RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Assignment

Course Code	CSC402A
Course Name	Data Mining
Programme	B.Tech
Department	CSE
Faculty	FET

Name of the Student	Satyajit Ghana
Reg. No.	17ETCS002159
Semester/Year	07/2021
Course Leader(s)	Prof. Mohan Kumar

Declaration Sheet

Student Name	Satyajit Ghana		
Reg. No	17ETCS002159		
Programme	B.Tech	Semester/Year	07/2021
Course Code	CSC402A		
Course Title	Data Mining		
Course Date		to	
Course Leader	Prof. Mohan Kumar		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

1 Question 1

Solution to Question No. 1 Part B

1.1 Introduction

A few more pre-processing steps were performed on the dataset, to clean it, which will then be used by the learning algorithms.

Columns like [user_id, isbn, city, state] are dropped, since they don't contribute to the rating classification, and they don't really tell a lot about rating of the book (not much of information gain), also something to note (since we are considering the country) column, that most of the users (~70%) are from usa, which may/may not contribute to lot to the accuracy of the classification.

```
[ ]  1 dataset.head()
```

	age	book_rating	book_title	book_author	year_of_publication	publisher	country
0	34	5	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
2	30	8	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
4	34	9	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
5	34	8	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
6	34	9	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada

The target column is `book_rating`, now we have 10 classes for `book_rating`, but to make the task a little simpler we will be labelling the `book_rating` into 3 classes, namely `[low, mid, high]`.

The stats for them are,

high	mid	low
58.482%	39.248%	2.269%

```
[ ]  1 bins = [0, 3, 7, 10]
     2 names = ['low', 'mid', 'high']
     3
     4 dataset['book_rating'] = pd.cut(dataset['book_rating'], bins, labels=names)
```

```
[ ]  1 dataset.head()
```

	age	book_rating	book_title	book_author	year_of_publication	publisher	country
0	34	mid	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
2	30	high	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
4	34	high	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
5	34	high	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada
6	34	high	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	canada

```
[ ]  1 dataset['book_rating'].value_counts()
```

```
high    213208
mid     143087
low       8275
Name: book_rating, dtype: int64
```

To perform any kind of learning on this dataset we would have to convert the categorical values into numerical, to do this there are two options

- OneHotEncoder
- LabelEncoder

Label Encoder is used for those categorical values which have a "ordering" associated with them, for example the temperature can be cold, warm or hot, we can assign the values 0, 1, 2 to them, that makes sense right? Yes! But now let's say the categorical values are Satyajit, Mohan, Ram how can a number be assigned to them? We cannot assign 0, 1, 2 to them, because we don't how to order them, to deal with this problem we use OneHotEncoder, which simply assigns them `[1, 0, 0]`, `[0, 1, 0]` and `[0, 0, 1]` which are basically indicating presence, of `[Satyajit, Mohan, Ram]` and there is no ordering involved here.

But looking at our dataset, we have a lot of data, and the since we are considering the book name, author, publisher all as categorical value, they will have quite a lot of categories, which

means that when we do one hot encoding, there will be a lot of columns generating, which will be very sparse ofcourse, but this means that we will be restricted by memory consumption.

```
[12]  1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

[13]  1 numeric_features = ['age', 'year_of_publication']
      2 numeric_transformer = Pipeline(steps=[
      3     ('imputer', SimpleImputer(strategy='median')),
      4     ('scaler', StandardScaler())])

[14]  1 categorical_features = ['book_title', 'book_author', 'publisher', 'country']
      2 categorical_transformer = OneHotEncoder(handle_unknown='ignore')

[15]  1 preprocessor = ColumnTransformer(
      2     transformers=[
      3         ('num', numeric_transformer, numeric_features),
      4         ('cat', categorical_transformer, categorical_features)])

[ ]   1 dummy = preprocessor.fit_transform(X_train)
      2 dummy.shape

      (291656, 177473)

If we were to store all the all data, and each cell occupying 4 bytes, this is how much space we would require

[ ]   1 f'{(dummy.shape[0] * dummy.shape[1] * 4) // 1024 // 1024 // 1024} giga bytes'

      '192 giga bytes'
```

From the above illustration, we can see if we try to save this data, we would approximately require 192 GB, and that is just the training set. (I did try it on a 128GB RAM machine, and I got out-of-memory error). We'll later discuss some of the solutions to this.

