Project #1 Report

110550080 何曉嫻

I. Description

After referencing the 20 Newsgroups dataset, I find news classification to be quite fascinating. I can also easily collect data from major news websites using web crawlers. So, I've decided to use "News Headline Classification" as my topic.

In this project, I will collect news headlines along with their categories, preprocessing data, and classify them using machine learning algorithms.

II. Dataset

I collected news headlines from 2020 to 2024, along with their categories on news websites, including "影劇", "體育", "政治", "財經", "健康", "時尚", "生活", "地方", "社會", "國際" I randomly selected 500 pieces of data for each category to create my dataset.

I write python code to crawl news title from the ETToday website. It generates random dates within a specified range and constructs URLs for scraping. After sending requests and parsing HTML content, it extracts article titles, dates, categories, and links. The data is then filtered and stored in a DataFrame, which is later concatenated to create a comprehensive dataset. Finally, the aggregated data is saved to a CSV file.

Dataset format: A csv file with columns “title”, “category”, “date”. Total 5000 pieces of data.

III. Methods

(a.) Feature Extraction

First, I try (1.) CKIP Tagger and as tokenizer and KyBERT for keyword extraction (2.) Jieba for word segmentation and found that (2) is better. Thus, I use Jieba in my experiment. Then, I use CountVectorizer in scikit-learn library to convert text data into a numerical representation, which can be processed by machine learning algorithm.

(b.) Supervised and unsupervised algorithms

In this dataset, I use five kinds of supervised learning algorithms, Logistic Regression, Random Forest Classifier, Multinomial Naive Bayes, Support Vector Machine, and K-Nearest Neighbors. One unsupervised learning algorithm K-Means.

Linear regression

Linear Regression (LR) is a foundational supervised learning algorithm widely used for multiclass prdiction tasks. It models the relationship between one or more independent variables (features) and a continuous dependent variable (target). In the context of multiclass prediction, LR can be adapted through techniques like One-vs-All or softmax regression to handle multiple classes. LR calculates a linear equation that best fits the training data, allowing it to make predictions for new data points.

Random Forest

Random Forest makes predictions by aggregating the decisions of multiple decision trees. During training, each tree is trained on a bootstrap sample of the data and considers only a random subset of features at each node. In classification tasks, each tree "votes" for the class label. This ensemble method reduces overfitting and enhances generalization. Final predictions are based on the combined decisions of all trees, ensuring robustness and accuracy.

Multinomial Naive Bayes

Multinomial Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with the "naive" assumption of feature independence. It's particularly suited for text classification tasks with discrete features (e.g., word counts). The algorithm calculates the probability of each class given a document's feature vector and selects the class with the highest probability as the prediction.

Support Vector Machines

Support Vector Machines (SVM) is a supervised machine learning algorithm used for classification and regression analysis. It works by finding the best possible line or hyperplane to separate different classes of data points. This line is chosen to have the maximum possible distance from the nearest data points of each class, called support vectors. During prediction, SVM assigns new data points to classes based on which side

of this line they fall. SVM is effective for both linear and non-linear decision boundaries and is robust to outliers.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a supervised learning algorithm used for classification and regression. In text classification, KNN assigns a label to a document based on the majority class of its K nearest neighbors in the feature space (e.g., TF-IDF vectors). For document similarity, KNN measures the similarity between documents based on their feature representations and retrieves the K most similar documents. KNN is non-parametric and instance-based, meaning it doesn't make any assumptions about the underlying data distribution.

K-Means clustering

K-Means clustering is a widely used unsupervised algorithm for clustering data into K groups. It initializes centroids, assigns data points to the nearest centroid, updates centroids, and repeats until convergence. Determining the optimal number of clusters is crucial. While computationally efficient, K-Means has limitations like sensitivity to initial centroids. Despite this, it's valuable for tasks like customer segmentation and anomaly detection.

IV. Experiments

I experimented with10-class at first, but the results weren't as satisfactory as I hoped. Therefore, I removed the categories that I personally found more challenging to differentiate. I ultimately decided to use the following 6 categories for further work: "影劇", "體育", "政治", "財經", "健康", "時尚".

