learning_rate = 0.01

with tf.name_scope("train"):
    optimizer = tf.train.AdagradOptimizer(learning_rate)
    training_op = optimizer.minimize(loss)

with tf.name_scope("accuracy"):
    correct = tf.nn.in_top_k(logits, y, 1)
    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))

def shuffle_batch(X, y, batch_size):
    rnd_idx = np.random.permutation(len(X))
    n_batches = len(X) // batch_size
    for batch_idx in np.array_split(rnd_idx, n_batches):
        X_batch, y_batch = X[batch_idx], y[batch_idx]
        yield X_batch, y_batch

from datetime import datetime
now = datetime.utcnow().strftime("%Y%m%d%H%M%S")
root_logdir = "."
logdir = "{}//run-{}//".format(root_logdir, now)
print('logdir ',logdir)
targetsindex = tf.argmax(n_inputs, axis=0)

n_epochs = 30          # how many iterations optimiser does
batch_size = 1000 # how many peices data is devided into

init = tf.global_variables_initializer()
saver = tf.train.Saver()

summary2 = tf.summary.scalar('Accuracy', summary1)
file_writer = tf.summary.FileWriter(logdir , tf.get_default_graph())

with tf.Session() as sess:
    init.run()
    acc_val = accuracy.eval(feed_dict={X: X_valid, y: y_valid})
    print("Val accuracy init:", acc_val)
    for epoch in range(n_epochs):
        cnt = 0
        for X_batch, y_batch in shuffle_batch(X_train, y_train, batch_size):
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch})
            cnt = cnt+1
            #print('Batch {}'.format(cnt))
        acc_batch = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
        acc_val = accuracy.eval(feed_dict={X: X_valid, y: y_valid})
```

```
            print(epoch, "Batch accuracy:", acc_batch, "Val accuracy:", acc_val)

            summary_str = summary2.eval(feed_dict={a: acc_val })
            file_writer.add_summary(summary_str, epoch)

        save_path = saver.save(sess, "./my_model_final.ckpt")


with tf.Session() as sess:
    saver.restore(sess, "./my_model_final.ckpt") # or better, use save_path
    X_new_scaled = X_test
    Z = logits.eval(feed_dict={X: X_new_scaled})
    print (Z)
    y_pred = np.argmax(Z, axis=1)

print("Predicted classes:", y_pred[:20]) # list 20 instance
print("Actual classes:      "    , y_test[:20])

file_writer.close()

### confusion matrix
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
#font size
plt.rcParams['axes.labelsize'] = 16
plt.rcParams['xtick.labelsize'] = 16
plt.rcParams['ytick.labelsize'] = 16

# Code cited from Scikit-learn.org: http://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
from sklearn.metrics import confusion_matrix
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
```

```
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

class_names = ['Apparel', 'Beverages','Chemical Wholesalers', 'Chemicals', 'Coal
Mining', 'Computers and Electronic Products', 'Electric Utilities', 'Electrical
Equipment', 'Fabricated Metals', 'Food', 'Furniture', 'Hazardous Waste', 'Leather',
'Machinery', 'Metal Mining', 'Miscellaneous Manufacturing', 'Nonmetallic Mineral
Product', 'Other', 'Paper', 'Petroleum', 'Petroleum Bulk Terminals', 'Plastics and
Rubber', 'Primary Metals', 'Printing', 'Textile Product', 'Textiles', 'Tobacco',
'Transportation Equipment', 'Wood Products']

tf.confusion_matrix(y_test,y_pred,num_classes=30,dtype=tf.int32,name=None,weigh
ts=None)
plt.figure(figsize=(16,16))
cm = confusion_matrix(y_test,y_pred)
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
plt.show()
```

```
# get precision        recall    f1-score      support
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

# support - how many facilities in each label
# recall - the ability of a model to find all the relevant cases within a dataset,    up to
50% we think is robust    ex: 3,6,8,9,14,17,27
# lower than 30% should be paid attention
```