Fundamentals of Data Analytics and Learning

HW1 – First Visit in Kaggle Data

3/9/2022

Name: Hoe Zi Onn / 何子安

Student ID: E44065020

Year: PSY 110 / 心理 110

Table of Contents

Data Description

Data Source:
- https://www.kaggle.com/sbhatti/financial-sentiment-analysis

Dataset:
- Financial Sentiment Analysis (Financial sentences with sentiment labels)

Introduction of Dataset:
- The following data is intended for advancing financial sentiment analysis research. It's two datasets (FiQA, Financial PhraseBank) combined into one easy-to-use CSV file. It provides financial sentences with sentiment labels.

Citations:
- Malo, Pekka, et al. "Good debt or bad debt: Detecting semantic orientations in economic texts." Journal of the Association for Information Science and Technology 65.4 (2014): 782-796.

Target:
- Classification

This dataset contains only single file and saved in *.csv* fomat. We will review the following sections through *python notebook cell* with theirs outputs. And all the libraries/packages code will be skipped.

(Completed code are available at https://github.com/onnnnn/Fundamentals-of-Data-Analytics-and-Learning/blob/main/hw1/HW1_E44065020.ipynb.)

Take a Look at the Dataset

This dataset has only two columns which is `Sentence` and `Sentiment`.

`Sentence`: Sentences that we are going to classify and analysis. (features)
`Sentiment`: Answers that we are going to predict. (labels)

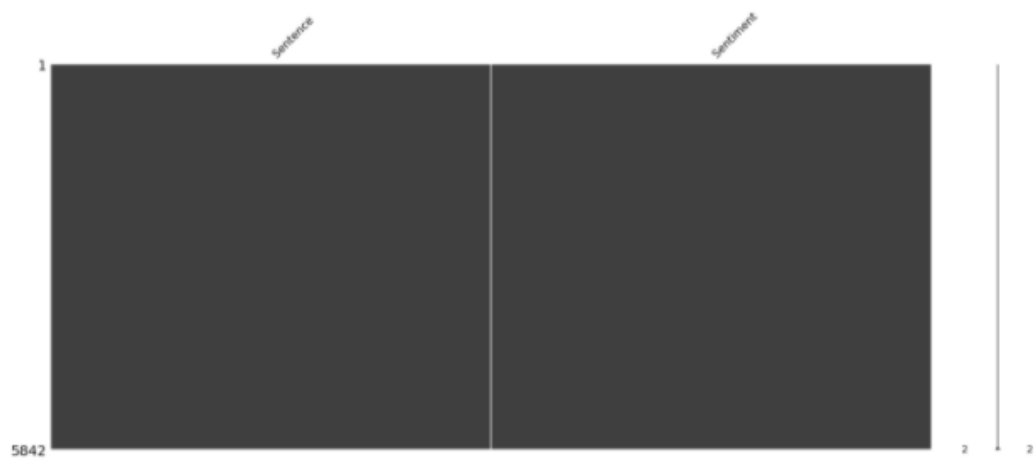
```
df = pd.read_csv('data.csv')
df.head()
```

	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	*ESIonlows, down*1.50 to $2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral

```
print(df.isnull().sum())
msn.matrix(df)
```

```
Sentence     0
Sentiment    0
dtype: int64

<AxesSubplot:>
```



- There is **no missing value** in the dataset.

EDA

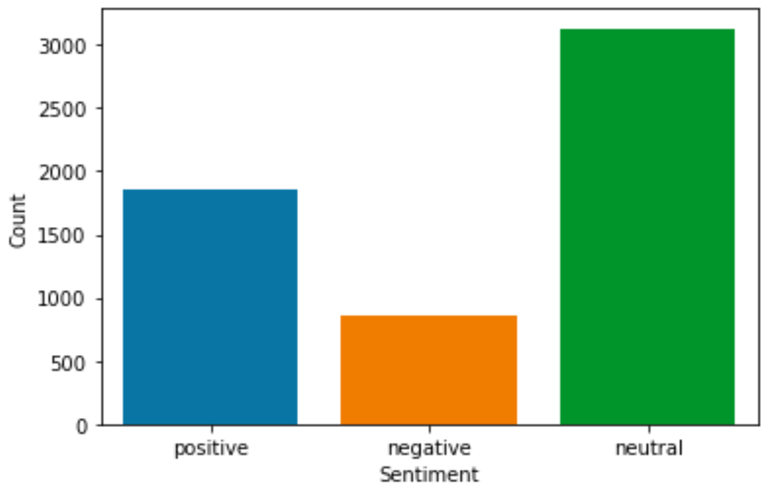
Labels Distribution

Counting the numbers of labels.

```
df['Sentiment'].value_counts()
```

```
neutral     3130
positive    1852
negative     860
Name: Sentiment, dtype: int64
```

```
sns.countplot(x='Sentiment', data=df)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```




```
neutral, pos, neg = df['Sentiment'].value_counts()
total = neutral + pos + neg
for i, j in zip(['neutral', 'positive', 'negative'], df['Sentiment'].value_co
unts()):
    print(f'{i}: {round(j/total*100, 2)}%')
```

```
neutral: 53.58%
positive: 31.7%
negative: 14.72%
```

- These values indicate the dataset is **imbalanced**.

Apply label encoding into `Sentiment` column

The labels of `Sentiment` column need to be translate into numbers.

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated.

