一、项目说明

参加项目：Two Sigma: Using News to Predict Stock Movements
队伍名称：SUFE_NLP_LIANG LI🦁
Leaderboard排名：1177

二、具体实现

1. 环境配置

```
# This Python 3 environment comes with many helpful analytics libraries
installed
# It is defined by the kaggle/python docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
list the files in the input directory

import os
print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.
```

```
import datetime
import random
import time
```

```
import warnings
warnings.filterwarnings('ignore')
from collections import Counter
from datetime import date, datetime

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy

import tensorflow
from keras import backend as K
from keras import optimizers
from keras.optimizers import SGD
from keras.applications.xception import Xception
from keras.callbacks import (Callback, EarlyStopping, ModelCheckpoint,
                             ReduceLROnPlateau)
from keras.engine import InputSpec
from keras.engine.topology import Input, get_source_inputs
from keras.engine.training import Model
from keras.layers import (LSTM, Activation, Add, BatchNormalization,
                          Concatenate, Conv2D, Dense, Dropout, Embedding,
                          Flatten, GlobalAveragePooling2D, Input,
LeakyReLU,
                          MaxPool2D, Permute, Reshape, ZeroPadding2D,
multiply)
from keras.layers.convolutional import Conv2D, Conv2DTranspose,
UpSampling2D
from keras.layers.core import Activation, Dense, Lambda, SpatialDropout2D
from keras.layers.merge import add, concatenate
from keras.layers.normalization import BatchNormalization
from keras.layers.pooling import MaxPooling2D
from keras.legacy import interfaces
from keras.losses import binary_crossentropy, mse
from keras.models import Model, Sequential
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.utils import conv_utils, to_categorical
from keras.utils.data_utils import get_file
from keras.utils.generic_utils import get_custom_objects

from scipy import stats
from scipy.sparse import csr_matrix, hstack
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
```

```
from sklearn.preprocessing import (LabelEncoder, MinMaxScaler,
OneHotEncoder,
                                   QuantileTransformer, RobustScaler,
                                   StandardScaler)

import plotly.graph_objs as go
import plotly.offline as py
import plotly.tools as tls
import seaborn as sb  # visualization
from plotly.tools import FigureFactory as FF
py.init_notebook_mode(connected=True)

%matplotlib inline
```

```
from kaggle.competitions import twosigmanews
env = twosigmanews.make_env()
(market_train_df, news_train_df) = env.get_training_data()

#复制两份数据备用
market_train, news_train = market_train_df.copy(), news_train_df.copy()
market_train_df1, news_train_df1 = market_train_df.copy(),
news_train_df.copy()
```

2. 探索性数据分析

market data EDA

```
#检查变量
market_train_df.dtypes
```

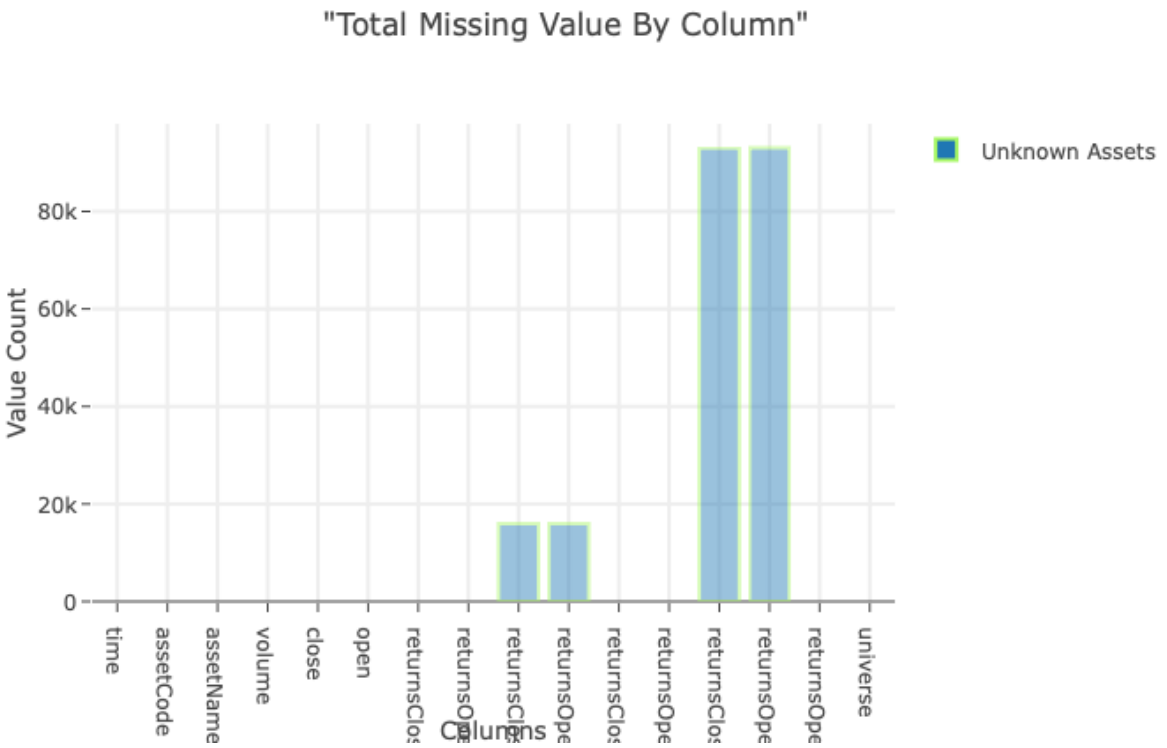
可能需要处理的变量:

