COMP5318: The Apps Market Text Categorisation

Beulah Shantini
baug8178@uni.sydney.edu.au
470162015

Emil Laurence Pastor
epas6415@uni.sydney.edu.au
460466466

Maureen Rocas
mroc9684@uni.sydney.edu.au
470313536

The University of Sydney

I. Introduction

Text categorisation is a useful tool in consolidating data in an organised manner based on the content instead of doing time consuming manual sorting of the text or creating a generalised rule. Having said that, this tool is useful in categorising the label in the Apps Market since new apps are created every day. In addition, text categorisation has many applications in the area document routing, filtering and reporting. Therefore, it is essential to create a classification model that labels the apps based on any textual description provided by the author.

The aim of this project is to build a classifier that has an optimal metrics in terms of accuracy, precision sensitivity within a feasible runtime. These measures are used to evaluate the results of each model. Furthermore, it is imperative is to follow the process model for data mining to ensure systematic delivery and development of the classifier. These study enables us to evaluate the factors affecting the over-all performance of the classifier and the importance of each phase from understanding data, data preparation, modeling and evaluation.

II. Methodology

1. Process Model

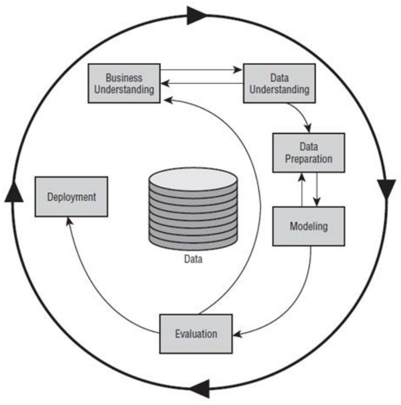


Figure 1 Cross Industry Standard Process for Data Mining (CRISP-DM)

The Cross Industry Standard Process for Data Mining is use as the process model to tackle the classification problem. The life cycle of a data mining project consists of six phases, shown in *Figure 1 Cross Industry Standard Process for Data Mining (CRISP-DM)*. The outcome of each phase determines which phase, or particular task of a phase, has to be performed next. The arrows indicate the most important and frequent dependencies between phases [1]. The first two phase is skipped since the data has already been pre-processed into multi-dimensional Vector Space Model (VSM) using term frequency-inverse document frequency (TF-IDF) weighting method.

2. Data

The data consists of the apps' name and its corresponding description, TF-IDF values of the word found in the description and label. It is Furthermore, the apps can be categorised into one of the 30 categories or labels (e.g. Education, Entertainment, Travel and Local, etc.). The data set has four main files:

1. *training_data.csv*
   This file consists of the apps' name and the pre-processed TF-IDF values from the 13,626 unique words extracted from the description of each application.
2. *training_desc.csv*
   This file contains the raw description data of the apps.
3. *training_labels.csv*
   This file consist of the apps' corresponding class label.
4. *test_data.csv*
   This is the 10% (per label) of the original data set with similar data structure as the training_data.csv.

3. Data Preparation

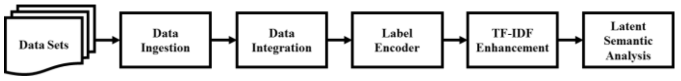


Figure 2 Data Preparation Process

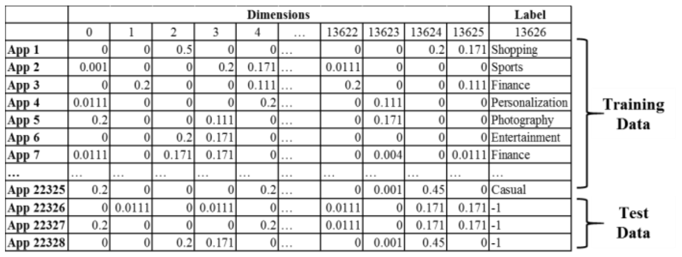
Data is processed as shown in *Figure 2 Data Preparation Process* to ingest data into the model, remove unnecessary records, label encoding and dimensionality reduction. These steps are necessary to ensure that the classifier has optimal values and efficient processing time.

3.1 Data Ingestion

Data sets are loaded using Python pandas.read_csv method because of its high performance and memory efficient parser engine which enables data to be ingested in a shorter span of time compare to numpy and default csv implementations.

3.2 Data Integration

The training_data.csv and training_labels.csv data is integrated and denormalised into a single matrix. Subsequently, rows with zero TF-IDF are removed because those records contributes to the noise level of the data set. This data set is called the *training data*. Meanwhile, label column with a dummy value of -1 is added to the test_data.csv so that the number of dimensions align with the training data. This data set is called *test data*. Furthermore, the two data set is combined into a single matrix (*Figure 3 Combined Training and Test Data*) for further processing.



