Task 6 Report

Review Text Preparation:

- Converting all text to lowercase
- Removing stopwords using genism preprocessing remove_stopwords module
- Using re to remove all non-alphabetical characters in the text and extra spaces between words
- Using WordNet lemmatizer to lemmatise words, and re-joining tokenised text into strings

Additional Features:

- A keyword csv was created, which contained two lines – one for 'unhygienic' related words, the other for 'hygienic' related words. This csv of keywords was used to create a list of labels for each restaurant containing 0's and 1's for every time an unhygienic- or hygienic-related word was found in the text. Labels were then summed and an average 'hygiene keyword' score calculated. For restaurants with reviews absent of any of the keywords were given a score of -1
- Review count, average star rating, categories and location features was also extracted for each restaurant

Train-Test split:

To effectively test the final model, the data was split 80:20 (train:test, which was 436 and 100 restaurants respectively) using sklearns model_selection train_test_split module. The test set was not used at any stage of investigating models (K Fold Cross Validation was used in initial model testing). The test set was only used at the very end to test the final model once it had been trained on the training data. This was to keep the models blind to the test data, prevent data leakage and ensure model testing was as accurate to new data as possible, so applicable to new incoming real-world data.

Methods:

11 different classifiers were used throughout, which were Multinomial Naïve Bayes, Logistic Regression, Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), Random Forest, Extra Trees, Gradient Boosting, K Nearest Neighbor (KNN), AdaBoost, XGBoost, Multi-Layer Perceptron (MLP). All classifiers (xgboost) were sklearn algorithms.

1. To first establish which text processing methods generated the best F1 score for classifiers, review text was processed in different ways and then using K Fold Cross Validation with pipelines (containing text processor and classifier) to prevent data leakage, an average F1 score for each classifier was calculated. Text processing methods were as follows:
    a. Count Vectorizer (sklearn) – none of the classifiers performed very well with this as input
    b. TFIDF unigrams – Naïve Bayes, Random Forest and Extra Trees performed best with this as input
    c. TFIDF bigrams – KNN and MLP performed best with this as input
    d. TFIDF trigrams – none of the classifiers performed very well with this as input

e. TDIDF ngrams (containing unigrams, bigrams and trigrams) - XGBoost, SGD, Logistic Regression, Gradient Boosting, SVM and AdaBoost performed best with this as input
f. LDA vectors – none of the classifiers performed very well with this as input

It is not surprising that classifiers do not perform well with Count Vectorizer; this text representation only provides a simple word count, without taking into account different frequencies of each word per restaurant. It does not provide classifiers with a very informative input.

TFIDF provides classifiers with much more informative input by indicating which terms are important for each restaurant. TFIDF trigrams most likely are not as useful as input because as ngram length increases, the information that they provide falls and becomes less meaningful (a larger number of options than single words and bigrams, but each occurring with less frequency, and so are less comparable across documents).

LDA is possibly not a particularly useful input as the topics it establishes are likely to have more to do with content of reviews rather than words used, for example cuisine or service. Performance may have been improved by experimenting with the number of topics, which may have drawn out one related to hygiene. But as noted in additional features keyword extraction, not all restaurants contain reviews that allude to hygiene.

2. To then establish the most effective way of preprocessing the additional feature categories for classifiers, one hot encoding and then label encoding was used on the categorical features restaurant categories and postcode. Once again pipelines (with MinMaxScaler from sklearn.preprocessing) and K Fold Cross Validation was used with each method, with an average F1 score calculated, for each classifier. All classifiers performed better with categorical features encoded with one-hot encoding.

3. Three data inputs were then created based on the text processing methods that performed best in the first stage. The one-hot encoded additional features was combined with the TFIDF unigram text, with the TFIDF bigram text and the TFIDF ngram text. GridSearchCV (sklearn, with scoring='f1', cv=5) was then used with each classifier-data input combination in order to establish the hyperparameters and data input combinations that generated the highest F1 score for each classifier. The following was found:

Classifier	Optimal data input	Optimal hyperparameters	F1 score
Multinomial Naïve Bayes	TFIDF Unigrams		0.693
Logistic Regression	TFIDF Ngrams	C=1.0 Penalty='l2'	0.650
Random Forest	TFIDF Ngrams	Criterion='entropy' Max_depth=15 Max_features=None Min_samples_leaf=1 Min_samples_split=2 N_estimators=200	0.637
Extra Trees	TFIDF Unigrams	Criterion='gini' Max_depth=25	0.670