```
label_encoder = preprocessing.LabelEncoder()
df['Sentiment'] = label_encoder.fit_transform(df['Sentiment'])
df['Sentiment'].unique()

df.head()
```

	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	2
1	$ESIonlows, down$1.50 to $2.50 BK a real po...	0
2	For the last quarter of 2010 , Componenta 's n...	2
3	According to the Finnish-Russian Chamber of Co...	1
4	The Swedish buyout firm has sold its remaining...	1

Wordcloud of Sentence's Text

By putting all words together with *stopwords* being removed, we shall see the word that showed up more often will be bigger than those who barely showed.

```
stopwords_ = set(STOPWORDS)

text = " ".join(i for i in df.Sentence)
wordcloud = WordCloud(stopwords=stopwords_, background_color="white").generat
e(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Cleaning the Sentences

Real-world datasets often contains a lot of incorrect/incomplete informations. I onces heard that your insight and analysis are only as good as the data you are using. Also as garbage data in, garbage analysis out. So in this step, we are going to 'clean' our data like removing stopwords, standardize the data and so on.

> **Data cleaning** is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct.

```
nltk.download('stopwords')

corpus = []
for i in range(0, len(df)):
    review = re.sub('[^a-zA-Z]', ' ', df['Sentence'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    review = [ps.stem(word)
              for word in review
              if not word in set(all_stopwords)]
    review = ' '.join(review)
    corpus.append(review)
```

```
corpus[0: 3]
```

```
['geosolut technolog leverag benefon gp solut provid locat base search technolog commun platf
orm locat relev multimedia content new power commerci model',
 'esi low bk real possibl',
 'last quarter componenta net sale doubl eur eur period year earlier move zero pre tax profit
pre tax loss eur']
```

X-y Preparation

For most ML algorithms are required x (features) , y (labels) as the inputs and we need to vectorize the features before we give it to our models, called **features extraction**.

```
cv = CountVectorizer(max_features=1500, ngram_range=(1,3))
X = cv.fit_transform(corpus).toarray()
y = df.iloc[:, -1].values
```

Model Training and Model Evaluation

After splitting our data into *training data* and *test data*, we use *training data* to train our models and *test data* to evaluate our model's performance.

I ran only `Naive Bayes` in this part, and the result will be the **baseline** to our improved method.

metrics: `confusion_matrix`, `accuracy_score`

> **Confusion matrix**: a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.

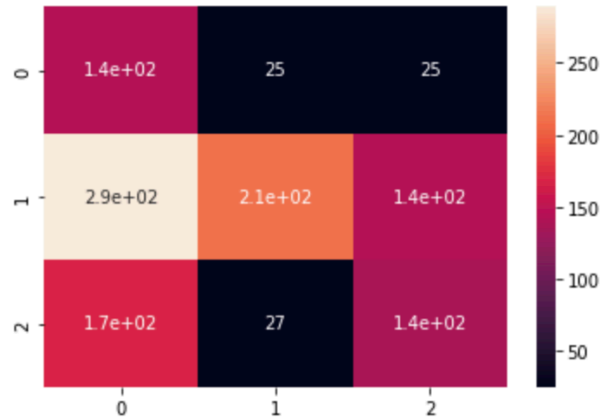
Naive Bayes

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
sns.heatmap(cm, annot=True)
plt.show()
```

```
[[142  25  25]
 [289 211 143]
 [169  27 138]]
```



X-y Preparation (using TF-IDF)

TF-IDF (term frequency-inverse document frequency) can quantify the importance or relevance of string representations (words, phrases, lemmas, etc) in a document amongst a collection of documents (also known as a corpus).