I design 8 kinds of experiment on dataset, for each dataset, I will first remove punctuation, perform word segmentation, vectorize the text, and then use algorithms for classification and test 5-fold cross-validation. I will analyze the results using metrics such as accuracy, F1 score, precision, recall, and confusion matrix for supervised learning. As well as inertia and silhouette for unsupervised learning.

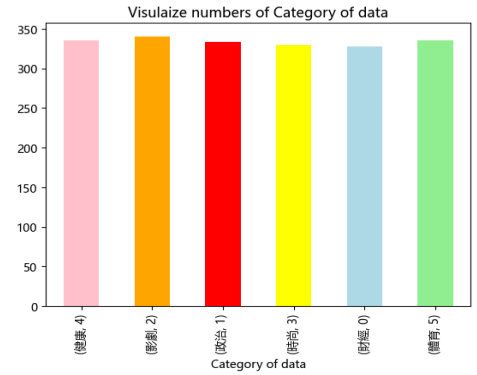
The experiment dataset

500s (baseline): Selecting 500 data points from each category.

Other_6: I conducted experiments with 9-class, 8-class, and 7-class classifications. In each iteration, I gradually eliminated the class with the lowest precision from the predictions. The result classes "影劇", "政治", "健康", "時尚", "生活", "地方" are different from the classes I select. I use these data to do 500s.

300s: Selecting 300 data points from each category.

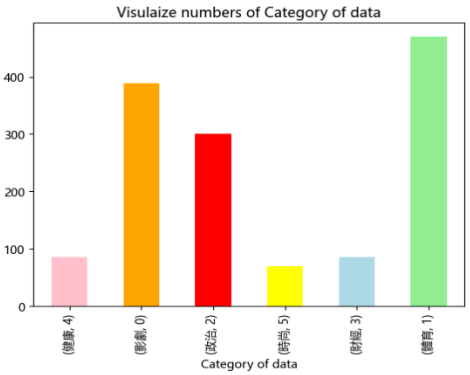
Rand: Randomly select 2000 records. The distribution is as follows:



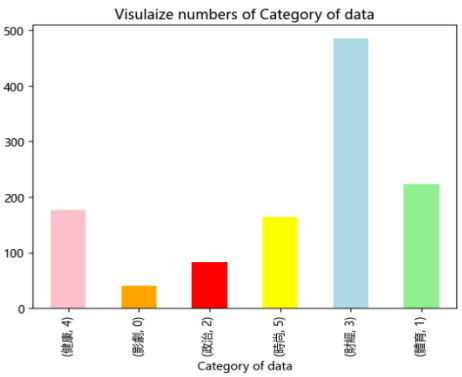
.

Raw: 500s without Jieba word segmentation.

Imb1: Imbalanced data, the distribution is as follows:



Imb2: Imbalanced data, the distribution is as follows.



Aug: The whole dataset only has 3000 records. Randomly duplicate some data to reach a total of 4000 records.

V. Analysis

Selecting the best result from the 5-fold cross-validation. And conduct comparison of performance across various algorithms for different datasets.

Linear regression (LR) Result

In Fig. 1, I can see that using logistic regression yields similar metric values across all datasets.

Comparing Rand and Aug, it's evident that increasing the size of the training and validation sets, along with some content repetition, leads to better performance. Conversely, comparing 500s and 300s, it's clear that adding data that hasn't appeared before doesn't result in a significant improvement in performance.

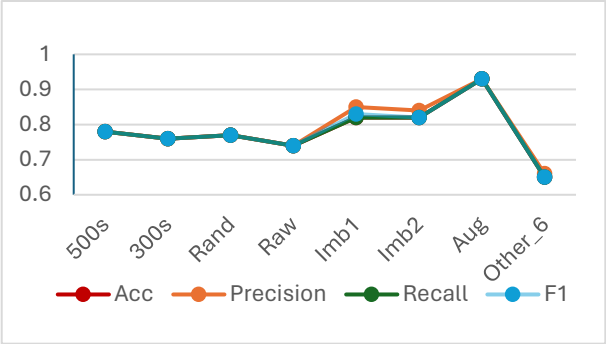


Fig. 1. The result of Linear regression

Random Forest Result

The trend in the charts is generally similar to LR, although the values are slightly lower. The results for Other_6 are notably lower than for 500s, indicating that the selection of "categories" is crucial. The categories within Other_6 may have minimal differences in content, making classification more challenging.

