# RK-2 MMO

王培YU5I-22M

选项：随机森林分类器；补朴素贝叶斯 (CNB)

```
将  numpy  导入为  np
将  熊猫  导入为  pd
pd.set_option ( 'display.max_colwidth',  100 )
导入  操作系统
for  dirname,  _,  filenames  in  os.walk('/kaggle/input'):
        for  filename  in  filenames:
                print(os.path.join(dirname,  filename))
```

```
from  google.colab  import  drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
```

```
%cd  /content/drive/MyDrive
```

```
/content/drive/MyDrive
```

```
df  =  pd.read_csv("spam.csv",encoding='latin-1')
```

```
df_nan_count  =  pd.DataFrame(df.isnull().sum())
df_nan_count  =  df_nan_count.reset_index()
df_nan_count.columns  =  ["colname","count  of  null  value"]
display(df_nan_count)
```

| | colname | count of null value |
|---|---|---|
| **0** | v1 | 0 |
| **1** | v2 | 0 |
| **2** | Unnamed: 2 | 5522 |
| **3** | Unnamed: 3 | 5560 |
| **4** | Unnamed: 4 | 5566 |

```
df=  df[['v1',  'v2']]
df_1=  df_1[['v1',  'v2']]
```

```
df.head()
```

| | v1 | v2 |
|---|---|---|
| **0** | ham | Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g... |
| **1** | ham | Ok lar... Joking wif u oni... |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ... |
| **3** | ham | U dun say so early hor... U c already then say... |
| **4** | ham | Nah I don't think he goes to usf, he lives around here though |

```
pd.DataFrame(df["v1"].value_counts()  /  df["v1"].count().sum()  *  100  )
```

| | v1 |
|---|---|
| **ham** | 86.593683 |
| **spam** | 13.406317 |

```
print("Input  data  has  {}  rows  and  {}  columns".format(len(df),  len(df.columns)))
```

```
Input data has 5572 rows and 2 columns
```

```
df.columns  =  ['label',  'body_text']
```

```
print("Out  of  {}  rows,  {}  are  spam,  {}  are  ham".format(len(df),
```

```
Out of 5572 rows, 747 are spam, 4825 are ham
```

```
print("Number  of  null  in  label:  {}".format(df['label'].isnull().sum()))
print("Number  of  null  in  text:  {}".format(df['body_text'].isnull().sum()))
```

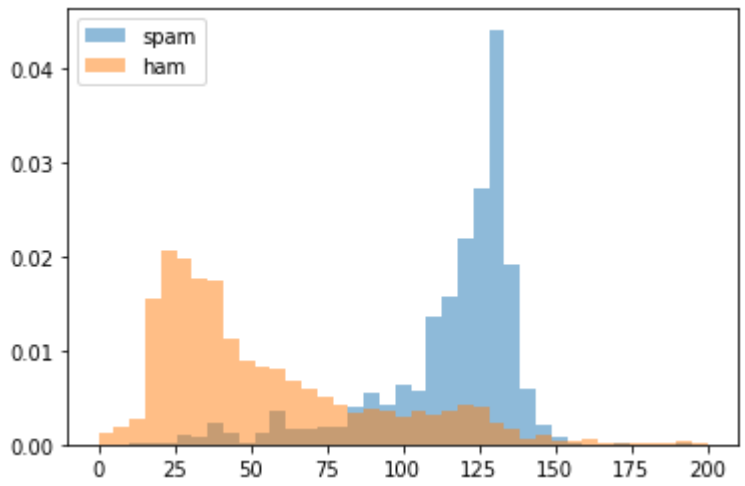
```
Number of null in label: 0
Number of null in text: 0
```

```
import  nltk
nltk.download('stopwords')
import  pandas  as  pd
import  re
from  sklearn.feature_extraction.text  import  TfidfVectorizer,CountVectorizer
import  string
```

```
stopwords = nltk.corpus.stopwords.words('english')
ps = nltk.PorterStemmer()



def count_punct(text):
    count = sum([1 for char in text if char in string.punctuation])
    return round(count/(len(text) - text.count(" ")), 3)*100


def clean_text(text):
    text = "".join([word.lower() for word in text if word not in string.pu
    tokens = re.split('\W+', text)
    text = [ps.stem(word) for word in tokens if word not in stopwords]
    return text
```