	Dimensions										Label
	0	1	2	3	4	…	13622	13623	13624	13625	13626
App 1	0	0	0.5	0	0	…	0	0	0.2	0.171	Shopping
App 2	0.001	0	0	0.2	0.171	…	0.0111	0	0	0	Sports
App 3	0	0.2	0	0	0.111	…	0.2	0	0	0.111	Finance
App 4	0.0111	0	0	0	0.2	…	0	0.111	0	0	Personalization
App 5	0.2	0	0	0.111	0	…	0	0.171	0	0	Photography
App 6	0	0	0.2	0.171	0	…	0	0	0	0	Entertainment
App 7	0.0111	0	0.171	0.171	0	…	0	0.004	0	0.0111	Finance
…	…	…	…	…	…	…	…	…	…	…	…
App 22325	0.2	0	0	0	0.2	…	0	0.001	0.45	0	Casual
App 22326	0	0.0111	0	0.0111	0	…	0.0111	0	0.171	0.171	-1
App 22327	0.2	0	0	0	0.2	…	0.0111	0	0.171	0.171	-1
App 22328	0	0	0.2	0.171	0	…	0	0.001	0.45	0	-1

Figure 3 Combined Training and Test Data

3.3 Label Encoder

The label is a categorical variable that pose a challenge especially during one-hot encoding process used in the succeeding process because of incompatibility with the $argmax$ function. Having said that, the label encoder component is converted into a numerical value as shown in *Table 1 Label Encoding Result Set*.

Table 1 Label Encoding Result Set

Encoded Label	Categorical Label	Encoded Label	Categorical Label
0	Arcade and Action	15	Medical
1	Books and Reference	16	Music and Audio
2	Brain and Puzzle	17	News and Magazines
3	Business	18	Personalization
4	Cards and Casino	19	Photography
5	Casual	20	Productivity
6	Comics	21	Racing
7	Communication	22	Shopping
8	Education	23	Social
9	Entertainment	24	Sports
10	Finance	25	Sports Games
11	Health and Fitness	26	Tools
12	Libraries and Demo	27	Transportation
13	Lifestyle	28	Travel and Local
14	Media and Video	29	Weather

3.4 TF-IDF Enhancement

The TF-IDF model is popular and easy to implement, there are two weaknesses [3]:

- The inverse document frequency show that the feature term t_i is more important for classification if the document numbers $N(t_i, C)$ is less distribution in set C.
- For the same category with different feature term t_1, t_2, t_1 appears in all documents of the category more often than t_2, in the average, the ability of the term t_1 to characterize more representative than the t_2.

Thus, the following TF-IDF variants are used to improve the over-all performance of the classification.

- Normalisation of the TF-IDF values by cosine normalisation [3].

$$w(t_{ij}) = \frac{tf_{ij} \cdot idf}{\sqrt{\sum_{j=1}^{n} [tf_{ij} \cdot idf]^2}}$$

- TF-IDF-CI calculates the term frequency in documents within one class. [4].

$$w(t_{ij}) = tf_{ij} \cdot idf \cdot \frac{n_{cij}}{N_{ci}}$$

where n_{cij} represents the number of documents where term j appears within the same class c document i belongs to, N_{ci} represents the number of documents within the same class c document i belongs to.

3.5 Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is a technique for creating a vector representation of a document. Having a vector representation of a document gives you a way to compare documents for their similarity by calculating the distance between the vectors. This in turn means you can do handy things like classifying documents to determine which of a set of known topics they most likely belong to [5]. The general process of doing LSA is to perform dimensionality reduction on sparse matrix of d dimension into denser m dimension of the TF-IDF vectors through Single Value Decomposition specifically, the Truncated SVD.

4. Modeling

The modeling phase involves designing and building of the algorithms, and optimising the parameters and hyper parameters. Furthermore, there are modeling techniques that requires specific form of data. Thus, adjustments to the data preparation maybe necessary.

The following models are implemented and evaluated:

- K-Nearest Neighbor
- Softmax Logistic Regression
- Multinomial Naïve Bayes
- Perceptron

5. Evaluation

This stage of the project involves thorough evaluation of the different model built to ensure that the classification categorises data within feasible time. The following metrics is used to evaluate the model:

5.1 Classification Runtime

The classification runtime is measured from the training phase to the prediction phase of the classification in minutes to seconds.

5.2 Confusion Matrix

This is a table that is used to describe the performance of a classification model on a set of test data for which the true values are known [5]. Ideally, the table has only values in its diagonal M_{ii} which means that the actual is the same as the expected result as shown in *Figure 4 Ideal Confusion Matrix*.

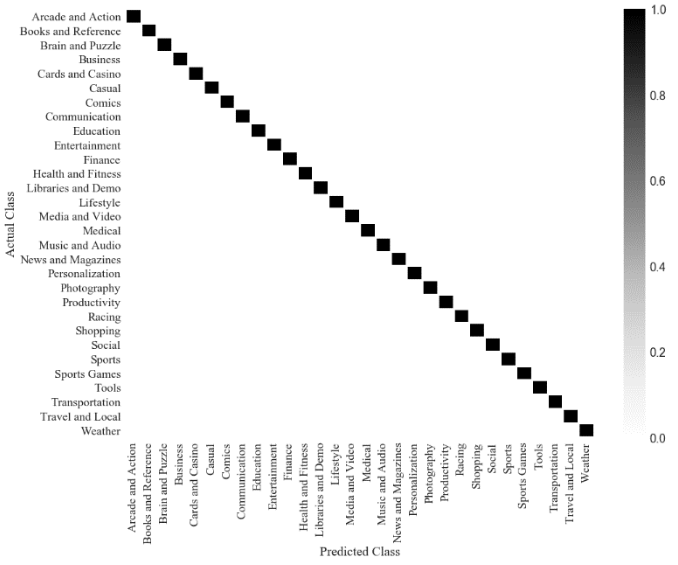


Figure 4 Ideal Confusion Matrix

5.3 Accuracy

Accuracy is the ratio of the correct number of prediction over the number of data as shown in $\boldsymbol{accuracy(y_p, t) = \frac{1}{N}\sum_{i=0}^{N-1} 1(y_p = t)}$.