In Imb1 and Imb2, precision is higher than recall, the model may prioritize reducing false positives, meaning it tends to correctly predict positives but might miss some true positives. However, upon observing the confusion matrix (Fig. A1), I found that most instances were misclassified into the "健康" category, which is not the largest in quantity.

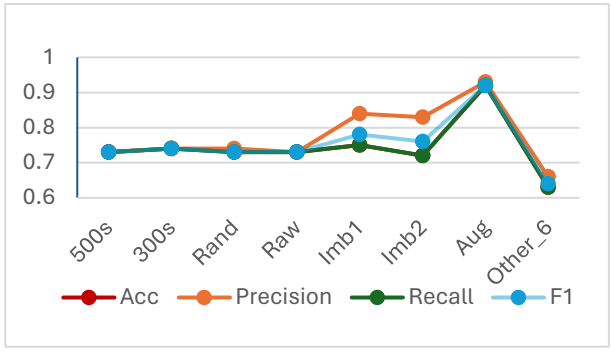


Fig. 2. The result of Random Forest

Multinomial Naive Bayes Result

Multinomial Naive Bayes has the best overall performance, with similar accuracy, precision, recall, and F1 scores. However, in terms of Aug, there is no significant increase.

There is nothing new based on the datasets that can be compared.

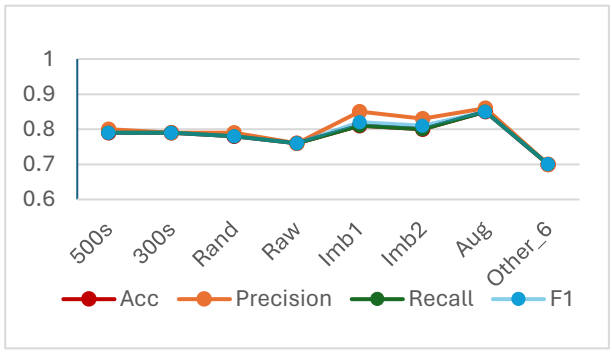


Fig. 3. The result of Multinomial Naive Bayes

Support Vector Machines (SVM) Result

The higher accuracy of SVM on the RAW dataset compared to the 500s dataset may be attributed to factors related to model performance.

In Imb2, there are significant differences in precision, recall, and F1 scores. Observing the confusion matrix (Fig. A2), I noticed severe overfitting, with almost every category being mainly classified into the class with the highest number of instances.

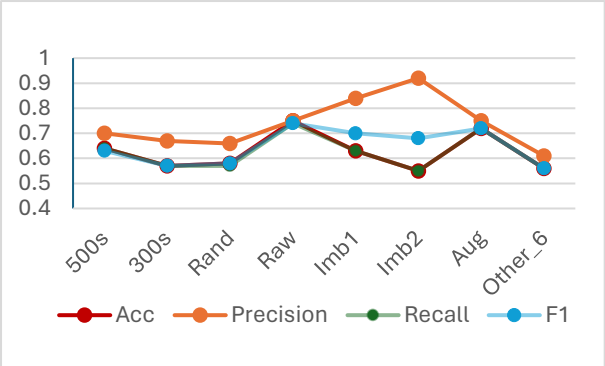


Fig.4. The result of SVM

K-Nearest Neighbors (KNN) Result

In all models, the accuracy of Imb1 is better than Imb2, the differences are particularly pronounced for SVM and KNN. I think this is because Imb1 has three categories that are significantly more abundant, while Imb2 has only one category with a higher quantity, making it more imbalanced.

