Task 1: Creating a Sentiment Prediction Platform (50/100) [max. 3 pages]

We consider a real-life dataset consisting of 50,000 labeled gourmet food reviews from Amazon extracted from the work of McAuley and Leskovec[1].

The goal is to train machine learning (ML) models for sentiment prediction, evaluate them, and deploy the best model on a platform. The result should be similar to the service proposed on the following website: https://monkeylearn.com/sentiment-analysis-online/

Subtasks:

All coding material is inspired by the CS5079 lecture and practical materials unless it is specified in the text.

1.1) Using your own words, please explain the several steps that you will need to go through to create your sentiment prediction platform. Your description should include (but not only) the following points (**2 marks**):

1-Data retrieval, 2-Exploratory Data Analysis (EDA), 3-Pre-processing, 4-Train and test data sets creating, 5-Feature Extraction and Feature Engineering, 6-Training model, 7-Model prediction, 8-Model Evaluation, 9-Deployment

1.2) Please provide a short description of the dataset provided, along with how you imported the data, providing snippets of code along with a detailed description **(2 marks).**

We are provided with a real-life dataset consisting of 50,000 labelled gourmet food reviews from Amazon (25,000 positive in the *positive_reviews.csv* file and 25,000 negative reviews in the negative_reviews.csv file). The dataset is balanced (negative_reviews=positive=25,000) and each food review is labelled as 1 (positive) and 0 (negative), no neutral comments included. The data sets imported from .csv files into the 'pandas' data frames (code snippet can be found in Appendix 1.2.1 Data Import).

1.3) Employ exploratory data analysis (EDA) techniques to gain an initial understanding of the data. Please provide appropriate visualization results and initial insights gained from EDA (**4 marks**).

From printing info on the data set (Appendix 1.3.1 Info) we can see that we have imported d ata in two data frames, one for the positive reviews and one for negative. Each data frame h as two columns "Review" and "Label" and 25,000 rows. The data types for "Review" is objec t and for "Label" is int64. There are no missing values. From the "Review" describe comman d (Appendix 1.3.2 Describe) we can see that only 23,755 positive and 21,682 negative comm ents are unique. Also, this command shows us the most frequent comment for each set as t he top and its frequency in its set. Also, I made the comment length analysis (Appendix 1.3.3 Max, mean, min) which shows max length of positive comment is 10,112 characters, negativ e 9,296, minimum length of positive comment is 45 and negative is 33, mean length of positi ve comment is `420.4514` and negative is `495.21012`.

1.4) Motivate, explain, and apply any necessary pre-processing techniques on your food reviews. Using an example, show how a string is transformed after each processing step. **(6 marks)**

The pre-processing in this case means the text preparation process where we are going to remove the unnecessary or meaningless for machine analysis text elements. We are going to take the third element from the positive comments data set, example original text and text for each stage can be found at the Appendix 1.4.1 Text pre-processing.

Step 1 is an html tags removing, you can see tags like
 removed from the example text.

Step 2 is an accented character remove, which replaces the characters like “á” with a for example, and change superscript and subscript into the normal text.
Step 3 is to expand contraction which would replace contraction with its long form. For example: don’t with do not, I’d with I would etc with the help of contraction library.
Step 4 is to make all text to the lower case, which is improve the text readability for machine.
Step 5 is to remove any extra new line, literally removes any new line symbols and straighten the string. This is to improve readability.
Step 6 is to insert spaces between the special characters. This step is to make special characters separate from the text and make them ready for removal.
Step 7 is to lemmatize text, i.e., change each word into its pronoun form for example doing will be replaced with do.
Step 8 is to remove special characters, which were separated from the words on step 6.
Step 9 is to remove extra whitespace, replace all spaces where we have more than one whitespace with one whitespace it makes sense considering steps 6 and 8.
Step 10 is to remove stopwords. This removes stopwords from the corpus such as a, the, and etc. Before doing this, we removed ‘no’ and ‘not’ from the nltk list of stopwords, as they change the meaning to the opposite, and we need them.