$$\boldsymbol{accuracy(y_p, t) = \frac{1}{N}\sum_{i=0}^{N-1} 1(y_p = t)}$$

5.4 Precision, Recall and f1-score

Precision is measure of exactness which is the percentage of label i are actual label i. The formula can be derived in the confusion matrix M.

$$\boldsymbol{precision_i = \frac{M_{ii}}{\sum_j M_{ji}}}$$

Precision is measure of completeness which is the percentage of actual label i over what is classified as label i. The formula can be derived in the confusion matrix M.

$$\boldsymbol{recall_i = \frac{M_{ii}}{\sum_j M_{ij}}}$$

The f1-score is the weighted average of the precision and recall, where an F1 score reaches its best value at 1 and `

$$\boldsymbol{f1 - score_i = \frac{2 \cdot precision_i \cdot recall_i}{precision_i + recall_i}}$$

5.5 k-fold Cross Validation

K-fold Cross-validation is a method to evaluate models by partitioning into k fold where $k-1$ training set to train the model, and the fold test set to evaluate it. The process is repeated k times and the performance are then average.

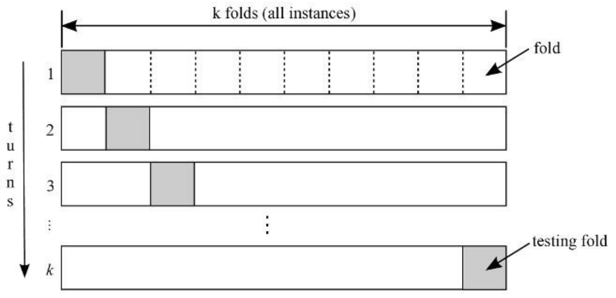


Figure 5 k-fold Cross Validation

III. Classifiers

1. Softmax Logistic Regression

1.1 General Architecture

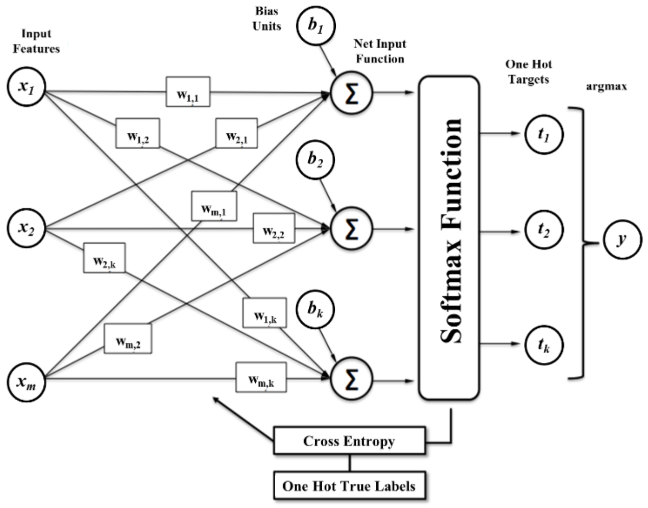


Figure 6 Softmax Logistic Regression Architecture [6]

In the Softmax Logistic Regression, the weights w across m dimensions and the bias term b are the inputs in the Softmax function as shown in *Figure 6 Softmax Logistic Regression Architecture [6]*. Subsequently, the result of the Softmax function is measured against the target using the cost function J with the goal to minimise the value after training and optimisation. Then, the Quantizer for one hot targets that determines the predicted output of the class using the $argmax$.

1.2 Softmax Function

In a multi-label classifier, the Softmax function is used as the activation function in replacement of the Sigmoid function typically used for binary classification.

$$P(y=j|z)=\phi_{softmax}(z)=\frac{e^{z}}{\sum_{j=0}^{K} e^{z}}$$

where input $z=\sum_{i=0}^{m} w_i x_i + b$

1.3 Objective Function

The goal of classification is to minimize the objective function that results to optimal metrics. The Softmax objective function is defined as the average cross entropy across n data and the L2 regularisation term.

$$J(W;b)=\frac{1}{n}\sum_{i=1}^{N} H(t_i,y_i)+\frac{\lambda}{2}\|w\|^2$$

The cross entropy function is defined as the negative of the summation of the target t_i and the logarithm of the predicted out y_i across all dimension m.

$$H(t_i,y_i)=-\sum_m t_i log(y_i)$$

1.4 Gradient Descent

In order of the Softmax to determine the weight coefficients and the bias term, gradient descent functions are used.

Weight coefficient w

$$\frac{\partial J}{\partial w}=\eta\sum_{i=1}^{N} x_i^T(y_i-t_i)$$

where η is the learning rate hyper parameter, x_i is the data in the dimensions, y_i is the predicted label and t_i is the actual label.