- object: assetCode
- category: assetName

```
#检查行列
market_train_df.shape
market_train_df.head()
```

```
plt.style.use("fivethirtyeight")

#Function  统计缺失值
def mis_value_graph(data):
    data = [
    go.Bar(
        x = data.columns,
        y = data.isnull().sum(),
```

```
        name = 'Unknown Assets',
        textfont=dict(size=20),
        marker=dict(
#         color= colors,
        line=dict(
            color=generate_color(),
            width=2,
        ), opacity = 0.45
    )
    ),
    ]
    layout= go.Layout(
        title= '"Total Missing Value By Column"',
        xaxis= dict(title='Columns', ticklen=5, zeroline=False,
gridwidth=2),
        yaxis=dict(title='Value Count', ticklen=5, gridwidth=2),
        showlegend=True
    )
    fig= go.Figure(data=data, layout=layout)
    py.iplot(fig, filename='misvalue')

#Function 缺失值处理，对数值型使用平均值填补
#填补方法有改进空间，也许可以使用算法进行拟合填补
def mis_impute(data):
    for i in data.columns:
        if data[i].dtype == "object":
            data[i] = data[i].fillna("other")
        elif (data[i].dtype == "int64" or data[i].dtype == "float64"):
            data[i] = data[i].fillna(data[i].mean())
        else:
            pass
    return data

#为了方便之后作图填色，引入色彩随机生成器
import random
def generate_color():
    color = '#{:02x}{:02x}{:02x}'.format(*map(lambda x: random.randint(0,
255), range(3)))
    return color

#作图
mis_value_graph(market_train_df)

#修剪缺失值
market_train_df = mis_impute(market_train_df)
market_train_df.isna().sum().to_frame()
```



