Task 6

Overview

Sometimes we make decisions beyond the rating of a restaurant. For example, if a restaurant has a high rating but it often fails to pass hygiene inspections, then this information can dissuade many people to eat there. Using this hygiene information could lead to a more informative system; however, it is often the case where we don't have such information for all the restaurants, and we are left to make predictions based on the small sample of data points.

In this task, you are going to predict whether a set of restaurants will pass the public health inspection tests given the corresponding Yelp text reviews along with some additional information such as the locations and cuisines offered in these restaurants. Making a prediction about an unobserved attribute using data mining techniques represents a wide range of important applications of data mining. Through working on this task, you will gain direct experience with such an application. Due to the flexibility of using as many indicators for prediction as possible, this would also give you an opportunity to potentially combine many different algorithms you have learned from the courses in the Data Mining Specialization to solve a real world problem and experiment with different methods to understand what's the most effective way of solving the problem.

Preprocessing

```
def build_collection(doc, filename):
    tok = metapy.analyzers.ICUTokenizer(suppress_tags=True)
    tok = metapy.analyzers.LowercaseFilter(tok)
    tok = metapy.analyzers.ListFilter(tok, filename,
                                      metapy.analyzers.ListFilter.Type.Reject)
    tok = metapy.analyzers.Porter2Filter(tok)
    tok.set_content(doc.content())
    return tok


def process_review(doc):
    tokens = []
    for d in doc:
        docu = metapy.index.Document()
        docu.content(d)
        tok = build_collection(docu, 'stopwords.txt')
        tokens.append(tok)
    return tokens
```

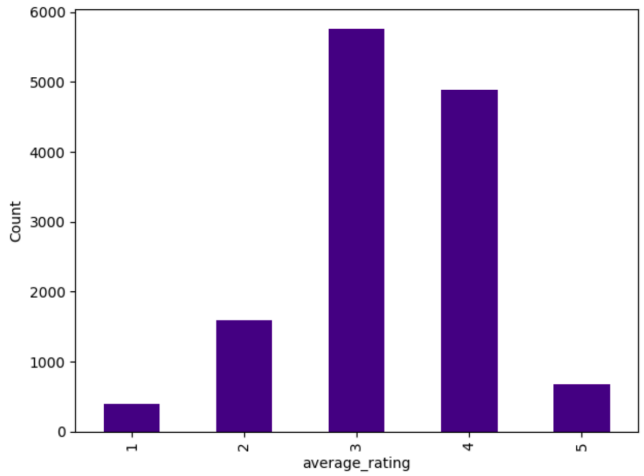
I used simple metapy based solution where I have tokenized each document with lowercase, used stop_word.txt with a bit of modification to improvise filtering and also applied porter.

Glimpse of data

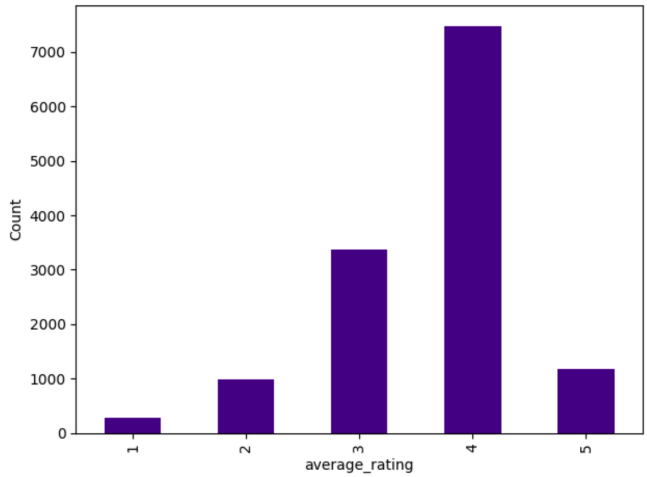
Shape of the data set: (13299, 8)

	zip_code	num_of_reviews	average_rating
count	13299.000000	13299.000000	13299.000000
mean	98113.023611	11.440935	3.290398
std	14.846820	18.422798	0.850737
min	98101.000000	1.000000	1.000000
25%	98104.000000	2.000000	3.000000
50%	98108.000000	6.000000	3.000000
75%	98119.000000	13.000000	4.000000
max	98199.000000	568.000000	5.000000