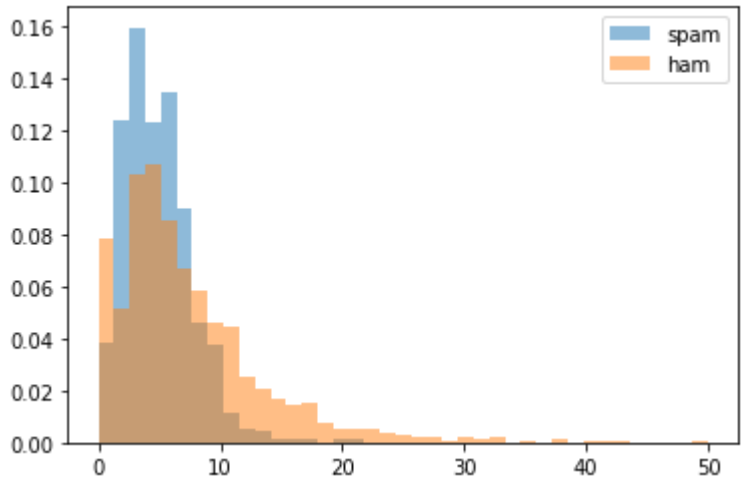
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
df['body_clean_text'] = df['body_text'].apply(lambda x: clean_text(x))
df['body_len'] = df['body_text'].apply(lambda x: len(x) - x.count(" "))
df['punct%'] = df['body_text'].apply(lambda x: count_punct(x))


from matplotlib import pyplot
import numpy as np
%matplotlib inline
bins = np.linspace(0, 200, 40)

pyplot.hist(df[df['label']=='spam']['body_len'], bins, alpha=0.5, density = True, l
pyplot.hist(df[df['label']=='ham']['body_len'], bins, alpha=0.5, density = True, la
pyplot.legend(loc='upper left')
pyplot.show()
```




```
bins = np.linspace(0, 50, 40)

pyplot.hist(df[df['label']=='spam']['punct%'], bins, alpha=0.5, density = True, lab
```

```
pyplot.hist(df[df['label']=='spam']['punct%'],  bins,  alpha=0.5,  density  =  True,  labe
pyplot.hist(df[df['label']=='ham']['punct%'],  bins,  alpha=0.5,  density  =  True,  labe
pyplot.legend(loc='upper  right')
pyplot.show()
```




## Создать TF-IDf и CountVectorizer

```
#  TF-IDF
tfidf_vect  =  TfidfVectorizer(analyzer=clean_text)
X_tfidf  =  tfidf_vect.fit_transform(df['body_text'])
X_tfidf_feat  =  pd.concat([df['body_len'],  df['punct%'],  pd.DataFrame(X_tfidf.toarray(

#  CountVectorizer
count_vect  =  CountVectorizer(analyzer=clean_text)
X_count  =  count_vect.fit_transform(df['body_text'])
X_count_feat  =  pd.concat([df['body_len'],  df['punct%'],  pd.DataFrame(X_count.toarray(

X_count_feat.head()
```

| | body_len | punct% | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 92 | 9.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 24 | 25.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 128 | 4.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 39 | 15.4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 49 | 4.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 8062 columns

```
df.head()
```

| | label | body_text | body_clean_text | body_len | punct% |
|---|---|---|---|---|---|
| **0** | ham | Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g... | [go, jurong, point, crazi, avail, bugi, n, great, world, la, e, buffet, cine, got, amor, wat] | 92 | 9.8 |
| **1** | ham | Ok lar... Joking wif u oni... | [ok, lar, joke, wif, u, oni] | 24 | 25.0 |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ... | [free, entri, 2, wkli, comp, win, fa, cup, final, tkt, 21st, may, 2005, text, fa, 87121, receiv,... | 128 | 4.7 |
| **3** | ham | U dun say so early hor... U c already then say... | [u, dun, say, earli, hor, u, c, alreadi, say] | 39 | 15.4 |
| **4** | ham | Nah I don't think he goes to usf, he lives around here though | [nah, dont, think, goe, usf, live, around, though] | 49 | 4.1 |

## Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.model_selection import train_test_split
print(dir(RandomForestClassifier))
print(RandomForestClassifier()) # exploring hyperparameters
```

```
['__abstractmethods__', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

## 分出测试集和训练集

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf_feat, df['label'], tes
```

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=50, max_depth=20, n_jobs=-1)
rf_model = rf.fit(X_train, y_train)
```

```
sorted(zip(rf_model.feature_importances_, X_train.columns), reverse=True)[0:10]
```

