DATA SCIENCE APPLICATIONS

Assignment 1

Purpose of this document

The overall objective of this assignment was to produce classification predictions and compare them; analyze the pros and cons of algorithms and generate and communicate the insights.

Project Group 2

Mohamed Abdalla

Stephanie Kahindo

Raha Salimi

Fatemeh Soltani

Table Of Content

1 Introduction

The purpose of this assignment is to produce classification predictions and compare them; analyze pros and cons of methods and different algorithms and generate and communicate the insights.

2 Required Libraries

These are the list of libraries we need to complete the project.

```
# Data manipulation libraries
import numpy as np
import pandas as pd
import string
import textwrap
import re
import spacy
import sklearn.manifold
import collections

# Natural Language Processing libraries
import nltk
nltk.download('gutenberg')
nltk.download('stopwords')
nltk.download('punkt')
nltk.download ('wordnet')
from nltk.corpus import gutenberg as gut
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.collocations import *
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

# Scikit-learn Packages
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.decomposition import LatentDirichletAllocation
from gensim.models import Word2Vec
from spacy import displacy

# Import Machine Learning Models
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
from wordcloud import WordCloud
from ast import keyword
import random
import itertools
import matplotlib.pyplot as plt
import seaborn as sns
import wordcloud
import string

import gensim
from gensim import corpora
from gensim.corpora import Dictionary
from gensim.models import Phrases, LdaModel
!pip install lda-classification tomotopy
from tomotopy import HDPModel
from lda_classification.model import TomotopyLDAVectorizer
```

Figure1: Environmental setup

3 Data Preparation

3.1 Data

The assignment is to classify books by authors. This was done by choosing 5 different samples of Gutenberg digital eBooks. They were chosen in the categories of Drama and Tragedy. The table below shows the names of the authors of each book chosen:

#	Book Name	Genre	Author
1	The Tragedy of Macbet	Tragedy/Drama	William Shakespeare
2	King Arthur's Socks and Other Village Plays	Drama	Floyd Dell
3	The Road to Damascus	Drama	August Strindberg
4	Queen Mary and Harold	Drama	Alfred Lord Tennyson
5	Romeo and Juliet	Tragedy/Drama	William Shakespeare

Table1: Gutenberg Digital Books

3.1 Data Preparation, Processing and Cleansing

Data Cleaning: The data selected from each book tokenization and cleaned by some functions. We followed these steps to clean the data:

- Tokenize the data
- Remove Punctuation
- Use stemming and Lemmatisation
- Remove English stop words from the text.
- Remove all the special characters
- Convert all the characters to lowercase

Then raw data is categorised into cleaned data and a data set with reduced number of words for further usage into Bow, TF-IDF, doc2vec and LDA.

Data Processing: We chose 5 different eBooks from Gutenberg library, then Created 200 documents of each book, and each document has 100 words. Additionally, we labeled each document with their authors' name.

3.3 Data Visualization

After extracting data from books, we have extracted the top 10 commonly used words for each book. We then visualized them by using FreqDist and word cloud visualization.

4 Feature Engineering

We have implemented Bag of words, Term Frequency-Inverse Document Frequency, Doc2vec and LDA to extract the features from raw data.

4.1 Transforming Data using Bag of Words (BOW):

Due to simplicity and flexibility, the BOW method is very practical and popular to be used for classification purposes. However, this method is not very fast, and it occupies a big portion of memory because there are many

zeros and spare locations in the assigned word vector. The accuracy of this model is not remarkably high, and it is about %80 for existing chose of data frame.

Pros	**Cons**
• Simple to understand and implement • Flexible text customization	• The assigned vocabulary vector is huge • Spare memory will be occupied • Lose all information about word order • No info of vocabulary meaning "John likes Mary" or "Mary likes John"

4.2 Transforming Data using TF-IDF Vectorizer:

We used the Term-frequency-inverse document frequency (TF-IDF) vectorizer to measure the word relevance in each document. With the help of TfidfTransformer, we transform the result of BOW to calculate frequently occurring words. Term frequency calculates the frequency of words and Inverse Document Frequency calculates how rare a word is in all documents. The accuracy of this model is about 60% for this project.

Pros	Cons
• Efficient, easily compute the similarity between the documents • Suitable for long document	• TF-IDF is based on the bag-of-words (Bow) model, therefore it does not capture the position in the text, semantics, co-occurrences in different documents, etc. • Cannot capture semantics.

4.3 Transforming Data using Doc2Vec:

The Doc2Vec model did surprisingly well at predicting the author. Since D2V works better with larger data, we decided to make the vector size 300. The accuracy for author's BCD is almost always above 90% but it was different with A and E since they were the same author, William Shakespeare. Accuracy for guessing correctly ranged from 82% to 94% and I believe this is because many of the words the author uses are similar and the languages are semantically very similar so the accuracy decreases. Changing the sample size also changes the result drastically, having fewer words in the dataset decreases accuracy drastically while having more words increases it.

4.4 Transforming Data using Latent Dirichlet Allocation (LDA):

LDA is a statistical model that performs grouping of words based on their frequency from a set of documents. The distribution was useful to find top topics from the books that were selected. Our top topics for all the books using LDA were words like king, love, foundation, etc., which again depends on the random samples picked at the data preparation section. However, with a large data text, it appears that this model can become ambiguous as same words can be assigned to multiple topics, and those words can have unique probabilities increasing the inaccuracy of the model.