```
tfidf_v = TfidfVectorizer(max_features=5000, ngram_range=(1,3))
X = tfidf_v.fit_transform(corpus).toarray()
y = df['Sentiment']

X1_train, X1_test, y1_train, y1_test = train_test_split(
    X, y, test_size = 0.25,random_state = 0)

print(X1_train.shape)
print(X1_test.shape)
print(y1_train.shape)
print(y1_test.shape)
```

```
(4381, 5000)
(1461, 5000)
(4381,)
(1461,)
```

```
count_df = pd.DataFrame(X1_train, columns=tfidf_v.get_feature_names())
count_df
```

	aapl	aapl http	aapl http co	ab	ab inbev	ab sto	abb	abl	abp	abp hel	...	yhoo	yhtyma	yhtyma oyj	yhtyma oyj hel	yit	yit corpor	york	zinc	znga	zone
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4376	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4377	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4378	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4380	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

4381 rows × 5000 columns

Model Training and Model Evaluation (using TF-IDF)

I ran multiple models (`Naive Bayes`, `MultinominalNB`, `CatBoost`, `XGBoost`) in this part as to be compare to other models.

metrics: `confusion_matrix`, `accuracy_score`

Naive Bayes

```
classifier = GaussianNB()
classifier.fit(X1_train, y1_train)
y1_pred = classifier.predict(X1_test)

acc2 = accuracy_score(y1_test, y1_pred)
print(f"Accuracy of Naive Bayes (Using TF - IDF technique): {acc2}")
```

```
Accuracy of Naive Bayes (Using TF - IDF technique): 0.5400410677618069
```

MultinominalNB

```
classifier = MultinomialNB()
classifier.fit(X1_train, y1_train)
pred = classifier.predict(X1_test)
score = accuracy_score(y1_test, pred)
score
```

```
0.6721423682409309
```

CatBoost

CatBoost is an algorithm for gradient boosting on decision trees.

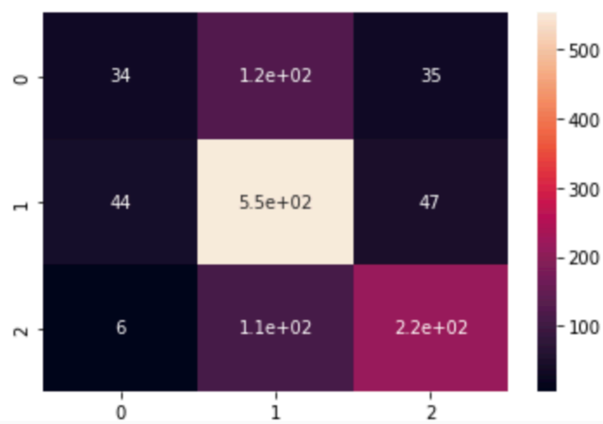
```
classifier = CatBoostClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

acc4 = accuracy_score(y_test, y_pred)
print(acc4)

sns.heatmap(cm,annot=True)
plt.show()
```

```
[[ 34 123  35]
 [ 44 552  47]
 [  6 109 219]]
0.688622754491018
```



XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

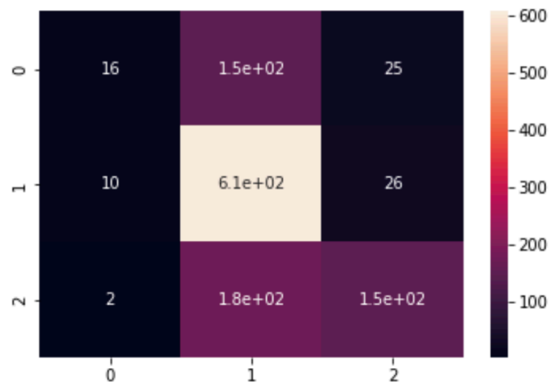
```
classifier = XGBClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)

acc3 = accuracy_score(y_test, y_pred)
print(acc3)

sns.heatmap(cm, annot=True)
plt.show()
```

```
[[ 16 151  25]
 [ 10 607  26]
 [  2 178 154]]
0.6646706586826348
```



Comparision Between Difference Notebook's Code

1. Different *packages* and *ML models* are being used.

2. Some even apply different *K-fold* method.

3. Different *evaluation metrics* are being used.

4. Different *text processing* would lead to different results on varies ML models.

Thoughts on This Dataset Among Others

- This dataset is not like the other classical's dataset. It's a **NLP (Natural Language Processing)** task, aims to do **classification**.

- There are lot of way to do text processing in NLP task, and I only scratch the surface.

References

1. Data Cleaning - https://www.tableau.com/learn/articles/what-is-data-cleaning
2. Label Encoding - https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/#:~:text=Label%20Encoding%20refers%20to%20converting,structured%20dataset%20in%20supervised%20learning.
3. TF-IDF - https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/
4. Confusion Matrix - https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/
5. CatBoost - https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/
6. XGBoost - https://xgboost.readthedocs.io/en/stable/
7. Kaggle - Dataset - https://www.kaggle.com/sbhatti/financial-sentiment-analysis
8. Kaggle - Code - https://www.kaggle.com/sanjoymondal0/financial-sentiment-analysis