The weight coefficient is initialised as a random small number matrix of dimension m by label k and 0.001 is the weight factor that gives the optimal result.

$$W = 0.001 * random(m,k)$$

Bias Term b

$$\frac{\partial J}{\partial b}=\eta\sum_{i=1}^{N}(y_i-t_i)$$

where η is the learning rate hyperparameter, y_i is the predicted label and t_i is the actual label.

The initial bias term values are set to zero.

2. K-Nearest Neighbour (KNN)

KNN is a simple algorithm which uses classification decision based on its K nearest neighbours. We consider the cosine similarity to K nearest neighbours.

SVD is used to identify the principal R components that rotate the document matrix on R features. The R features are used to rotate the incoming test data for prediction and K similar neighbours with similar cos angle are found and their class becomes our prediction result.

A good selection of K should help us achieve robustness to outliers especially with noisy data.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

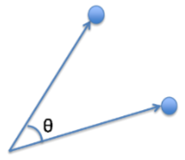


Figure 7 Similarity between 2 Documents Vectors

where, A and B are low dimensional document vectors

$$X = U_M \, . \, \in_M \, . \, {V^t}_M$$

$$\hat{X} = U_R \, . \, \in_R$$

where, X is a matrix in M dimensions, R is the reduced dimension. With $\hat{X}$ our document model in R dimension we find similarity with unlabelled test data.

3. Multinomial Naïve Bayes

Naïve Bayes is a probabilistic model which uses Bayes rule to calculate the probability of X belonging to class C. It is simple and fast. Faster to train and predict. The main advantage being it uses conditional probability. It calculates probability of discrete independent words being together in a document. The assumption though being words are independent.

$$P(D_i/C) \approx P(C) \prod_{t=1}^{N} P(W_t/C)^{x_{it}}$$

where, D is the ith unclassified document, W is the N bag of words, x is the TF-IDF value, t the dimensionality of features

The prior probability of a document to be in a class, likelihood of each word in a class is calculated in training. For a new unclassified document the posterior probability is calculated using the above equation where first part is the prior P(C) and second part is the multinomial probability.

The biggest disadvantage of Naïve Bayes is zero likelihood of word not in training set. This lead to probability of some documents in class to be zero. This is handled using add one smoothing technique.

The feature selection technique like LSA, SVD has negative TF-IDF value producing negative likelihood. Hence feature selection is done by choosing columns with high TF-IDF weight.

Likelihood is calculated using the formula

$$P\left(\frac{W_i}{C}\right) = \frac{\sum_{j=1}^{N} X_{ij}}{\sum_{V=1}^{M} \sum_{j=1}^{N} X_{ij}}$$

Where, i is the word index in column, j is the number of rows in a category and V is the total M dimensions of a category in column.

4. Perceptron

Perceptron Classification is an online learning algorithm where the weights of the features or inputs are calculated during the learning phase and is used as input to predict the target variable. For multiclass classification, each class has its own set of weights and each class data is represented by its own set of inputs or features. The weights of the class, calculated during the training phase, are updated when a wrong prediction is made. The new weights are then use in the next iteration.

Weights formula: $w^{t+1} = w^t \pm f(x,y) - f(x,y^*)$

Where, y is the expected category and y^* is the predicted category

The calculated weights of each class are then averaged at the end of all the training-epoch phase and is then use to predict the class of the test data. Like many linear regression models, perceptron uses a bias variable, which is used to activate the perceptron. When all features are 0 or loss is 0, bias value is used to move the value beyond 0. For perceptron implementation, bias “1” is set at weight[0].

4.1 Feature Extraction

In order to use Perceptron for the Apps Market classification, several processes are done to find the features specific for each class. For all market app belonging to the same class, their TF-IDF values are sum up, creating 1 row with N features. Non-zero values are used to determine the features and TF-IDF values, normalizing by the number of apps per class, are sorted from highest to lowest. The sorted TF-IDF values now represent the top features for that class.

To simplify the input to our Perceptron classifier, the sum of the TF-IDF that corresponds to the features of the class. This means for 30 categories; each instance of the Market App has 30 TF-IDF that corresponds to each 30 categories. Since TF-IDF represents the words that best describes a Market App, each of the calculated TF-IDF value represents the words that best describe the Market App for a specific category.

For example, “Photography” top feature indexes are [0, 1, 2, 3, 4], while “Sports” top feature index are [0, 2, 5, 6, 7].

	0	1	2	3	4	5	6	7	Total	Average
App1	.12	0	.1	0	0	.23	.1	0		
Comics	.0012	0	.13	0	0	0	0	0	.1312	.02624
Sports	.0012	0	0.13	0	0	.23	0.1	0	0.46	.092

For app title “App1”, its TFI-DF value to represent Photography is 0.02624; while Sports is 0.092. Note that we normalize the value by the number of top features.

4.2Naïve Bayes with Perceptron

A document is represented by its TF-IDF values; and what we want to know is which category has the highest probability. We have calculated the document’s TFI-DF representation for each category. We also know the Probability of a category. Given these data, we can now use Naïve Bayes to find the Category given the document’s TFI-DF value.

$$f(x, y) = P(y|x) = \frac{P(x|y)P(y)}{\mathrm{P(x)}}$$

P(y) - Total Apps in Category / Total App

P(x|y) - is the total TFI-DF, given the features of that category.