news data EDA

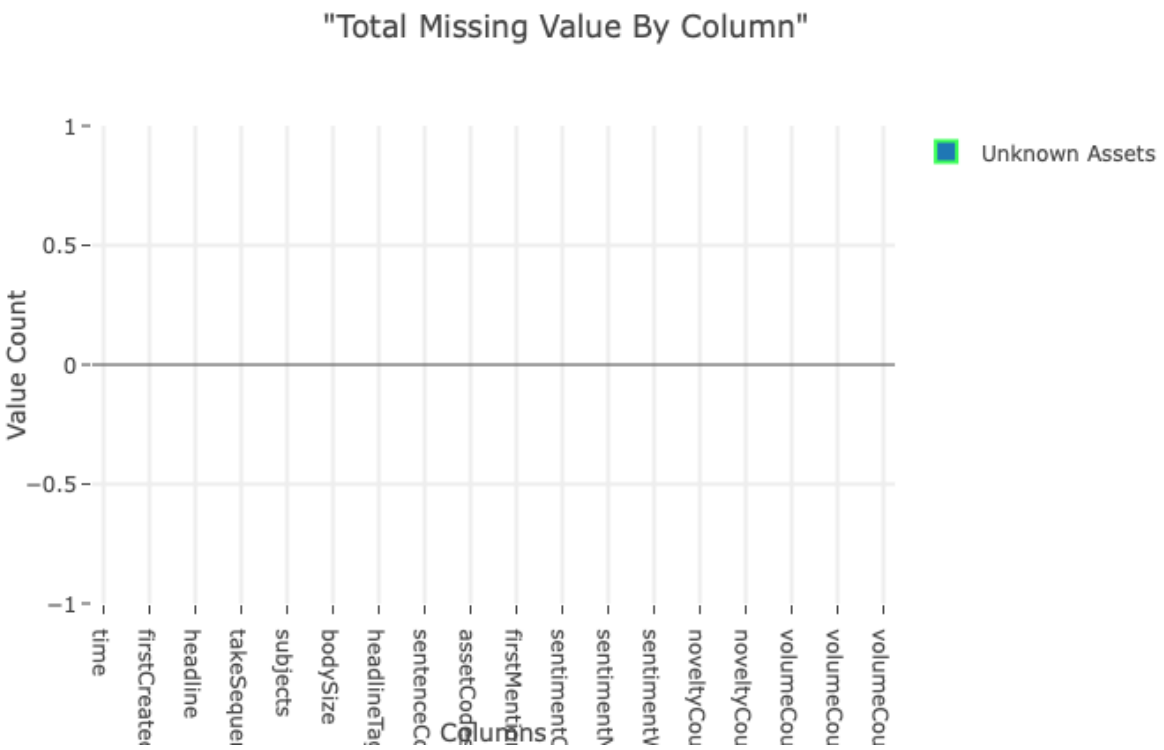
```
#检查变量
news_train_df.dtypes
```

可能需要处理的变量：

- object:
  - sourcId
  - headline
  - headlineTag
- category:
  - provider
  - subjects
  - audiences
  - assetCodes
  - assetName
- bool:
  - marketCommentary

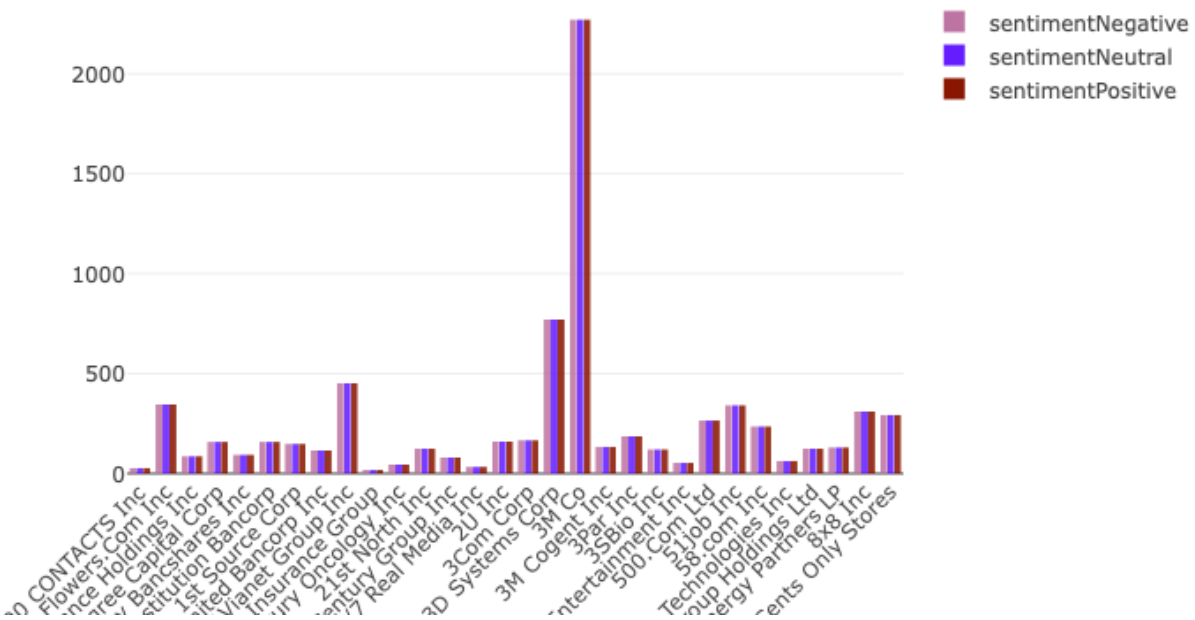
```
#检查行列
news_train_df.shape
```

```
#作图
mis_value_graph(news_train_df)
news_train_df = mis_impute(news_train_df)

#修剪缺失值
news_train_df.isna().sum().to_frame()
```