1.2 Supervised Learning

Supervised Learning was performed on the cleaned dataset, with OneHotEncoding and StandardScaler with the following algorithms

- Decision Tree
- SVC (Support Vector Classification)
- Passive Aggressive
- Ridge Classifier
- AdaBoost
- LinearSVC
- MLP (Multi Later Perceptron)
- RandomForest

Refer to the notebook attached at the end for more information of accuracy, confusion matrix, f1 score, etc., here we will compare the accuracies and time taken to train for all the models.

Algorithm	**F1 Score**	**Balanced Accuracy**	**Avg Precision**	**Avg Recall**	**Time Taken (s)**
DecisionTree	0.60	0.36	0.58	0.60	146
SVC	NULL	NULL	NULL	NULL	>1000
PassiveAggresive	0.55	0.37	0.55	0.55	**8**
Ridge	0.58	0.37	0.56	0.59	22
AdaBoost	0.59	0.34	0.57	0.59	50
LinearSVC	0.58	0.37	0.56	0.59	200
MLP	**0.61**	**0.38**	**0.60**	**0.61**	630
RandomForest	0.58	0.33	0.60	0.59	537

All of the above models were trained by using Imputation, and Standard Scaler, but we can see and vary the scaler to see how it affects the accuracy, to do that we consider the following scalers, and since Passive Aggressive algorithm was the faster, we will stick with that,

- StandardScaler
- MinMaxScaler
- RobustScaler
- MaxAbsScaler
- PowerTransformer
- QuantileTransfomer
- Normalizer
- UnScaled (Original Data)

Scaler	**F1 Score**	**Balanced Accuracy**	**Avg Precision**	**Avg Recall**
StandardScaler	0.561	**0.378**	0.55	0.56
MinMaxScaler	0.541	0.369	0.55	0.54
MaxAbsScaler	**0.588**	0.359	**0.55**	**0.59**
RobustScaler	0.576	0.366	0.54	0.58
PowerTransform (Yeo-Johnson)	0.525	0.374	0.55	0.53
QuantileTransform	0.519	0.371	0.54	0.52
Normalizer (L2)	0.570	0.366	0.54	0.57
UnScaled	0.583	0.364	0.55	0.58

1.3 Unsupervised Learning

In the case of unsupervised algorithms, most of them crashed on the original data, since the data was very sparse, and a lot of memory was required.

Learning from the problems faced in supervised learning, one solution to deal with text data is to perform NLP (Natural Language Processing) on the text itself, i.e. we can use pretrained word embeddings to encode the book names, authors, publishers, country into a dense matrix, which can be easily done by using spacy language models.

All of the columns in the dataset was concatenated to form a single sentence, we will later encode this sentence and try to apply clustering algorithm to predict rating.

```
[8]  1 c_dataset = dataset["book_title"].astype(str) + " "  + \
     2             dataset["book_author"].astype(str) + " " + \
     3             dataset["publisher"].astype(str) + " " + \
     4             dataset["year_of_publication"].astype(str) + " " + \
     5             dataset["age"].astype(str) + " " + \
     6             dataset["country"].astype(str)
     7
```

```
[9]  1 for ex in c_dataset.sample(frac=0.2)[:5]:
     2     print(ex)
```

```
The Vanishing Vampire (The Accidental Monsters , No 1) David Lubar Scholastic 1997.0 12.0 usa
REMEMBER ME Mary Higgins Clark Simon &amp;amp; Schuster 1994.0 34.0 usa
The Mummy or Ramses the Damned Anne Rice Ballantine Books 1991.0 22.0 usa
Bittersweet Rain Sandra Brown Warner Books 2000.0 34.0 usa
Arthur Stephen R. Lawhead Zondervan Publishing Company 1996.0 21.0 usa
```

Also learning from the fact that I don't have enough time to fit a model on this large (300,000) data, we can simplify evaluation of algorithms by sampling a fraction of the original dataset. Even on a really good machine (i7-5930K, 128GB RAM), the model takes a lot of time to fit.