1.5) Implement the following techniques for sentiment prediction:
• Logistic regression with BOW and TF-IDF word features
• Support vector machine with BOW and TF-IDF word features
• Long-short term memory network with an embedding layer
For each of them, describe in detail how you deployed them and adjusted their parameters, going into detail on what each parameter does as well. You may use open source code and libraries as long as you acknowledge them (**14 marks**).
The Bag of Words starts from feature extraction and engineering. The BOW is one of the simplest methods to transform text into the vectors, where we are not taking into the account the order of words. The BOW just counts the number of appearances of each word in each review. For feature extraction we are using the CountVectorizer (Appendix 1.5.1 BOW feature engineering). The parameters binary=False, means each word is an integer, when True all non-zero are set to 1, min_df and max_df to ignore items frequency below and above, ngram_range=(1,2) means using both unigrams and bigrams. We can now transform and fit our reviews into the logistic regression (LR) and support vector machine (SVM) with stochastic gradient descent (SGD) models (Appendix 1.5.2 BOW training). The max number of iterations is set to 100 for both models, for lr the penalty is specified that “l2” is used as the regularization, the parameter C in LR specifies the inversion strength here as default 1. The SVM is defined with SGDClassifier with loss equals to hinge.

The Team Frequency-Inverse Document Frequency (TF-IDF) tries to evaluate the word importance for a review in our corpus. For feature extraction we are using the TfidfVectorizer. The parameter use_idf=True, enables inverse-document-frequency reweighting. The sublinear_tf=True, enables the sublinear tf scaling (replaces tf with 1+log(tf)). The rest of the parameters are set analogical to the BOW method (Appendix 1.5.3 TF-IDF feature engineering). We are ready to train the SVM and LR models described above with TF-IDF feature vectors.

The Long-Short memory network (LSTM) with embedding layer is a variant of Recurrent Neural Network (RNN) that is learning the relations between elements (in our case words) in an input sequence. We are engineering our features with use of keras library Tokenizer (Appendix 1.5.4 LSTM feature engineering) with the feature limit 1000 which means that each review will be split in 1000 features (can be increased considering data exploration but takes much more time to process). The LSTM model (Appendix 1.5.4 LSTM model) parameters embedded_dim, lstm_out, batch_size and dropout are selected intuitively and can be adjusted for better result. The embedding layer learns a word embedding for all the words in corpus. The embedding layer has parameters max_features represents the input dimension in our case the 1000 words, the data input length where we have 35000 reviews, and embed_dim=64 which defines the vector length to hold our word coordinates. The LSTM layer has 200 memory units…….

1.6) Split your dataset into a training and test set of size 35,000 and 15,000 respectively. Train and evaluate the performance of the techniques developed in task 1.5. Please present and discuss your results using metrics and/or tables. (**8 marks**)
The data split I did by taking the 7500 of each positive and negative reviews into the testing set and the rest 35000 placed into the training set (the code snippet in Appendix 1.6.1 Train and test data sets). After splitting I transformed the data from data frame into the numpy array (the code snippet in Appendix 1.6.2 Train and test arrays). The table below represents the metrics results for each model, code snippets can be found in Appendix 1.6.3 Results

	accuracy	precision	recall	F1-score
BOW with LR	0.876	0.876	0.876	0.876
BOW with SVM	0.869	0.869	0.869	0.869
TF-IDF with LR	0.865	0.865	0.865	0.865
TF-IDF with SVM	0.865	0.865	0.865	0.865
LSTM	0.832	0.832	0.832	0.832

The linear regression with BOW model has the best results, but the LSTM can be improved with parameters manipulating and perform similar or maybe even better than other models.

1.7) Select and save the “best” model, then deploy it on an online platform of your choice. The end-user should be able to input a string of text and receive its polarity (along with a confidence score). The report should contain the URL of the online platform along with detailed explanations and screenshots. You may use Flask or Django and services like Heroku.
This task as not as difficult as time consuming when I am doing it for the first time. So, considering the time left I am going to write the plan on what I need to do. I need to separate the ‘best’ method into the separate file and save the model using joblib or pickle python library. After that I need to make the python application for the model deployment using Flask or Django library along with the .html file which will mark-up our application user interface on the web. Next steps are saving this project on GitHub, create an account on Heroku, Link GitHub and Heroku, deploy the model. Merry Christmas and Happy New Year!!!