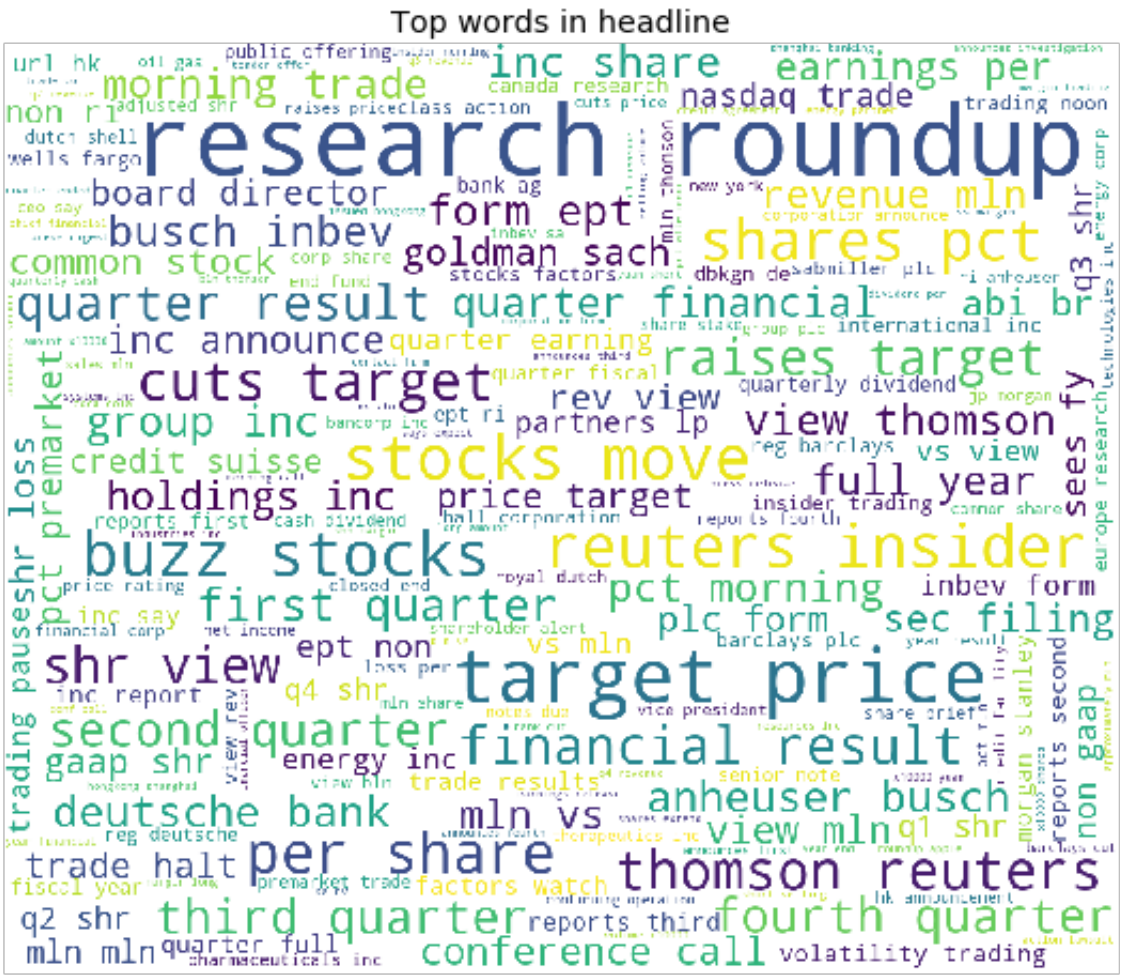



```
#情感分析
news_sentiment_count = news_train_df.groupby(["urgency","assetName"])
[["sentimentNegative","sentimentNeutral","sentimentPositive"]].count()
news_sentiment_count = news_sentiment_count.reset_index()

trace = go.Table(
    header=dict(values=list(news_sentiment_count.columns),
                fill = dict(color='rgba(55, 128, 191, 0.7)'),
                align = ['left'] * 5),
    cells=dict(values=
[news_sentiment_count.urgency,news_sentiment_count.assetName,news_sentiment
_count["sentimentNegative"], news_sentiment_count["sentimentPositive"],
news_sentiment_count["sentimentNeutral"]],
               fill = dict(color='rgba(245, 246, 249, 1)'),
               align = ['left'] * 5))

data = [trace]
py.iplot(data, filename = 'sentiment_table')


trace0 = go.Bar(
```

```
    x= news_sentiment_count.assetName.head(30),
    y=news_sentiment_count.sentimentNegative.values,
    name='sentimentNegative',
    textfont=dict(size=20),
        marker=dict(
        color= generate_color(),
        opacity = 0.87
    )
)
trace1 = go.Bar(
    x= news_sentiment_count.assetName.head(30),
    y=news_sentiment_count.sentimentNeutral.values,
    name='sentimentNeutral',
    textfont=dict(size=20),
        marker=dict(
        color= generate_color(),
        opacity = 0.87
    )
)
trace2 = go.Bar(
    x= news_sentiment_count.assetName.head(30),
    y=news_sentiment_count.sentimentPositive.values,
    name='sentimentPositive',
    textfont=dict(size=20),
    marker=dict(
        color= generate_color(),
        opacity = 0.87
    )
)

data = [trace0, trace1, trace2]
layout = go.Layout(
    xaxis=dict(tickangle=-45),
    barmode='group',
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='angled-text-bar')
```




```
#对倒数100万个做词云
from wordcloud import WordCloud
from nltk.corpus import stopwords
stop = set(stopwords.words('english'))

text = ' '.join(news_train_df['headline'].str.lower().values[-1000000:])
wordcloud = WordCloud(max_font_size=None, stopwords=stop,
background_color='white',
                      width=1200, height=1000).generate(text)
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud)
plt.title('Top words in headline')
plt.axis("off")
plt.show()
```