The following algorithms were used,

- KMedoids Clustering

- Agglomerative Clustering
- DBSCAN
- KMeans
- HDBSCAN
- MiniBatchKMeans

Algorithm	**F1 Score**	**Balanced Accuracy**	**Avg Precision**	**Avg Recall**	**Time Taken (s)**
KMedoids	0.30	0.31	0.50	0.31	223
Agglomerative	0.27	0.30	0.48	0.27	119
DBSCAN	**0.58**	**0.33**	0.42	**0.58**	**116**
KMeans	0.31	0.33	0.50	0.32	222
HDBSCAN	**0.58**	**0.33**	**0.54**	**0.58**	121
MiniBatch KMeans	0.33	0.32	0.49	0.34	217

To see the classification report, and the confusion matrix, see appendix.

1.4 Comparative Analysis and Conclusion

If we were to compare the supervised and unsupervised models, they have pretty much comparable accuracies (~58-61%), but one thing that cannot be compared is the execution time, since we used clustering algorithms, they require heavy memory usage, and a lot of processing time.

The highest accuracy in Supervised Learning was achieved by MLP (Multi-Layer Perceptron) of 61%, and in Unsupervised Learning, highest was achieved by HDBSCAN of 58%. And in terms of Scaler, it was observed that MaxAbsScaler gave the highest accuracy compared to other scalers.

Something even more important to address is why the accuracy is so low, the `"high"` class of `book_rating` has 58% total records, and this means that if we were to classify any input as `"high"` we have a probability of `0.58` to be correct, this is exactly like a `OneR` learner, but the reason why we are observing such bad accuracy is because there isn't much correlation between the attributes and the target value, this dataset was never meant to do something like this, it was more towards recommending books to a new user with a given past book purchase history, predicting rating based on country, book title, publisher just does not make sense, just based on the book title, how could a book be rated better? One thing that could make sense is the author, there are certain authors that have a good reputation and they have a higher chance of getting a good rating. But in this assignment, we have considered all the attributes, which could have been a mistake, one lesson learnt is that, grabbing all the available data can have negative effects to the accuracy.

But there is another issue, if we see the number of categorical values for each of the categorical columns,

```
[136]  1 dataset['book_title'].cat.categories.shape
       (132033,)

[139]  1 dataset['book_author'].cat.categories.shape
       (60652,)

[140]  1 dataset['publisher'].cat.categories.shape
       (11311,)

[141]  1 dataset['country'].cat.categories.shape
       (51,)
```

Something we can clearly observe is that, that is a lot of categories, for specially the book_title, this means that if we use OneHotEncoding, we will get those many columns added to the dataset. This does make it really difficult to train a model, scipy has a csr representation (Compressed Sparse Row Matrix) which does help to a lot of extend, but still this is the biggest issue that has to be addressed.

We can take inspiration from text classification models and NLP (Natural Language Processing), what is done for it, is that each of the words in dictionary is a sparse matrix, and then a model is used to convert this sparse matrix into a dense matrix, for example lets say the word `"the"` is represented as `[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`, if we use a word embedding model, such as from spacy, we would get something like `[0.45673, 0.67395, 0.98729]` which is a much better representation as before, this matrix can then be used to train. This inspiration was tried in the clustering algorithms in 1.3, and it quite did not produce good results. But I would blame the dataset for that. On top of giving the dense matrix, we can further reduce the dimensionality by using SVD (Singular Value Decomposition).

I had a hunch that if I were to use GPT-2 or BERT, the model would have been really great (although it might overfit, but still), but due to time constrains, is wasn't possible, but it would be a good exploration domain.

1.4.1 Conclusion

Never rely on SciKit Learn, the algorithms are really slow (except K-Mean) and are not quite optimized enough, instead `C++` backend algorithms give much better speed.

Also, when a classification model has to be made, in which there is a lot of attributes in text in them, it would be better, to just convert those specific attributes to a dense matrix using a pretrained word embedding model, or an even better improvement can be to use `n-gram` models.

To conclude, we were not really successful in creating a good classification model for the given dataset, but we were definitely able to compare different supervised models, unsupervised models and numeric scaling techniques, overall, the outcome of this course was covered.

2 Appendix