P(x) - is the total TF-IDF of the document

Since all category instances of document have the same P(x) value, we can eliminate this from our equation. Our new equation would now be:

$$P(y|x) = P(x|y)\ P(y)$$

This equation is used by the Perceptron classifier to predict the category of a document. Whichever has the highest probability is the class of the Market App.

Prediction formula: $y^{*} = \underset{y}{\operatorname{argmax}}\, \mathrm{w} \cdot \mathrm{f(x, y)}$

IV. Experimental Analysis

1. Data Set

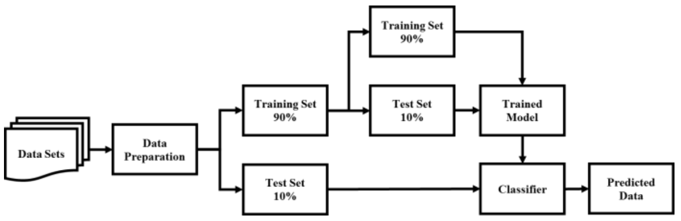


Figure 8 Training and Test Data Split

The data set is obtained from the Apps Market that contains 30 classes. The training set is composed of the 20,096 records whilst the test set has 2,266 records. After the data preparation as shown in *Figure 8 Training and Test Data Split* the training data set is to be split into 90% Training Set and 10% Test Set for training and parameter and hyper parameter optimisation.

2. Hardware and Software Configuration

The table below is the hardware and software specifications used to run the experiment and benchmark the result.

Table 2 Hardware and Software Configuration

Processor	Intel Core i5-6300U @ 2.4GHz 64-bit
RAM	8 GB
Operating System	Windows 10 Enterprise 64-bit
Python	Python 3.6.0 Anaconda 4.3.1 (64-bit)|
Python Libraries	Datetime OS

	Pandas Numpy Scikit.decomposition TruncatedSVD

3. Chosen Model: Softmax Logistic Regression

The data set has 13,626 features that serves as an input to the classifier. However, the execution time is significantly impacted if the dimension is not reduced. Having said that, the dimension is reduced to 500 components using Latent Semantic Analysis, the best dimensionality reduction technique for TF-IDF data, to have the optimal accuracy whilst not sacrificing the execution time of the classifier as shown in *Figure 9 Number of Components vs Accuracy, Execution Time*.

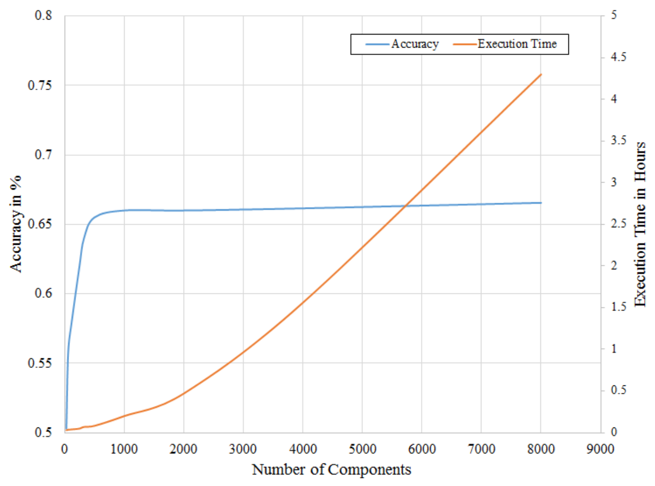


Figure 9 Number of Components vs Accuracy, Execution Time

Different variations of TF-IDF is also used to verify if the effect in the overall accuracy of the model. The TF-IDF-CI model cannot be implemented because the label of the test data is not available and it should have given the highest accuracy of more than 80%. The result is shown in *Table 3 Weighting Method Comparison*.

Table 3 Weighting Method Comparison

Weighting Method	Maximum Accuracy
TF-IDF	66.55%
TF-IDF Normalised	63.25%
TF-IDF-CI	N/A (>80%) Test Data Label is not Available

The learning rate hyperparameter that is used in gradient descent is optimised to ensure maximum accuracy. After series of trial and error, 0.001 is the learning that provides maximum accuracy.

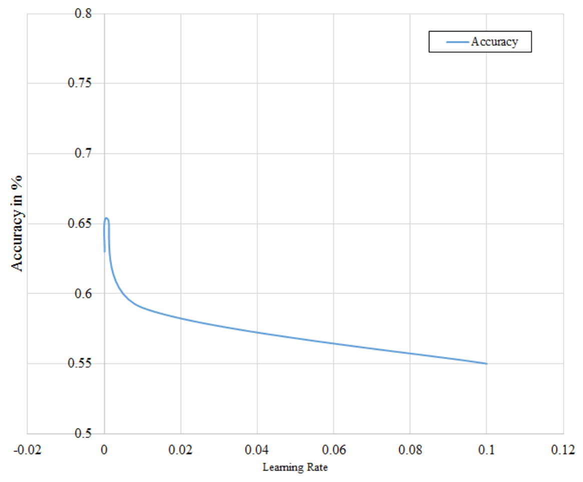


Figure 10 Learning Rate

Furthermore, the cost function shown in *Figure 11 Cost Function of Train and Test Data*that the optimal number of iteration for the learning of 0.001 where the model does not overfit is observed at 1400.