3. 数据预处理

market data预处理

```
market_train_df.assetCode=market_train_df.assetCode.astype('category')
```

```
cat_cols = ['assetCode']
num_cols = ['volume', 'close', 'open', 'returnsClosePrevRaw1',
'returnsOpenPrevRaw1', 'returnsClosePrevMktres1',
                    'returnsOpenPrevMktres1', 'returnsClosePrevRaw10',
'returnsOpenPrevRaw10', 'returnsClosePrevMktres10',
                    'returnsOpenPrevMktres10']
```

```
#分离出训练集和测试集，用到了副本market train df1
from sklearn.model_selection import train_test_split
train_indices, val_indices =
train_test_split(market_train_df1.index.values,test_size=0.25,
random_state=20)
```

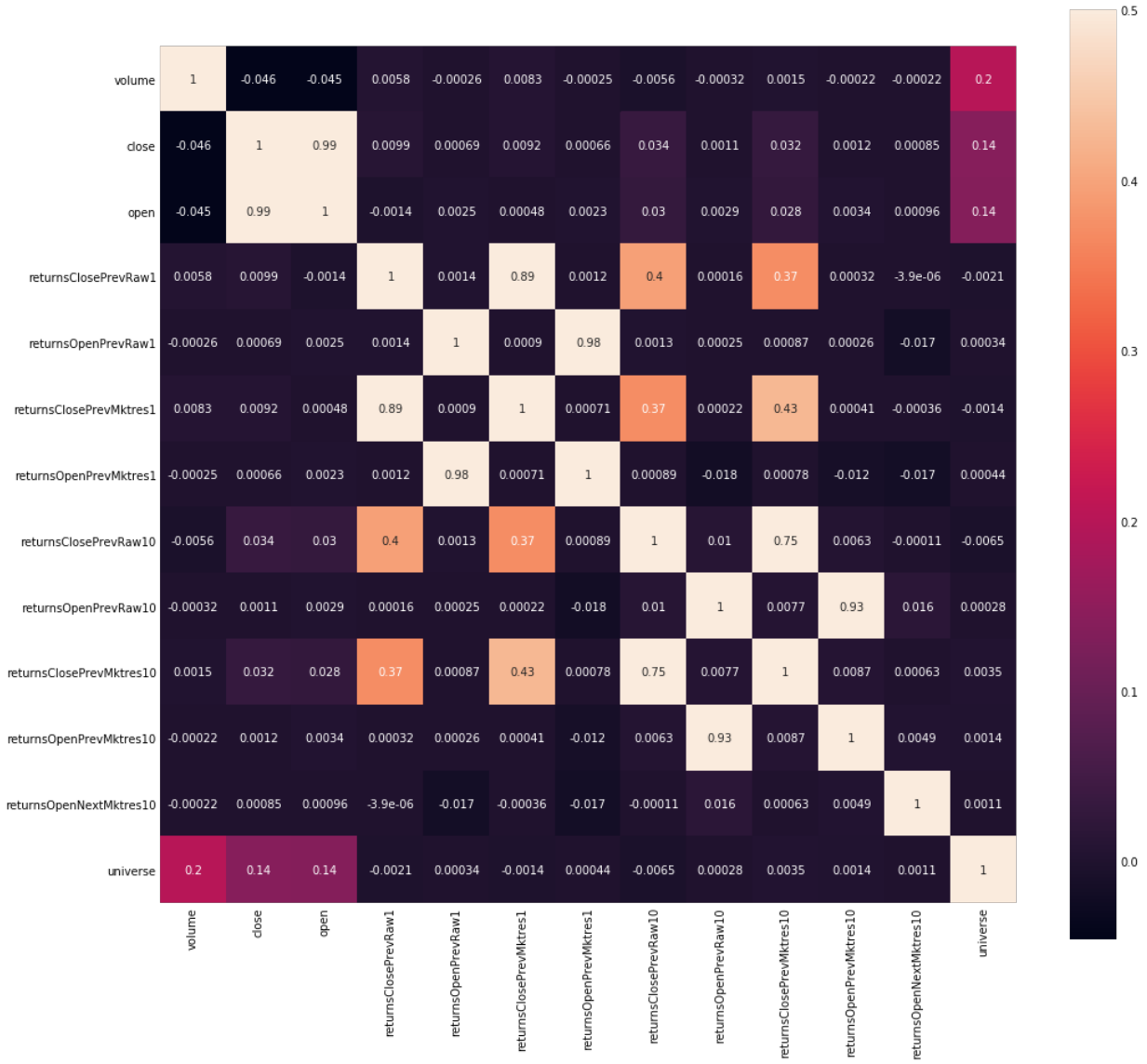
因为股票代码和股票含义相同，所以只需要对股票代码进行编码。

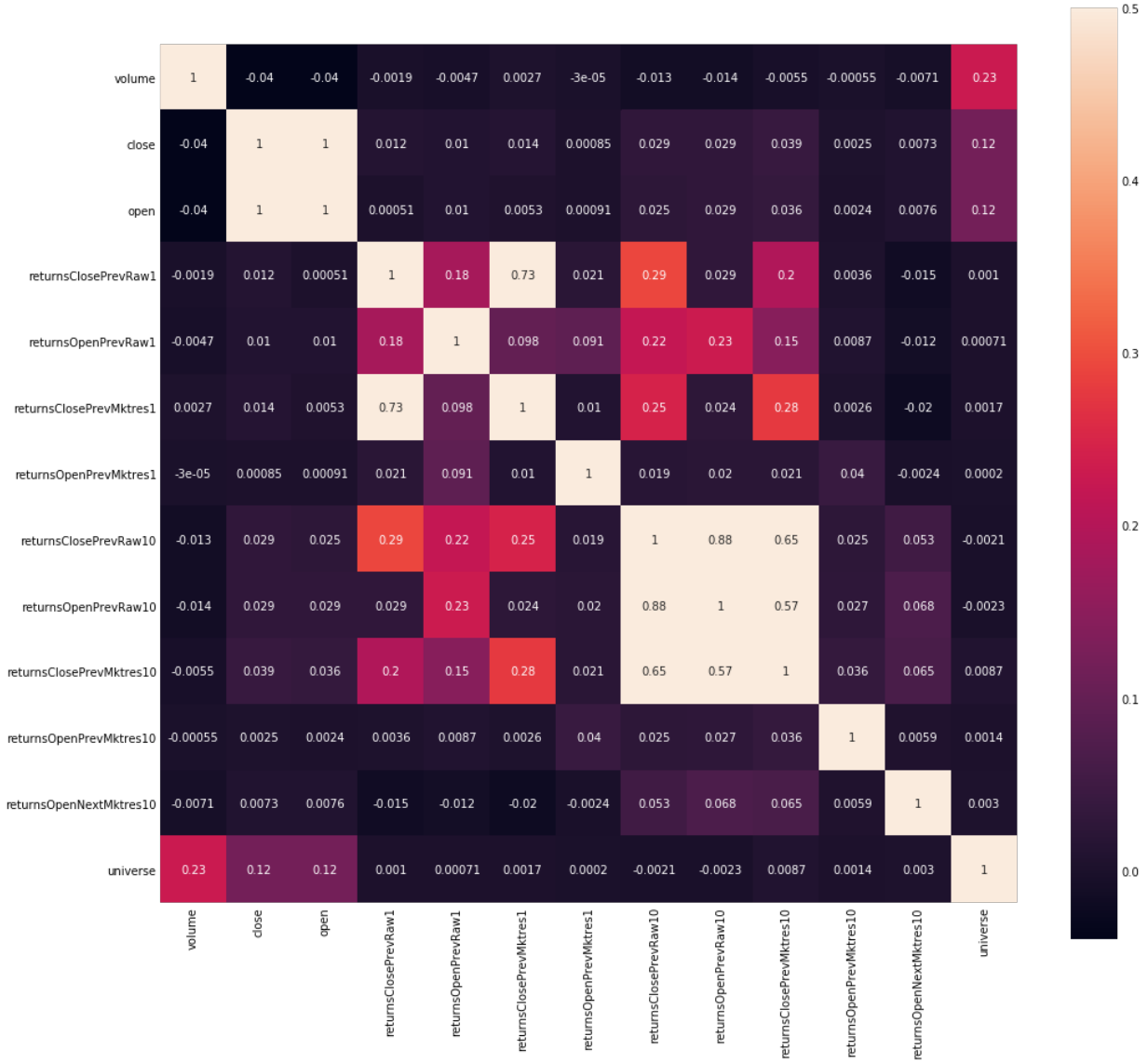
```
def encode(encoder, x):
    len_encoder = len(encoder)
    try:
        id = encoder[x]
    except KeyError:
        id = len_encoder
    return id

encoders = [{} for cat in cat_cols]


for i, cat in enumerate(cat_cols):