Without rounding average rating:



Rounding average rating to '0' decimal:



There is a significant change in representation by rounding off the average rating, I chose to go with rounding. After this I combined the data from different files for ease of working with it. Below is snapshot of the data frame.

	review_text	label	cuisines	zip_code	num_of_reviews	average_rating	token_text	clean_text
0	The baguettes and roll...	1	['Vietnamese', 'Sandwi...	98118	4	4	<metapy.metapy...	baguett roll excel haven't tri i'...
1	I live up the street fro...	1	['American (New)', 'Re...	98109	21	4	<metapy.metapy...	live street betti 160 sister town ...
2	I\'m worried about how...	1	['Mexican', 'Restaurants']	98103	14	3	<metapy.metapy...	i'm worri review place strong th...
3	Why can\'t you access...	0	['Mexican', 'Tex-Mex', ...	98112	42	4	<metapy.metapy...	can't access googl street view i...
4	Things to like about thi...	0	['Mexican', 'Restaurants']	98102	12	3	<metapy.metapy...	thing place homemad guacamo...
5	I had been holding off ...	1	['Breakfast & Brunch', '...	98107	59	4	<metapy.metapy...	hold visit bastill month time tho...
6	I had gone by this plac...	1	['Creperies', 'Restaura...	98105	6	4	<metapy.metapy...	gone place move prepar open ...
7	Any chance I get to ea...	0	['Vegetarian', 'Ethiopia...	98102	17	4	<metapy.metapy...	chanc eat hand social accept i'...
8	My favorite Thai restau...	0	['Thai', 'Restaurants']	98105	16	4	<metapy.metapy...	favorit thai restaur u district.loc...
9	I\'m pretty sure someo...	0	['Barbeque', 'Restaura...	98108	19	3	<metapy.metapy...	i'm pretti sure born rais washin...
10	This is a great bistro. ...	1	['French', 'Restaurants']	98112	3	4	<metapy.metapy...	great bistro wish citi nice decor...
11	Great little cafe! 4.25 ...	1	['Vietnamese', 'Chines...	98104	1	4	<metapy.metapy...	great cafe 4.25 star 160 love e...
12	The lady behind the co...	0	['Japanese', 'Restauran...	98122	2	2	<metapy.metapy...	ladi counter annoy make take o...
13	Sh_t. Stilly only in Saig...	0	['Delis', 'Vietnamese', '...	98106	2	4	<metapy.metapy...	sh_t stilli saigon 160 160 160 m...
14	Loved the Murder Mys...	0	['Breakfast & Brunch', '...	98101	1	4	<metapy.metapy...	love murder mysteri dinner crui...
15	This is an old school re...	1	['Breakfast & Brunch', '...	98105	7	3	<metapy.metapy...	old school retro fast food burg...
16	Wow - what a pit.....	0	['Delis', 'Sandwiches', '...	98134	1	2	<metapy.metapy...	pit drove morn say sign bahn mi...
17	The interior of the Fro...	0	['Barbeque', 'Southern'...	98121	29	4	<metapy.metapy...	interior frontier room kinda ugli ...
18	- Locale Destination - I...	1	['Thai', 'Restaurants']	98105	11	4	<metapy.metapy...	local destin it 45 th st latona av...
19	Mia's chicken salad sa...	0	['Asian Fusion', 'Veget...	98122	1	4	<metapy.metapy...	mia chicken salad sandwich 6.2...
20	I went to Ocho on a Th...	1	['Tapas Bars', 'Restaur...	98107	37	4	<metapy.metapy...	went ocho thursday 160 2 seat ...