Appendix 1.2.1 Data Import

```
#We import the dataset
negative_reviews_raw = pd.read_csv("D:/MScAI/Semester1/CS5079AppliedAI/Assessm3/negative.csv")
positive_reviews_raw = pd.read_csv("D:/MScAI/Semester1/CS5079AppliedAI/Assessm3/positive.csv")
print(negative_reviews_raw.head(), negative_reviews_raw.shape)
print(positive_reviews_raw.head(), positive_reviews_raw.shape)
```

```
                                              Review  Label
0  We love Malibu Rum but they sure missed the ma...      0
1  I just wanted to say that if you want to get y...      0
2  These seeds were accompanied by small broken p...      0
3  Way way way overpriced. I can get this same se...      0
4  I bought these on the strength of the reviews....      0 (25000, 2)
                                              Review  Label
0  Better than Wolff's Kasha. I grew up eating Ka...      1
1  It was such good product. Came in two differen...      1
2  MMMM Yes all chocolate is good.<br />But some ...      1
3  This is, as all of their cereals I've ordered ...      1
4  Whoever Photoshopped the cookie on the front o...      1 (25000, 2)
```

Appendix 1.3.1 Info

```
negative_reviews_raw.info()
positive_reviews_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  25000 non-null  object
 1   Label   25000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 390.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  25000 non-null  object
 1   Label   25000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 390.8+ KB
```

Appendix 1.3.2 Describe

```
positive_reviews_raw['Review'].describe()
```

```
count                                                25000
unique                                               23755
top       I'm addicted to salty and tangy flavors, so wh...
freq                                                     6
Name: Review, dtype: object
```

```
negative_reviews_raw['Review'].describe()
```

```
count                                                25000
unique                                               21682
top       This review will make me sound really stupid, ...
freq                                                    29
Name: Review, dtype: object
```

Appendix 1.3.3 Max, mean, min

```
print (positive_reviews_raw.Review.map(lambda x: len(x)).max())
print (negative_reviews_raw.Review.map(lambda x: len(x)).max())
```

```
10112
9296
```

```
print (positive_reviews_raw.Review.map(len).mean())
print (negative_reviews_raw.Review.map(len).mean())
```

```
420.4514
495.21012
```

```
print (positive_reviews_raw.Review.map(len).min())
print (negative_reviews_raw.Review.map(len).min())
```

```
45
33
```

Appendix 1.4.1 Text pre-processing

Original text from positive_reviews_raw.Review[2]: 'MMMM Yes all chocolate is good.
But some chocolates are better than others and this is a "better than others chocolate"
It has a smooth, easy creamy taste not overly sweet IMO.
It is just pretty much perfect'
Step 1: MMMM Yes all chocolate is good.But some chocolates are better th an others and this is a "better than others chocolate"It has a smooth, easy creamy taste not overly sweet IMO.It is just pretty much perfect
Step 2: MMMM Yes all chocolate is good.But some chocolates are better th an others and this is a "better than others chocolate"It has a smooth, easy creamy taste not overly sweet IMO.It is just pretty much perfect
Step 3: MMMM Yes all chocolate is good.But some chocolates are better th an others and this is a "better than others chocolate"It has a smooth, easy creamy taste not overly sweet IMO.It is just pretty much perfect
Step 4: mmmm yes all chocolate is good.but some chocolates are better th an others and this is a "better than others chocolate"it has a smooth, easy creamy taste not overly sweet imo.it is just pretty much perfect
Step 5: mmmm yes all chocolate is good.but some chocolates are better th an others and this is a "better than others chocolate"it has a smooth, easy creamy taste not overly sweet imo.it is just pretty much perfect
Step 6: mmmm yes all chocolate is good . but some chocolates are better than others and this is a "better than others chocolate"it has a smooth , easy creamy taste not overly sweet imo . it is just pretty much perfe ct
Step 7: mmmm yes all chocolate be good . but some chocolate be well than other and this be a " well than other chocolate"it have a smooth , easy creamy taste not overly sweet imo . it be just pretty much perfect
Step 8: mmmm yes all chocolate be good but some chocolate be well than other and this be a well than other chocolateit have a smooth easy cr eamy taste not overly sweet imo it be just pretty much perfect
Step 9: mmmm yes all chocolate be good but some chocolate be well than o ther and this be a well than other chocolateit have a smooth easy cream y taste not overly sweet imo it be just pretty much perfect
Step 10: mmmm yes chocolate good chocolate well well chocolateit smooth easy creamy taste not overly sweet imo pretty much perfect