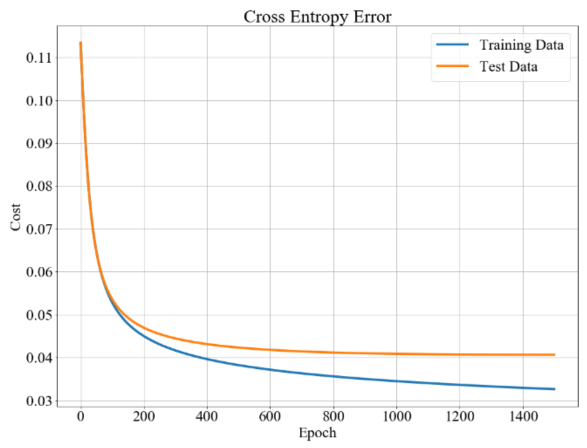


Figure 11 Cost Function of Train and Test Data

After performing optimisation of the parameters and hyper parameters, the 10-fold Cross Validation is performed to obtain the mean accuracy of the classifier.

V. Evaluation

1. Softmax Logistic Regression

The table below shows the general performance of the Softmax Logistic Regression classifier:

Table 4 General Result for Softmax Logistic Regression

Min Accuracy	Mean Accuracy	Max Accuracy
62.5%	64.16%	66.55%

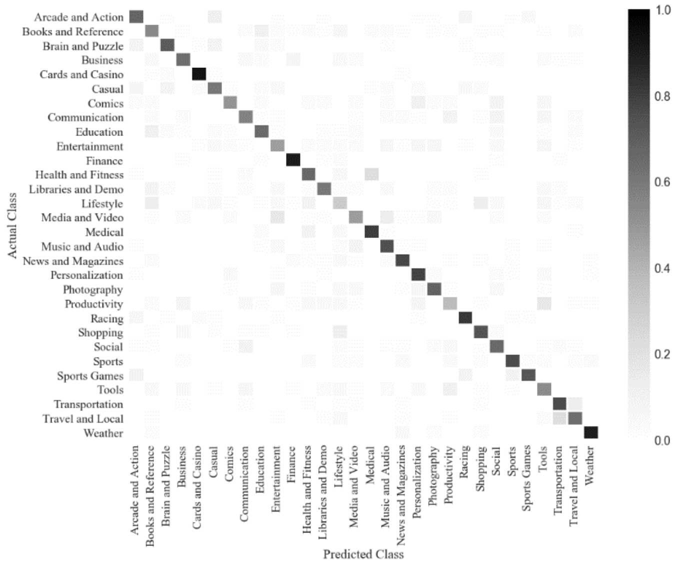


Figure 12 Confusion Matrix for Softmax Logistic Regression

The confusion matrix as shown in *Figure 12 Confusion Matrix for Softmax Logistic Regression* is a heatmap of the actual label against the predicted label where the darkest shade of grey means higher accuracy.

The classification report as shown in *Table 5 Softmax Logistic Regression Classification Report* provides the individual measures of each label. Labels such as Weather, Cards and Casino, Sports Games, Finance, Sports provides the strongest result because the terms or words in the description generally used for these labels are definitive over Books and Reference, Productivity, Entertainment, Tools and Lifestyle that has lacklustre performance where there are overlaps with other labels.

Table 5 Softmax Logistic Regression Classification Report

Class	Precision	Recall	f1-score	Support
Arcade and Action	0.66	0.68	0.67	68
Books and Reference	0.49	0.55	0.52	73
Brain and Puzzle	0.79	0.71	0.75	70
Business	0.69	0.64	0.66	69
Cards and Casino	0.89	0.93	0.91	71
Casual	0.61	0.60	0.61	73
Comics	0.60	0.50	0.55	36
Communication	0.62	0.57	0.59	72
Education	0.67	0.65	0.66	75
Entertainment	0.47	0.47	0.47	70
Finance	0.84	0.90	0.87	71
Health and Fitness	0.64	0.66	0.65	77
Libraries and Demo	0.64	0.60	0.62	48
Lifestyle	0.32	0.32	0.32	73
Media and Video	0.59	0.47	0.53	74
Medical	0.73	0.80	0.76	66
Music and Audio	0.65	0.75	0.70	73
News and Magazines	0.79	0.77	0.78	71
Personalization	0.67	0.79	0.73	72
Photography	0.74	0.68	0.71	73
Productivity	0.49	0.38	0.43	66
Racing	0.79	0.82	0.81	66
Shopping	0.70	0.74	0.72	73
Social	0.61	0.66	0.63	73
Sports	0.83	0.76	0.79	63
Sports Games	0.86	0.73	0.79	44
Tools	0.47	0.54	0.50	71
Transportation	0.71	0.76	0.74	72
Travel and Local	0.66	0.64	0.65	72
Weather	0.90	0.90	0.90	49
Avg/ Total	**0.66**	**0.66**	**0.66**	**2024**

The runtime statistics of the classifier is shown in *Table 6 Softmax Logistic Regression Runtime Statistics* where the actual runs within 5 minutes and the 10 fold cross validation runs about 15 minutes.