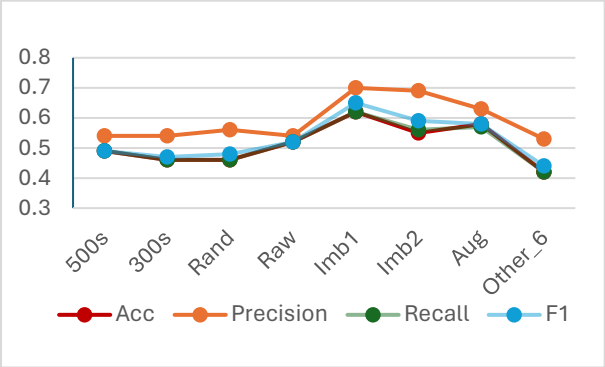


Fig.5. The result of KNN

K-Means clustering

The results produced by K-means are poor and they are different from the trend in supervised learning method. I think the Inertia just based on the amount of data.

However, the inertia of the Raw dataset is smaller than that of the 500s dataset. I think this is because, without tokenization, sentences are treated as long strings, which may result in a higher-dimensional feature space and more refined distance calculations between samples, leading to smaller inertia.

The silhouette score of the Raw dataset is the lowest, but the silhouette scores of all datasets are all close to 0, indicating that the clustering performance is not very ideal and the Silhouette Score here is not particularly meaningful for comparison. Maybe K-means is not suitable for text classification.

	Inertia	Silhouette Score
500s	69577	0.05941
300s	41777	0.06004
Rand	46509	0.06083
Raw	52104	0.02613
Imb1	31938	0.06080
Imb2	29862	0.05764
Aug	92912	0.05808
Other_6	70256	0.05946

Table 1. The result of K-Means

VI. Discussion

Based on your experiments, are the results and observed behaviors what you expect?

1. I initially thought that using a stronger classifier like SVM would yield the best results. However, it turned out to be one of the lowest performers.
2. Before designing the experiment, I believed that having sufficient data for each category would help the model better learn the structure and features of the data. Therefore, I mostly tried datasets with an equal number of instances for each class. However, after experimenting with various imbalanced datasets, I found that the metrics obtained were consistently higher than those from balanced datasets.
3. Other_6 perform worse than 500s. Because Other_6 is selected based on experiment, 500s is selected in my opinion.

Discuss factors that affect the performance, including dataset characteristics.

As mentioned in V. Analysis,

1. Selected categories.
2. Numbers of data.
3. Data balance.
4. Different data preprocessing method.
5. Different Algorithm.

Describe experiments that you would do if there were more time available.

1. Crawling news content and classifying based on the article rather than the title.
2. Trying other word segmentation and keyword extraction (remove stop words) methods.
3. Use NLP model like Bert, or deep learning-based algorithm.
4. While creating Other_6, I only relied on the precision of each category to decide which one to remove, without observing the confusion matrix to identify which categories are more prone to misclassification. I want to consider more factors to identify the six categories that can achieve the highest accuracy.

Indicate what you have learned from the experiments and remaining questions.

This is my first time going through the process of coming up with topics, preparing datasets, processing raw data, finding models, adjusting models, and analyzing various datasets. It's also my first time to apply text on machine learning techniques. Unfortunately, the accuracy of most classifications is not high.

In experiment, I learned how to use functions from scikit-learn to perform tasks such as cross-validation and utilizing algorithms. The discussion and let me learn to think about the impact of different input data on prediction results and understand the reasons behind the changes in metrics.

Appendix

Linear regression

lr	Acc	Precision	Recall	F1
500s	0.78	0.78	0.78	0.78
300s	0.76	0.76	0.76	0.76
Random	0.77	0.77	0.77	0.77
Raw	0.74	0.74	0.74	0.74
Imb1	0.82	0.85	0.82	0.83
Imb2	0.82	0.84	0.82	0.82
Aug	0.93	0.93	0.93	0.93
Other_6	0.65	0.66	0.65	0.65

Random forest

rf	Acc	Precision	Recall	F1
500s	0.73	0.73	0.73	0.73
300s	0.74	0.74	0.74	0.74
Rand	0.73	0.74	0.73	0.73
Raw	0.73	0.73	0.73	0.73
Imb1	0.75	0.84	0.75	0.78
Imb2	0.72	0.83	0.72	0.76
Aug	0.92	0.93	0.92	0.92
Other_6	0.63	0.66	0.63	0.64