Appendix 1.5.1 BOW feature engineering

```
# Libraries for feature engineering
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# ML Models
from sklearn.linear_model import SGDClassifier, LogisticRegression
```

```
# build BOW features on train reviews
cv = CountVectorizer(binary=False, min_df=0.0, max_df=1.0, ngram_range=(1,2))
cv_train_features = cv.fit_transform(X_train)
```

```
# transform test reviews into features
cv_test_features = cv.transform(X_test)
```

```
print('BOW model:> Train features shape:', cv_train_features.shape, ' Test features shape:', cv_test_features.shape)
```

```
BOW model:> Train features shape: (35000, 623304)  Test features shape: (15000, 623304)
```

Appendix 1.5.2 BOW LR and SVM training

```
# We define our SVM and LR models
lr = LogisticRegression(penalty='l2', max_iter=1000, C=1)
svm = SGDClassifier(loss='hinge', max_iter=100)
```

```
# Logistic Regression model on BOW features
lr.fit(cv_train_features,y_train)
```

Appendix 1.5.3 TF-IDF feature engineering

```
# build TFIDF features on train reviews
tv = TfidfVectorizer(use_idf=True, min_df=0.0, max_df=1.0, ngram_range=(1,2),sublinear_tf=True)
tv_train_features = tv.fit_transform(X_train)
```

```
tv_test_features = tv.transform(X_test)
```

```
print('TFIDF model:> Train features shape:', tv_train_features.shape, ' Test features shape:', tv_test_features.shape)
```

```
TFIDF model:> Train features shape: (35000, 623304)  Test features shape: (15000, 623304)
```

```
# Logistic Regression model on TF-IDF features
lr.fit(tv_train_features,y_train)
```

Appendix 1.5.4 LSTM feature engineering

```
# Truncate and pad the review sequences
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.utils.np_utils import to_categorical

max_features = 1000
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(X_train)
tokenizer.fit_on_texts(X_test)
X_train_token = tokenizer.texts_to_sequences(X_train)
X_train_token = pad_sequences(X_train_token, maxlen=max_features)

X_test_token = tokenizer.texts_to_sequences(X_test)
X_test_token = pad_sequences(X_test_token, maxlen=max_features)
Y_train = to_categorical(y_train)
Y_test = to_categorical(y_test)

print(X_train_token.shape,Y_train.shape)
print(X_test_token.shape,Y_test.shape)
```

```
(35000, 1000) (35000, 2)
(15000, 1000) (15000, 2)
```

Appendix 1.5.4 LSTM model

```
embed_dim = 64
lstm_out = 200

model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length = X_train_token.shape[1]))
model.add(LSTM(lstm_out))
model.add(Dense(2,activation='sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 1000, 64)          64000
_________________________________________________________________
lstm_10 (LSTM)               (None, 200)               212000
_________________________________________________________________
dense_10 (Dense)             (None, 2)                 402
=================================================================
Total params: 276,402
Trainable params: 276,402
Non-trainable params: 0
_________________________________________________________________
None
```