5 Modelling

We used 3 machine learning algorithms for classification:

- **Random Forest Classifier**
- **Support Vector Machine (SVM)**
- **K-Nearest Neighbors (K-NN)**

Random Forest Classifier is a popular machine learning algorithm that belongs to the supervised learning technique. This algorithm contains several decision trees on various subsets of the dataset and takes the average to improve the predictive accuracy of that dataset.

Support Vector Machine (SVM) is a supervised machine learning model that uses classification algorithms for classification problems. Once we give the SVM model sets of labeled training data for each category, they can categorize a text. In this project, we spit the data into trains and tests and provided the test size with different values and fit the data and measured the accuracy.

K-Nearest Neighbors (K-NN) is a simple and easy to implement algorithm for both regression and classifier. The K-NN algorithm assumes that similar things exist in proximity. K-NN is known to be a non-parametric and lazy learning algorithm. Non-parametric describes there is no assumption for primary data distribution. In other words, the model structure is determined from the data provided which is the data set. Lazy algorithm does not need any training data points for model generation because it uses all the data in testing phase. This makes training faster but when it comes to testing its phase is slower. In another case, K-NN requires more time to scan all data points and scanning all data points will require more memory for storing training data.

5.1 Models Fitted with Bow

With the ten-fold cross validation the accuracy reaches the %99 which is very impressive. For this model linear regression provides the best accuracy %99.6 and KNN gave the worse accuracy %71.6.

5.2 Models Fitted with TF-IDF

We fitted TF-IDF with three machine learning models that reached the highest accuracy with Random Forest Classifier and SVM 67% and 68% respectively. The worst accuracy belongs to KNN classifier as it showed 60% accuracy.

5.3 Models fitted with Doc2Vec

For this part, we created 2 tagged documents, one for testing and one for training, and then used Doc2Vec to train the training tagged set. After that, we create a function that gets the inferred vector with the target and words. We then split the words and targets to test and train. We later call a function that loops through a list of models and fits the testing and training data on those models and does tenfold cross-validation and prints the accuracies for each model. Random Forest and Linear regression, and KNN performed the best respectively while SVM performed the worst. This was surprising since SVM is known to be accurate for text classification.

	Accuracy
RF	0.99
LR	1.00
SVM	0.54
KNN	0.94

```
1 print(classification_report(y_test, predictions["RF"]))
2
```

	precision	recall	f1-score	support
lfred Lord Tennyson	1.00	1.00	1.00	21
August Strindberg	0.90	1.00	0.95	19
Floyd Dell	1.00	1.00	1.00	23
William Shakespeare	1.00	0.95	0.97	37
accuracy			0.98	100
macro avg	0.98	0.99	0.98	100
weighted avg	0.98	0.98	0.98	100

```
1 print(classification_report(y_test, predictions["LR"]))
```

	precision	recall	f1-score	support
lfred Lord Tennyson	1.00	1.00	1.00	21
August Strindberg	1.00	1.00	1.00	19
Floyd Dell	1.00	1.00	1.00	23
William Shakespeare	1.00	1.00	1.00	37
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

```
1 print(classification_report(y_test, predictions["SVM"]))
```

	precision	recall	f1-score	support
Alfred Lord Tennyson	1.00	1.00	1.00	21
August Strindberg	1.00	0.95	0.97	19
Floyd Dell	0.96	1.00	0.98	23
William Shakespeare	1.00	1.00	1.00	37
accuracy			0.99	100
macro avg	0.99	0.99	0.99	100
weighted avg	0.99	0.99	0.99	100

6. Error Analysis:

6.1 Models Fitted without removing StopWords and Punctuations

Result:

the model accuracy without removing stopwords and punctuation

	Accuracy
RFC	0.20
SVM	0.19
KNN	0.17

6.2 Models Fitted with a dataset containing a smaller number of words

Result:

the model accuracy with less number of words

	Accuracy
RFC	0.475
SVM	0.505
KNN	0.430

6. RESULTS

Algorithms	***Random Forest Classifier***	***Support Vector Machine Classifier***	***K-Nearest Neighbors Classifier***
BOW	**0.30**	**0.37**	**0.22**
TF-IDF	**0.72**	**0.72**	**0.62**
Doc2Vec	**0.99**	**-**	**0.94**
BOW without removing *StopWords and Punctuations*	**0.20**	**0.19**	**0.17**
A dataset with a fewer number of words per sample	**0.47**	**0.50**	**0.43**

7. Conclusion

Champion Model

We decided that RF would be our champion model based on the cross-validation accuracy of each model. We also decided to partially hyperparameter -optimize the model but didn't continue as the score of the test data was above 98% and specifying our combinations' hyperparameters would've probably increased accuracy but would have a higher runtime. We also decided to do 2-fold cross-validation since it took less time to run.

References

- https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
- https://machinelearningmastery.com/confusion-matrix-machine-learning/
- https://medium.com/nlplanet/two-minutes-nlp-doc2vec-in-a-nutshell-25be546a8342
- https://machinelearningmastery.com/gentle-introduction-bag-words-model/
- https://towardsdatascience.com/a-machine-learning-approach-to-author-identification-of-horror-novels-from-text-snippets-3f1ef5dba634
- https://www.javatpoint.com/machine-learning-random-forest-algorithm
- https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761