Table 6 Softmax Logistic Regression Runtime Statistics

Stage	Runtime in minutes seconds
Reading Training Data	0:01:12.892439
Reading Test Data	0:00:07.343911
Data Cleansing	0:00:07.507405
Dimensionality Reduction	0:02:53.705306
Classification	0:04:47.836302
Cross Validation	0:15:34.955421

2. Other Classifiers

Table 7 Performance Measures of Other Classifiers

Classifiers	Max Accuracy	Precision	Recall
KNN	0.59	0.59	0.59
Naïve Bayes	0.65	0.65	0.65
Perceptron	0.65	0.65	0.65

The other classifiers performs well though the runtime performance is more than 5 minutes resulting for the Softmax Logistic Regression to be the main classifier.

VI. Reflections

Each phase in data mining plays a vital role in over-all success. Firstly, understanding the data and having a domain knowledge is particularly important to develop a cohesive and robust solution. Secondly, data preparation is critical in the success of the model since feature extraction and data cleansing removes any nuisance that may contribute in the noise level of the input. In this case, the sparsity and volume of the dataset presents interesting challenges in terms of performance in prediction statistics as well as runtime optimisations. Also, having a pre-processed TF-IDF data restrict our options during feature selection that contributes in enhancing the over-all performance. Furthermore, having a good hand in data preparation puts the modeling phase in a good shape.

In the data development phase, the libraries for loading data, processing is chosen with considerations of their runtime. The data dimensionality reduction techniques used are unique to a model. The extraction of features posed challenges to achieve the right reduction technique which still has room for improvement. The tuning of parameters in each model to achieve good statistical data revealed the amount of complexity in a model.

In terms of simplicity in algorithm and parameter tuning KNN and Naïve Bayes stands out. The assumption of independence of words in Bayes Naïve does not correspond to training data, which needed better data processing techniques to overcome such assumptions. KNN based on similarity did gain advantage with low dimensional document representation but insensitivity to polysemy and synonyms could not achieve better results. The Softmax Logistic Regression can be considered as a probabilistic framework because of the numerous parameters and hyper parameters that can be utilised for optimising the result of the classifier.

Over-all this activity provides the team members a wide perspective in machine learning and data machine that it tickles our imagination to push the boundaries given limited resources. This provides us strong understanding and experience in systematically solving data mining problems.

VII. Conclusion

Based on the experiments and classifiers, it has been found that the Softmax Logistic Regression has the most optimal result in terms of runtime, accuracy and other classification metrics like recall, precision. The maximum accuracy of the model is at 66.55% with a cross validated accuracy of 64.16% and 19 times better than the random guess of 3.33% given 30 possible labels.

The experiment has still room to improve the over-all performance of the classifier. In the data set perspective, the raw training and test data is beneficial because it gives the project proponents to do more experimentation like taking into consideration the class information as part of the TF-IDF weighting algorithm, anomaly detection to remove unwanted data and handling of synonyms and polysemy in the raw data. Furthermore, collecting more training data significantly aid the classifier in building a more robust and accurate model. Future work can consider using reinforcement learning and deep learning models.

VIII. References

[1] P. Chapman, T. Khabaza, C. Shearer (2000). CRISP-DM 1.0 Step-by-step Data Mining Guide, 67-68

[2] W. McKinney (2012). A new high performance, memory-efficient file parser engine for pandas. Retrieved from http://wesmckinney.com/blog/a-new-high-performance-memory-efficient-file-parser-engine-for-pandas/.

[3] M. Zhanguo, F. Jing, C. Liang, H. Xiangyi, S. Yanqin (2011). An Improved Approach to Terms Weighting in Text Classification, 2

[4] M. Liu, J. Yang (2012). An improvement of TF-IDF weighting in text categorization, 2

[5] K. Markham (2014). Simple guide to confusion matrix terminology. Retrieved from

http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/.

[6] S. Raschka (2014). Softmax Regression. Retrieved from http://rasbt.github.io/mlxtend/user_guide/classifier/SoftmaxRegression.

[7] Kei Kubo, Peter Nelson, Erik Ruggles, James Sheridan, and Sam Tucker http://cs.carleton.edu/cs_comps/0910/netflixprize/final_results/knn/index.html

[8] Wikipedia Perceptron. Retrieved from https://en.wikipedia.org/wiki/Perceptron

[9] P. Abbeel (2012). Perceptron Update. Retrieved from https://www.youtube.com/watch?v=mH4QSs5IGao&t=17s