Appendix 1.6.1 Train and test data sets

```
X_test_df = pd.concat([normalized_positive_reviews.Review.iloc[:7500],
                       normalized_negative_reviews.Review.iloc[:7500]],
                       ignore_index=True)
y_test_df = pd.concat([normalized_positive_reviews.Label.iloc[:7500],
                       normalized_negative_reviews.Label.iloc[:7500]],
                       ignore_index=True)
X_train_df = pd.concat([normalized_positive_reviews.Review.iloc[7500:42500],
                        normalized_negative_reviews.Review.iloc[7500:42500]],
                        ignore_index=True)
y_train_df = pd.concat([normalized_positive_reviews.Label.iloc[7500:42500],
                        normalized_negative_reviews.Label.iloc[7500:42500]],
                        ignore_index=True)

print(X_test_df.shape, y_test_df.shape)
print(X_train_df.shape, y_train_df.shape)
X_train_df=X_train_df.astype('str')
X_train_df
```

```
(15000,) (15000,)
(35000,) (35000,)

0        always favorite coffee cremor make coffee smoo...
1        cat discern palate eat fancy feast fragrant de...
2        first attempt gluten free bread make regular b...
3        fiasconaro panettone make wonderful christmas ...
4        sparkle blackberry not get cola hope nice drin...
                               ...
34995    many glowing review feel little ashamed put lu...
34996    give 3 star purpose bake mix really good panca...
34997    affordable bully stick office fill incredible ...
34998    get daughter love blood hot chocolate get stuf...
34999    not enough flavor murky brown color lack aroma...
Name: Review, Length: 35000, dtype: object
```

Appendix 1.6.2 Train and test arrays

```
X_test = np.array(X_test_df)
y_test = np.array(y_test_df)
X_train = np.array(X_train_df)
y_train = np.array(y_train_df)
print (len(X_test), len(y_test))
print (len(X_train), len(y_train))
X_test
```

```
15000 15000
35000 35000
```

Appendix 1.6.3 Results

```
# Logistic Regression model on BOW features
lr.fit(cv_train_features,y_train)
y_predicted = lr.predict(cv_test_features)

print("The model accuracy score is: {}".format(accuracy_score(y_test, y_predicted)))
print("The model precision score is: {}".format(precision_score(y_test, y_predicted, average="weighted")))
print("The model recall score is: {}".format(recall_score(y_test, y_predicted, average="weighted")))
print("The model F1-score is: {}".format(f1_score(y_test, y_predicted, average="weighted")))

print(classification_report(y_test, y_predicted))

display(pd.DataFrame(confusion_matrix(y_test, y_predicted), columns=["Pred. negative", "Pred. positive"], i
```

```
The model accuracy score is: 0.876
The model precision score is: 0.876000240640154
The model recall score is: 0.876
The model F1-score is: 0.8759999801599969
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      7500
           1       0.88      0.88      0.88      7500

    accuracy                           0.88     15000
   macro avg       0.88      0.88      0.88     15000
weighted avg       0.88      0.88      0.88     15000
```

	Pred. negative	Pred. positive
Act. negative	6573	927
Act. positive	933	6567

```
# SVM model on BOW
svm.fit(cv_train_features,y_train)
y_predicted = svm.predict(cv_test_features)

print("The model accuracy score is: {}".format(accuracy_score(y_test, y_predicted)))
print("The model precision score is: {}".format(precision_score(y_test, y_predicted, average="weighted")))
print("The model recall score is: {}".format(recall_score(y_test, y_predicted, average="weighted")))
print("The model F1-score is: {}".format(f1_score(y_test, y_predicted, average="weighted")))

print(classification_report(y_test, y_predicted))

display(pd.DataFrame(confusion_matrix(y_test, y_predicted), columns=["Pred. negative", "Pred. positive"], i
```

```
The model accuracy score is: 0.8690666666666667
The model precision score is: 0.8691519555344733
The model recall score is: 0.8690666666666667
The model F1-score is: 0.869059103520486
              precision    recall  f1-score   support

           0       0.86      0.88      0.87      7500
           1       0.87      0.86      0.87      7500

    accuracy                           0.87     15000
   macro avg       0.87      0.87      0.87     15000
weighted avg       0.87      0.87      0.87     15000
```