```
[(0.059909847233463084, 'body_len'),
 (0.04583549285541195, 8038),
 (0.034112666319669936, 5683),
 (0.029993193109238304, 3118),
 (0.028679253195039306, 7292),
 (0.022629381428073126, 2018),
 (0.019985468335243332, 6695),
 (0.01856465145632109, 1789),
 (0.017647335891347312, 6976),
 (0.016023078385443746, 391)]
```

```
y_pred = rf_model.predict(X_test)
precision, recall, fscore, support = score(y_test, y_pred, pos_label='spam', aver
```

```
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3),
```

```
Precision: 1.0 / Recall: 0.618 / Accuracy: 0.939
```

## Naive Bayes

## Build Vocab

```
df = pd.read_csv("spam.csv",encoding='latin-1')
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
```

```
X_train, X_val, y_train, y_val = train_test_split(list(df['v2']), list(df['v1']),
print(f'Training Set: X_train Shape: {len(X_train)} | y_train Shape: {len(y_train
print(f'Validation Set: X_val Shape: {len(X_val)} | y_val Shape: {len(y_val)}')
```

```
Training Set: X_train Shape: 4457 | y_train Shape: 4457
Validation Set: X_val Shape: 1115 | y_val Shape: 1115
```

```
from tqdm import tqdm
```

```
class tokenizer:
    def __init__(self, text_data):
        self.vocab = {}
        self.__get_vocab(text_data)

    def convert_text_dataset_to_matrix(self, X):
        vocab = self.vocab
```

```
                vocab = self.vocab
                result = []
                for text in tqdm(X):
                        vector = self.__text_to_vector(text)
                        result.append(vector)
                return result
        def __get_vocab(self, text_data):
                word_id = 0
                for text in text_data:
                        words = text.split()
                        for word in words:
                                word = word.lower()
                                if word not in self.vocab:
                                        self.vocab[word] = word_id
                                        word_id += 1
                print(f'Length of Dictionary: {len(self.vocab)}')

        def __text_to_vector(self, text):
                vocab = self.vocab
                result = list(np.zeros(len(vocab)))
                words = text.split()
                for word in words:
                        word = word.lower()
                        if word in vocab:
                                result[vocab[word]] += 1
                return result


tz = tokenizer(X_train)
```

```
Length of Dictionary: 11706
```

```
X_train_matrix = tz.convert_text_dataset_to_matrix(X_train)
X_val_matrix = tz.convert_text_dataset_to_matrix(X_val)
```

```
100%|██████████| 4457/4457 [00:03<00:00, 1133.90it/s]
100%|██████████| 1115/1115 [00:00<00:00, 1221.71it/s]
```

## Naive Bayes Model

```
from matplotlib import pyplot as plt
从 sklearn.naive_bayes 进口 MultinomialNB
从 sklearn.metrics 进口 accuracy_score，confusion_matrix，classification_报告


def plot_confusion_matrix ( conf_mtrx , classes , cmap = plt.cm.Blues):
        num_class = conf_mtrx.shape [ 0 ]

        图, ax = plt.subplots ()
        im = ax.imshow（conf_mtrx，插值 = '最近' , cmap = cmap）
        ax.figure.colorbar (im, ax = ax)
```

```
        ax.set (xticks = np.arange (num_class), yticks = np.arange (num_class),
                    xticklabels = 类, yticklabels = 类,
                    ylabel = '真实标签', xlabel = '预测标签')

        中间阈值 = conf_mtrx。最大() /   2 。
        对于  行  中的范围（num_class）：
               为  山口  在范围（num_class）：
                       ax.text (col, row,   format (conf_mtrx [row, col],   '.0f'), ha
                                       color = "white" if   conf_mtrx [row, col]> middle
        fig.tight_layout ()
        plt.show()


nb = 多项式NB ()
nb.fit (X_train_matrix, y_train)

      MultinomialNB（alpha = 1.0, class_prior = None, fit_prior = True）


y_pred = nb.predict (X_val_matrix)


打印（'准确度：'，accuracy_score（y_val，y_pred））
打印（'混淆矩阵'）
打印（混淆矩阵（y_val，y_pred））
打印（'分类报告'）
打印（分类报告（y_val，y_pred））

      准确度：0.9775784753363229
      混淆矩阵
      [[948 1]
       [24 142]]
      分类报告
                    精确召回f1-score支持

              火腿 0.98 1.00 0.99 949
             垃圾邮件 0.99 0.86 0.92 166

          精度 0.98 1115
         宏观平均 0.98 0.93 0.95 1115
      加权平均 0.98 0.98 0.98 1115
```

CountVectorizer 和 Complement Naive Bayes (CNB) 表现出最好的准确性。