Just to get a feel of the output and classification I used below TFIDF technique and just simple random forest classifier. I liked the precision/f1 score but not impressed.

```
vectorizer = TfidfVectorizer(min_df=15, max_df=0.5, stop_words="english", sublinear_tf=True, use_idf=True,
                             smooth_idf=True,
                             norm='l2', ngram_range=(1, 3))
pipeline = Pipeline([('vect', vectorizer),
                     ('chi', SelectKBest(chi2, k=500)),
                     ('clf', RandomForestClassifier())])
```

	precision	recall	f1-score	support
0	0.66	0.68	0.67	76
1	0.72	0.69	0.71	88
accuracy			0.69	164
macro avg	0.69	0.69	0.69	164
weighted avg	0.69	0.69	0.69	164

A little background

Reference - https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Tokenizing text with scikit-learn

Text preprocessing, tokenizing and filtering of stop words are all included in Count Vectorizer, which builds a dictionary of features and transforms documents to feature vectors

Count Vectorizer

Transforms text into a sparse matrix of n-gram counts. Convert a collection of raw documents to a matrix of TF-IDF features.

I used below parameters:

```
CountVectorizer(
stop_words=set(stopwords.words('english')),
strip_accents='unicode',
min_df=15,
max_df=0.5,
ngram_range=(1, 3))
```

From occurrences to frequencies

Occurrence count is a good start but there is an issue: longer documents will have higher average count values than shorter documents, even though they might talk about the same topics.

TfidfVectorizer

Convert a collection of raw documents to a matrix of TF-IDF features. Equivalent to Count Vectorizer` followed by `TfidfTransformer`. **TfidfTransformer** - Transform a count matrix to a normalized tf or tf-idf representation

I used below setting in my program:

```
TfidfVectorizer(
stop_words='english',
strip_accents='unicode',
min_df=3,
max_df=0.5,
ngram_range=(1, 3),
max_features=500)
```

It is very common to want to perform different data preparation techniques on different columns in your input data. Traditionally, this would require to separate the numerical and categorical data and then manually apply the transforms on those groups of features before combining the columns back together in order to fit and evaluate a model.

Reference: https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/

```
def classification_task(data_frame):
    print('Begin classification_task()...')
    train_set = data_frame[data_frame["label"] != "[None]"]
    test_set = data_frame[data_frame["label"] == "[None]"]

    text_type_list = ['review_text', 'clean_text']
    for text_type in text_type_list:
        print('**********************************************************')
        print('text type:' + text_type)

        train_review = train_set[text_type]
        train_label = train_set['label']
        X_train, X_test, y_train, y_test = train_test_split(train_review, train_label, test_size=0.3, random_state=123)

        vectorizer = TfidfVectorizer(min_df=1, stop_words="english", sublinear_tf=True, norm='l2', ngram_range=(1, 1))

        classifier_list = [MultinomialNB(), RandomForestClassifier(), LogisticRegression()]
        for classifier in classifier_list:
            print('classifier:' + str(classifier))
            pipeline = Pipeline([('vect', vectorizer),
                                 ('chi', SelectKBest(chi2, k=1200)),
                                 ('clf', classifier)])
            model = pipeline.fit(X_train, y_train)
            ytest = np.array(y_test)
            print(classification_report(ytest, model.predict(X_test)))
            print(confusion_matrix(ytest, model.predict(X_test)))
    print('End classification_task()...')
```