		Max_features='auto' Min_samples_leaf=1 Min_samples_split=2 N_estimators=200	
Gradient Boosting	TFIDF Unigrams	Max_depth=3 N_estimators=150 Subsample=0.5	0.649
XGBoost	TFIDF Bigrams	Colsample_bytree=0.8 Gamma=0.2 Max_depth=5 Min_child_weight=1 Reg_alpha=1 Subsample=0.7	0.668
KNN	TFIDF Unigrams	N_neighbors=21 P=2	0.649
AdaBoost	TFIDF Unigrams	N_estimators=200	0.624
SVM	TFIDF Ngrams	C=10 Gamma=0.1 Kernel='sigmoid'	0.667
SGD	TFIDF Bigrams	Alpha=1 Loss='log' Penalty='l2'	0.681
MLP	TFIDF Unigrams	Activation='identity' Alpha=0.01 Early_stopping=True Max_iter=5000 Solver='adam'	0.672

The best performed in terms of F1 score was Multinomial Naïve Bayes.

From the above, it is apparent that although it has been previously established that TFIDF text representation (in uni-, bi, or n-gram form) is best, different classifiers perform better with different TFIDF gram representations.

On each of the best performing classifier hyperparameters-data input combinations, further evaluation was carried out – including F1 scores, classification reports and confusion matrices – once again with K Fold Cross Validation in order to establish

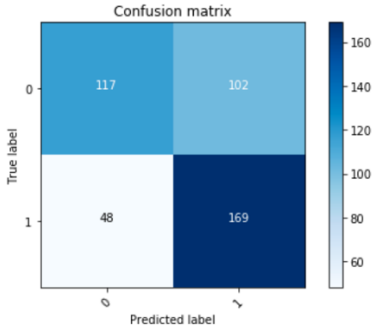
strengths and weaknesses of each. For example, for Naïve Bayes classifier:

```
5-Fold Cross-Validation with Naive Bayes Classifier
Average f1 score from 5-fold cross-validation: 0.6929068562133219
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.53      0.61       219
           1       0.62      0.78      0.69       217

    accuracy                           0.66       436
   macro avg       0.67      0.66      0.65       436
weighted avg       0.67      0.66      0.65       436

Confusion Matrix Values:
[[117 102]
 [ 48 169]]
```



From this evaluation, Naive Bayes, SGD and SVC all performed relatively well for overall F1, but whilst all have amongst the best performances with true positives and false negatives, they all perform poorly in true negatives and false positives. On the other hand, KNN, Random Forest and XGBoost show the reverse pattern - they perform better with true negatives and false positives, and worse with true positives and false negatives

4. In order to take advantage of the differences in performance, predictions made with the classifiers by K Fold Cross Validation were then stacked, and different combinations of these predictions used as input with Logistic Regression used as the second stage classifier to make the final predictions. Rather than using an existing stacking algorithm, my own was created to enable each classifier to be used with its optimal data input (for sklearn and mlxtend versions, only a single data version input is possible). Two stacking methods were tested:
    a. Stacking of label predictions: the best performing stack was Multinomial Naïve Bayes, AdaBoost, SGD, SVM (weighted by 3), and Random Forest, which achieved an F1 score of 0.694 (a marginal improvement on the score achieved by Multinomial Naïve Bayes alone). This was the model used as the final model
    b. Stacking of label prediction probabilities: no combination performed as well as the first stacking version

Final Model:

Text processing for input was as specified as above – conversion to lowercase, stopword and non-alphabetical character removal, lemmatisation. Additional features were also as above – keyword score and one-hot encoding of categorical features.

Firstly, using K Fold Cross Validation and the models Multinomial Naïve Bayes (with TFIDF unigrams), AdaBoost (with TFIDF unigrams), SGD (with TFIDF bigrams), SVM (TFIDF ngrams), and Random Forest (with TFIDF ngrams) – all are sklearn models – with

their optimal hyperparameters (specified above), the training set was used to generate predictions, which were stacked (with the SVM predictions weighted by 3). The full training set was then used to train each of the 5 models, and then the stacked data from the training set was used to train the second stage Logistic Regression model (sklearn again, with default parameters).

The trained first-stage models were then used to generate predictions on the test set, which were stacked (SVM predictions weighted by 3). This stacked test data was then used with the trained Logistic Regression second stage model to generate final predictions.

The F1 score for the test set was 0.645; not as good as that achieved when testing the model with the training set only, but not too far off indicating the model did not overfit the training set. There are a number of possibilities for a slightly poorer performance:

- When searching for optimal hyperparameters, all additional features were scaled prior to GridSearch and K Fold Cross Validation. Therefore, the scaler would have been fit on all data being used to generate predictions in K Fold, whereas the scaler was used only to transform the test data input (not fitted onto it).
- The dataset is quite small. A larger dataset would be likely to greatly improve the predictive power of a model.