Appendix A: How to Run the Code

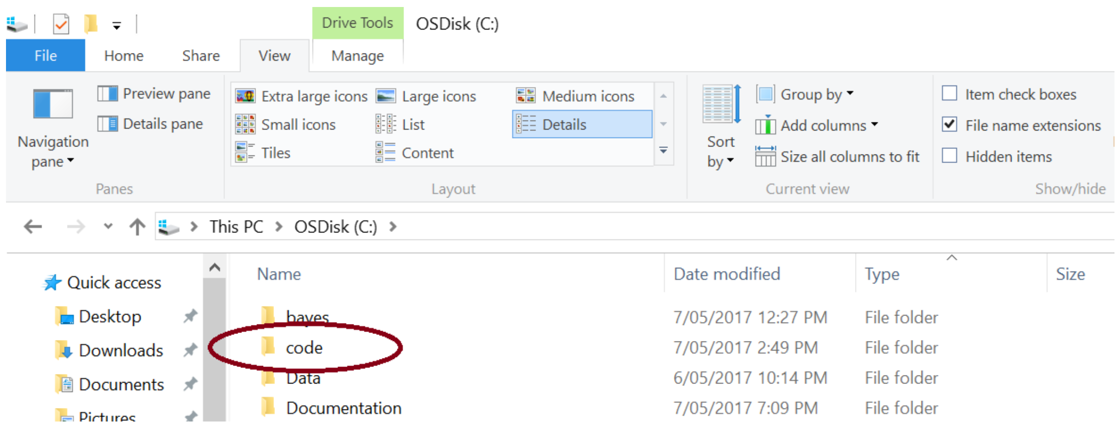
Pre-Requisite:

- Python 3.6.0 Anaconda 4.3.1 (64-bit) Distribution

Instruction:

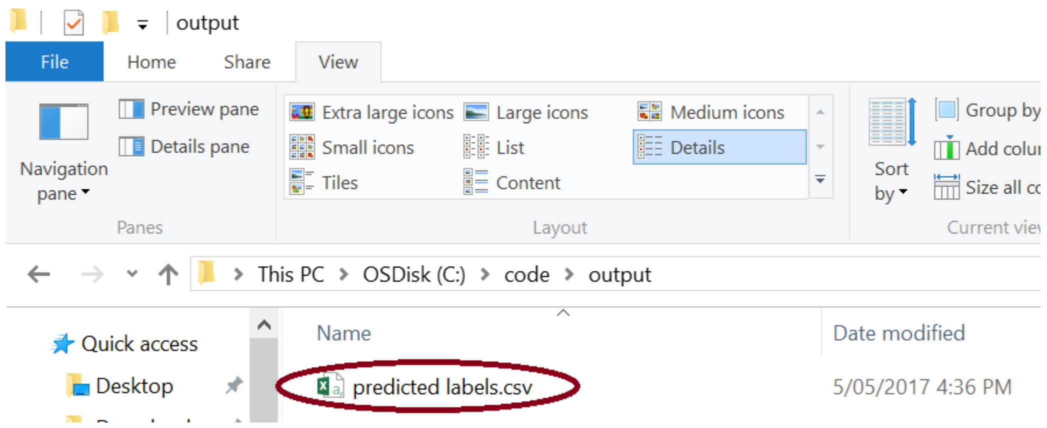
1. Go to C: Drive

2. Unzip the 470162015_460466466_470313536.zip and make sure that the code folder is extracted in the C drive.



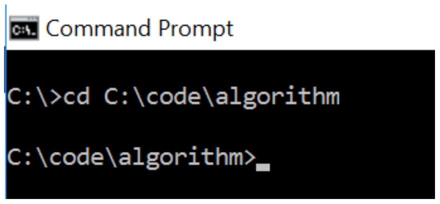
Name	Date modified	Type	Size
bayes	7/05/2017 12:27 PM	File folder	
code	7/05/2017 2:49 PM	File folder	
Data	6/05/2017 10:14 PM	File folder	
Documentation	7/05/2017 7:09 PM	File folder	

Note: The predicted_labels.csv is found in C:\code\output folder.



Name	Date modified
predicted labels.csv	5/05/2017 4:36 PM

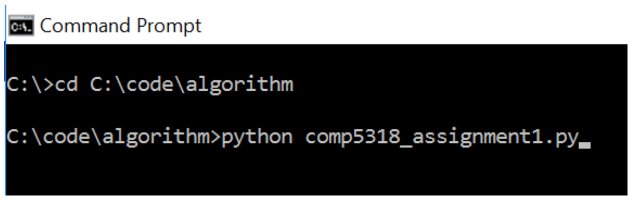
3. Open the command prompt. Change directory using the command cd C:\code\algorithm



```
C:\>cd C:\code\algorithm

C:\code\algorithm>_
```

4. Run the program using the command *python comp5318_assignment1.py* to run the Softmax Regression.


Command Prompt

```
C:\>cd C:\code\algorithm

C:\code\algorithm>python comp5318_assignment1.py
```

***Running other classifiers and additional parameters:**

Main Class: comp5318_assignment1.py

Python Version: 3.0+

Arguments:

-path The source directory which contain the following folders: algorithm, input, output

-c Indicates the classifier to run: softmax, perceptron, naive, knn. The default is softmax.

-cv A value of True will run the cross validation on the training data, setting kfold to 10 if not given. The default is False.

To run softmax logistic regression:

comp5318_assignment1.py -path <the source directory>

To run perceptron:

comp5318_assignment1.py -path <the source directory> -c perceptron

To run naive bayes:

comp5318_assignment1.py -path <the source directory> -c naive

To run KNN:

comp5318_assignment1.py -path <the source directory> -c knn

To run the classifiers with cross validation on the training data, add -cv True