任务描述：建立一个预测模型，通过从序列中提取特征，对microRNA和gene的序列关系进行预测

数据集：microRNA的序列信息和gene信息已经提供，并且提供了一部分的关系对信息，需要使用这些关系对信息建模，预测测试集中的microRNA和gene序列是否有关系

方法概述：首先从序列中提取3-mer的信息并且计数，获取microRNA和gene的3-mer的分布信息，再将二者拼接做成数据集，使用随机森林进行预测。

[1]import pandas as pd

import itertools

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

from sklearn.metrics import roc\_auc\_score

import os

import joblib

1.数据准备

1.1 数据说明

1.2 数据预处理

1.3 数据集生成

[2]#数据说明

[3]dataset=pd.read\_csv('./datasets/Train.csv')

mirna\_seqdf=pd.read\_csv('./datasets/mirna\_seq.csv')#(['mirna', 'seq']

gene\_seqdf=pd.read\_csv('./datasets/gene\_seq.csv')#'label', 'sequence'

[4]dataset\_mirna=dataset['miRNA']

dataset\_gene=dataset['gene']

dataset\_label=dataset['label']

gene\_index=gene\_seqdf['label'].values.tolist()

gene\_seq=gene\_seqdf['sequence']

mirna\_index=mirna\_seqdf['mirna'].values.tolist()

mirna\_seq=mirna\_seqdf['seq']

1.1 展示序列信息以及标签信息,以及关系对的标签信息

[4]gene\_seq.head

[5]<bound method NDFrame.head of 0

TGCAGCCGCGGGCCCAGGGTGCTGTTGGTGTCCTCAGAAGTGCCGG...

1 AGATGCTTTTTTAAATTTAAGAATAAGACACACAAAACTCTATTAA...

2 CCTGCCCGCGAAGTTGCGGACAGTGCGTGAGAAACCAGCACCCCCT...

3 GGGGGACGGGCAGGGTGGGCCTAGGTTTGAGAGCCCTGGGGCTCCA...

4 ATGACCAGGACAATGAAGACTGCATCAATCGTCATAATGTCTATAT...

...

16122 CAGGTTTGATGCTGCCAAGTTTGTCATGAGATGGCTCTGTAAGCAT...

16123 TAGCCATAAGTATTGGATAGTTGAATCACAGGAATCCTTTTTGTGA...

16124 AGGCATGGAATAGGTGCTCTGACCTCTGACCCTCTAGCCCAGGGAG...

16125 CTCCTTTTTGATCCTGTCCCTCACCCACCTTCACACACACACACGC...

16126 GGCCAGGTCCAGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGG...

Name: sequence, Length: 16127, dtype: object>

[6]gene\_index[0:10]

[6]['A1BG',

'A1CF',

'A2BP1',

'A2LD1',

'A2M',

'A2ML1',

'A4GNT',

'AAAS',

'AACS',

'AADAC']

[7]dataset\_label.head

[7]<bound method NDFrame.head of 0 Functional MTI

1 Functional MTI

2 Non-Functional MTI

3 Functional MTI

4 Non-Functional MTI

...

733 Functional MTI

734 Functional MTI

735 Functional MTI

736 Functional MTI

737 Functional MTI

Name: label, Length: 738, dtype: object>

[8]dataset.columns

[8]Index(['gene', 'miRNA', 'label'], dtype='object')

1.2 数据预处理，特征提取和生成

[9]key\_set={}

key\_set\_T={}

for i in itertools.product('UCGA', repeat =3):#itertools.product('BCDEF', repeat = 2):

#print(i)

obj=''.join(i)

# print(obj)

ky={'{}'.format(obj):0}

key\_set.update(ky)

for i in itertools.product('TCGA', repeat =3):#itertools.product('BCDEF', repeat = 2):

#print(i)

obj=''.join(i)

# print(obj)

ky={'{}'.format(obj):0}

key\_set\_T.update(ky)

[10]def clean\_key\_set(key\_set):

for i,key in enumerate(key\_set):