Original text **Vs** Cleaned text: (just review text, **unigram**)

```
classifier:MultinomialNB()
              precision    recall  f1-score   support

           0       0.69      0.70      0.69        76
           1       0.74      0.73      0.73        88

    accuracy                           0.71       164
   macro avg       0.71      0.71      0.71       164
weighted avg       0.71      0.71      0.71       164

[[53 23]
 [24 64]]
classifier:RandomForestClassifier()
              precision    recall  f1-score   support

           0       0.63      0.75      0.69        76
           1       0.74      0.62      0.68        88

    accuracy                           0.68       164
   macro avg       0.69      0.69      0.68       164
weighted avg       0.69      0.68      0.68       164

[[57 19]
 [33 55]]
```

```
classifier:MultinomialNB()
              precision    recall  f1-score   support

           0       0.70      0.64      0.67        76
           1       0.71      0.76      0.74        88

    accuracy                           0.71       164
   macro avg       0.71      0.70      0.70       164
weighted avg       0.71      0.71      0.71       164

[[49 27]
 [21 67]]
classifier:RandomForestClassifier()
              precision    recall  f1-score   support

           0       0.59      0.71      0.64        76
           1       0.69      0.57      0.62        88

    accuracy                           0.63       164
   macro avg       0.64      0.64      0.63       164
weighted avg       0.64      0.63      0.63       164

[[54 22]
 [38 50]]
```

```
classifier:LogisticRegression()
              precision    recall  f1-score   support

           0       0.65      0.82      0.72        76
           1       0.79      0.61      0.69        88

    accuracy                           0.71       164
   macro avg       0.72      0.71      0.71       164
weighted avg       0.73      0.71      0.71       164

[[62 14]
 [34 54]]
```

```
classifier:LogisticRegression()
              precision    recall  f1-score   support

           0       0.64      0.84      0.73        76
           1       0.81      0.59      0.68        88

    accuracy                           0.71       164
   macro avg       0.73      0.72      0.71       164
weighted avg       0.73      0.71      0.70       164

[[64 12]
 [36 52]]
```

Comparing the 3 classifiers I have used, **MultinomialNB** performed much better than Logistic regression and random forest. Plus, there is slight improvement when these classifiers were applied on cleaned text.

For multiple columns (features other than reviews) was getting below error using my code above.
ValueError: Found input variables with inconsistent numbers of samples

Then I treated the columns such as "cuisines", "zip code", "number of reviews" and "average rating" as string so that I can use them as categories, so get it into the dimension as I chose to represent review text. And as another **"text representation"**, I used trigram model below. Also have used the **ColumnTransformer** to perform this operation.

Since the dataset is skewed as mentioned in the task, I have used the "cross validation score" for score type as **"f1 macro"**

- Count Vectorizer with **trigram** model, I used below classifiers:
  - Review text **without** any processing:
    - Multinomial NB - 0.65
    - SVC - 0.58
    - Logistic Regression - 0.60
    - Random Forest - 0.61
  - Review text **with** processing:
    - Multinomial NB - 0.65
    - SVC - 0.57
    - Logistic Regression - 0.61
    - Random Forest - 0.63

- TF IDF with trigram model, I used below classifiers:
  - Review text **without** any processing:
    - Multinomial NB - 0.66
    - SVC - 0.65
    - Logistic Regression - 0.65
    - Random Forest - 0.64
  - Review text **with** processing:
    - Multinomial NB - 0.66

- SVC - 0.64
- Logistic Regression - 0.65
- Random Forest - 0.63

Summary

Most of the explanation is provided above, just to capture concise applied methods, I have summarized below.

- Text Representation: I have used multiple different ways of text representation for example using unigram and trigram model. Apart from that I have used variations of Counter Vectorizer and TF IDF vectorizers.

- Learning Algorithm: Some of the example of classifiers are SVM, Multinomial Naïve Bayes, Logistic Regression, Random Forest.
- What toolkit was used?
    - Pandas
    - Metapy
    - Sklearn
    - Genism
    - Nltk
    - Numpy
- How was text preprocessed?
    - For data cleaning, I primarily removed words occurring in stop words text file and at some places also used NLTK stop work library. In general, used metapy tokenization methods.
- How was text represented as features?
    - Along with variations of the text representations, I have used support of multiple other features that were provided such as "cuisines", "average rating", "number of reviews", etc.
- What was the learning algorithm used?
    - Multinomial NB
    - SVC
    - Logistic Regression

- Random Forest

That explains why in my local execution I did not achieve over 66% but when I used the MultinomialNB with TFIDF on cleaned data and predicted the rest of labels as per the ask from the task, I got (atifa2).

- Overall score: 0.7863
- Precision: 0.7923
- Recall: 0.7774

Using more features along with MultinomialNB has certainly improved my overall result.