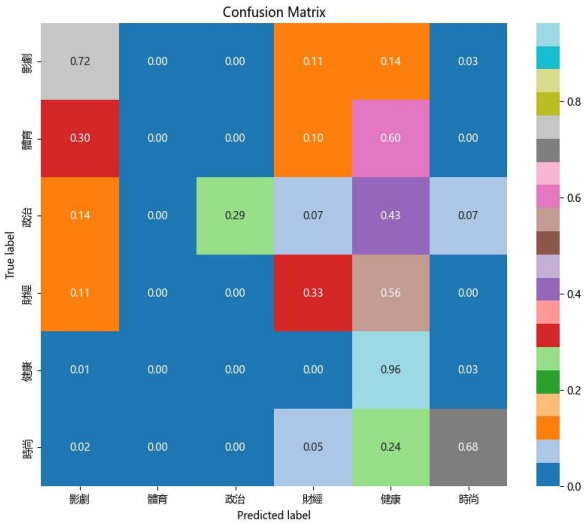


Fig.A1. Confusion matrix of imb2 using lr

Multinomial Naive Bayes

mnb	Acc	Precision	Recall	F1
500s	0.79	0.80	0.79	0.79
300s	0.79	0.79	0.79	0.79
Rand	0.78	0.79	0.78	0.78
Raw	0.76	0.76	0.76	0.76
Imb1	0.81	0.85	0.81	0.82
Imb2	0.80	0.83	0.80	0.81
Aug	0.85	0.86.	0.85	0.85
Other_6	0.70	0.70	0.70	0.70

Support Vector Machines

svm	Acc	Precision	Recall	F1
500s	0.64	0.70	0.64	0.63
300s	0.57	0.67	0.57	0.57
Rand	0.58	0.66	0.57	0.58
Raw	0.75	0.75	0.74	0.74
Imb1	0.63	0.84	0.63	0.70
Imb2	0.55	0.92	0.55	0.68
Aug	0.72	0.75	0.72	0.72
Other_6	0.56	0.61	0.56	0.56

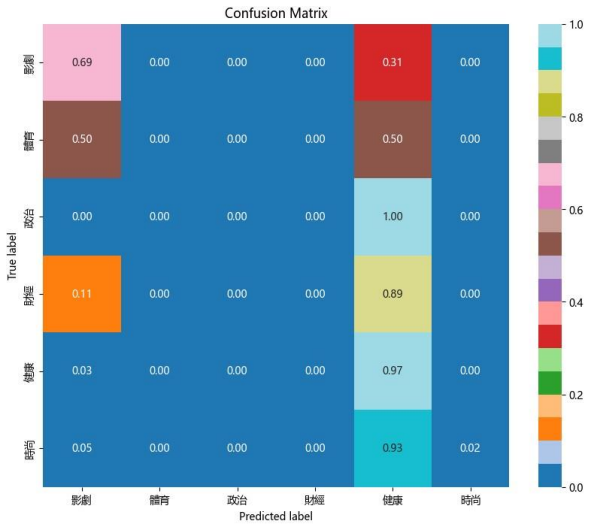


Fig.A2. Confusion matrix of imb2 using svm

K-Nearest Neighbors

knn	Acc	Precision	Recall	F1
500s	0.49	0.54	0.49	0.49
300s	0.46	0.54	0.46	0.47
Rand	0.46	0.56	0.46	0.48
Raw	0.52	0.54	0.52	0.52
Imb1	0.62	0.70	0.62	0.65
Imb2	0.55	0.69	0.56	0.59
Aug	0.58	0.63	0.57	0.58
Other_6	0.42	0.53	0.42	0.44

Code

The part of code I modify to process Chinese. The version of no tokenizer, I set max features to 500, different from with tokenizer one. Because it has less words.

```
from sklearn.feature_extraction.text import
CountVectorizer
import jieba

def chinese_tokenizer(text):
    words = jieba.cut(text)
    # print(" ".join(words))
    return " ".join(words)

def dummy_tokenizer(text):
    print(text)
    return text

# tokenizer
cv =
CountVectorizer(tokenizer=chinese_tokenizer,token_patte
rn=None, max_features=1000)

# no tokenizer
# cv = CountVectorizer(tokenizer=dummy_tokenizer,
max_features=500)

x = cv.fit_transform(dataset.title).toarray()
x = cv.fit_transform(dataset.title).toarray()

print("X.shape = ", x.shape)
print("y.shape = ", y.shape)
```