#print(i,key,key\_set[key])

key\_set[key]=0

return key\_set

def return\_features(n,seq):

clean\_key\_set(key\_set)

key=key\_set

if '\n' in seq:

seq=seq[0:-1]

for i in range(n,len(seq)+1-n):

win=seq[i:i+n]

#print(win)

ori=key\_set['{}'.format(win)]

key\_set['{}'.format(win)]=ori+1

return key\_set

def return\_gene\_features(n,seq):

clean\_key\_set(key\_set\_T)

key=key\_set\_T

if '\n' in seq:

seq=seq[0:-1]

for i in range(n,len(seq)+1-n):

win=seq[i:i+n]

#print(win)

ori=key\_set\_T['{}'.format(win)]

key\_set\_T['{}'.format(win)]=ori+1

return key\_set\_T

1.3 使用拼接方法构建数据集

[11]def construct\_dataset(dataset\_mirna,dataset\_gene):

list\_mirna\_feature=[]

list\_gene\_feature=[]

for i in range(0,len(dataset\_mirna)):

try:

mirna=dataset\_mirna[i]

m\_index=mirna\_index.index(mirna)

#print(m\_index)

mirna\_f=return\_features(3,mirna\_seq[m\_index])

gene=dataset\_gene[i]

g\_index=gene\_index.index(gene)

gene\_f=return\_gene\_features(3, gene\_seq[g\_index])

#print(gene\_f)

mirna\_feature=mirna\_f.copy()

gene\_feature=gene\_f.copy()

list\_mirna\_feature.append(mirna\_feature)

list\_gene\_feature.append(gene\_feature)

except:

mirna=dataset\_mirna[i]

gene=dataset\_gene[i]

print('error detected',i,mirna,gene)

lmpd=pd.DataFrame(list\_mirna\_feature)

lgpd=pd.DataFrame(list\_gene\_feature)

X=pd.concat([lmpd,lgpd],axis=1)

return X

[12]#标签换为数字

Y=[]

for i,label in enumerate(dataset\_label):

if label =='Functional MTI':

Y.append(1)

else:

Y.append(0)

[13]X=construct\_dataset(dataset\_mirna,dataset\_gene)

#print(X)

#lmpd.to\_csv('gene\_features.csv',index=None)

2模型训练&3.模型评估

模型训练切分训练集调参，

使用ACC和F1score作为评估标准

[14]X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y,test\_size=0.8, random\_state=2)

[27]#切分训练集进行调参

def train():

clf = RandomForestClassifier(n\_estimators=30)

clf.fit(X\_train,y\_train)

y\_p=clf.predict(X\_test)

#acc = metrics.accuracy\_score(y\_test,y\_p)

#print('RF\_ACC',acc)

y\_pb=clf.predict\_proba(X\_test)

#print(y\_p)

f1score=metrics.f1\_score(y\_test, y\_p)

print('RF\_F1',f1score)

MCC=metrics.matthews\_corrcoef(y\_test, y\_p)

train()

RF\_F1 0.7986577181208053

[17]#最终模型

clf\_final = RandomForestClassifier(n\_estimators=30)

clf\_final.fit(X,Y)

[17]RandomForestClassifier(n\_estimators=30)

[19]#存储模型与重新调用

joblib.dump(clf\_final, "./model/train\_model.m") #存储

clf\_final = joblib.load("./model/train\_model.m") #调用

4.模型预测

加载测试数据

测试数据生成器

预测结果

结果展示

生成提交文件

[21]#加载测试数据

#test\_filenames = os.listdir("./datasets")

df\_predict=pd.read\_csv('./datasets/Test\_without\_label.csv')

[22]predict\_mirna=df\_predict['miRNA']

predict\_gene=df\_predict['gene']

[23]#测试数据生成器

X\_predict=construct\_dataset(predict\_mirna,predict\_gene)

[24]#预测结果

final\_result=clf\_final.predict(X\_predict)

[25]#结果展示

print(final\_result)

[1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1

1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1

1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1

1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1]

[26]#生成提交文件

df\_predict['results']=final\_result

df\_predict.to\_csv('submission.csv',index=None)