	Pred. negative	Pred. positive
Act. negative	6575	925
Act. positive	1039	6461

```
# Logistic Regression model on TF-IDF features
lr.fit(tv_train_features,y_train)
y_predicted = lr.predict(tv_test_features)

print("The model accuracy score is: {}".format(accuracy_score(y_test, y_predicted)))
print("The model precision score is: {}".format(precision_score(y_test, y_predicted, average="weighted")))
print("The model recall score is: {}".format(recall_score(y_test, y_predicted, average="weighted")))
print("The model F1-score is: {}".format(f1_score(y_test, y_predicted, average="weighted")))

print(classification_report(y_test, y_predicted))

display(pd.DataFrame(confusion_matrix(y_test, y_predicted), columns=["Pred. negative", "Pred. positive"],
```

```
The model accuracy score is: 0.8651333333333333
The model precision score is: 0.8652661213673928
The model recall score is: 0.8651333333333333
The model F1-score is: 0.8651210749371617
              precision    recall  f1-score   support

           0       0.86      0.87      0.87      7500
           1       0.87      0.86      0.86      7500

    accuracy                           0.87     15000
   macro avg       0.87      0.87      0.87     15000
weighted avg       0.87      0.87      0.87     15000
```

	Pred. negative	Pred. positive
Act. negative	6560	940
Act. positive	1083	6417

```
#SVM model on TF-IDF
svm.fit(tv_train_features,y_train)
y_predicted = svm.predict(tv_test_features)

print("The model accuracy score is: {}".format(accuracy_score(y_test, y_predicted)))
print("The model precision score is: {}".format(precision_score(y_test, y_predicted, average="weighted")))
print("The model recall score is: {}".format(recall_score(y_test, y_predicted, average="weighted")))
print("The model F1-score is: {}".format(f1_score(y_test, y_predicted, average="weighted")))

print(classification_report(y_test, y_predicted))

display(pd.DataFrame(confusion_matrix(y_test, y_predicted), columns=["Pred. negative", "Pred. positive"],
```

```
The model accuracy score is: 0.8652
The model precision score is: 0.865370467245198
The model recall score is: 0.8652
The model F1-score is: 0.865184275093847
              precision    recall  f1-score   support

           0       0.86      0.88      0.87      7500
           1       0.87      0.85      0.86      7500

    accuracy                           0.87     15000
   macro avg       0.87      0.87      0.87     15000
weighted avg       0.87      0.87      0.87     15000
```

	Pred. negative	Pred. positive
Act. negative	6570	930
Act. positive	1092	6408

In [92]:
```
# LSTM model prediction on test data
y_pred_lstm = model.predict(X_test_token)
```

In [93]:
```
#LSTM model evaluation

print("The model accuracy score is: {}".format(accuracy_score(Y_test, y_pred_lstm.round())))
print("The model precision score is: {}".format(precision_score(Y_test, y_pred_lstm.round(), average="weighted")))
print("The model recall score is: {}".format(recall_score(Y_test, y_pred_lstm.round(), average="weighted")))
print("The model F1-score is: {}".format(f1_score(Y_test, y_pred_lstm.round(), average="weighted")))

print(classification_report(Y_test, y_pred_lstm.round()))

display(pd.DataFrame(confusion_matrix(y_test,np.argmax(y_pred_lstm.round(), axis=1)), columns=["Pred. negative", "
```

```
The model accuracy score is: 0.8321333333333333
The model precision score is: 0.8321622683575991
The model recall score is: 0.8321333333333333
The model F1-score is: 0.8321296774907542
              precision    recall  f1-score   support

           0       0.83      0.84      0.83      7500
           1       0.84      0.83      0.83      7500

   micro avg       0.83      0.83      0.83     15000
   macro avg       0.83      0.83      0.83     15000
weighted avg       0.83      0.83      0.83     15000
 samples avg       0.83      0.83      0.83     15000
```

	Pred. negative	Pred. positive
Act. negative	6276	1224
Act. positive	1294	6206