The part of code I modify to use the algorithm and metrics.

```
def plot_confusion_matrix(cm, target_names, title='Confusion
matrix', cmap=None, normalize=True, name=None):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]
    plt.figure(num= name, figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt=".2f", cmap=cmap,
xticklabels=target_names, yticklabels=target_names)
    plt.title(title)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
```

```
def run_model(model_name):
    mdl=''
    if model_name == 'Logistic Regression':
        mdl = LogisticRegression()
    elif model_name == 'Random Forest':
        mdl = RandomForestClassifier(n_estimators=100
,criterion='entropy' , random_state=0)
    elif model_name == 'Multinomial Naive Bayes':
        mdl = MultinomialNB(alpha=1.0,fit_prior=True)
    elif model_name == 'Support Vector Classifer':
        mdl = SVC()
    elif model_name == 'K Nearest Neighbour':
        mdl = KNeighborsClassifier(n_neighbors=10 , metric=
'minkowski' , p = 4)
    elif model_name == 'K-Means Clustering':
        mdl = KMeans(n_clusters=6, random_state=0)


    kfold = KFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(mdl, x, y, cv=kfold)
    best_acc = 0

    if mdl != '':
        if model_name == 'K-Means Clustering':
            mdl.fit(x)
            inertia = mdl.inertia_
            silhouette = silhouette_score(x, mdl.labels_)
            print(f'Inertia: {inertia}')
            print(f'Silhouette Score: {silhouette}')

        else:
            for i, (train_index, test_index) in
enumerate(kfold.split(x, y)):
                X_train_fold, X_test_fold = x[train_index],
x[test_index]
                y_train_fold, y_test_fold = y[train_index],
y[test_index]

                mdl.fit(X_train_fold, y_train_fold)

                y_pred = mdl.predict(X_test_fold)

                accuracy = accuracy_score(y_test_fold,
y_pred)
                precision = precision_score(y_test_fold,
y_pred, average='micro')
                recall = recall_score(y_test_fold, y_pred,
average='micro')
                f1 = f1_score(y_test_fold, y_pred,
average='micro')

                if accuracy > best_acc:
                    best_acc = accuracy
                    x_train = X_train_fold
                    x_test = X_test_fold
                    y_train = y_train_fold
                    y_test = y_test_fold


                perform_list.append({
                    'Model': model_name,
                    'Fold': i + 1,
                    'Accuracy': round(accuracy, 2),
                    'Precision': round(precision, 2),
                    'Recall': round(recall, 2),
                    'F1': round(f1, 2)
                })

            mdl.fit(x_train, y_train)
            y_pred = mdl.predict(x_test)
            score = accuracy_score(y_test, y_pred)
            print("accuracy of "+model_name+" is:",score)
            print(classification_report(y_pred, y_test,
target_names=target_category))
            cm = confusion_matrix(y_test, y_pred)

            plot_confusion_matrix(cm,
target_names=target_category, title='Confusion Matrix',
cmap='tab20', name = model_name)
```

Complete code & dataset:

https://github.com/lydiahoho/AI-Capstone/blob/main/hw1

Reference

Web crawl:

https://mdium.com/@ethan.chen927/%E7%B6%B2%E8%B7%AF%E7%88%AC%E8%9F%B2-ett-day%E6%96%B0%E8%81%9E%E7%AC%AC%E4%B8%80%E9%83%A8%E5%88%86-8d1002783c27

Word tokenizer:

https://blog.droidtown.co/post/188714326387/articutnlp04

KeyBert:

https://tako-analytics.com/2023-06-19-how-to-extract-key-words-from-traditional-chinese-articles-in-nlp/

model:

https://github.com/deepak0437/natural_language_processing/tree/main/News_Article