    print('encoding %s ...' % cat, end=' ')
    encoders[i] = {l: id for id, l in
enumerate(market_train_df1.loc[train_indices, cat].astype(str).unique())}
    market_train_df1[cat] = market_train_df1[cat].astype(str).apply(lambda
x: encode(encoders[i], x))
    print('Done')

embed_sizes = [len(encoder) + 1 for encoder in encoders] #+1 for possible
unknown assets
```

引入热图，对离群点进行处理

```
Corr_matrix = market_train_df.corr()
fig = plt.figure(figsize=(15,15))
sb.heatmap(Corr_matrix,vmax=0.5,square=True,annot=True)
plt.show()

def remove_outlier_by_percentile(df,col_list,lower_p,upper_p):

    for i in range(len(col_list)):
        df = (df[(df[col_list[i]]<np.percentile(df[col_list[i]],upper_p)) &
(df[col_list[i]]>np.percentile(df[col_list[i]],lower_p))])
    return df

outlier_removal_list = [ 'returnsClosePrevRaw1',
                         'returnsOpenPrevRaw1',
                         'returnsClosePrevRaw10',
                         'returnsOpenPrevRaw10',
                         'returnsOpenNextMktres10']

print('There are ', len(market_train_df['returnsOpenNextMktres10']), 'data
before removing outliers')
market_train_df =
remove_outlier_by_percentile(market_train_df,outlier_removal_list,2,98)
print('There are ', len(market_train_df['returnsOpenNextMktres10']), 'data
after removing outliers')
```

```
# Heat map after removing outliers
Corr_matrix = market_train_df.corr()
fig = plt.figure(figsize=(15,15))
sb.heatmap(Corr_matrix,vmax=0.5,square=True,annot=True)
plt.show()
```






```
market_train_df.shape
```

```
# 对数值型数据正则化
from sklearn.preprocessing import StandardScaler

market_train_df1[num_cols] = market_train_df1[num_cols].fillna(0)
print('scaling numerical columns')

def scale_data(df,features):
    scaler = StandardScaler()
    df[features]=scaler.fit_transform(df[features])
    return df

market_train_df1 = scale_data(market_train_df,num_cols)
```

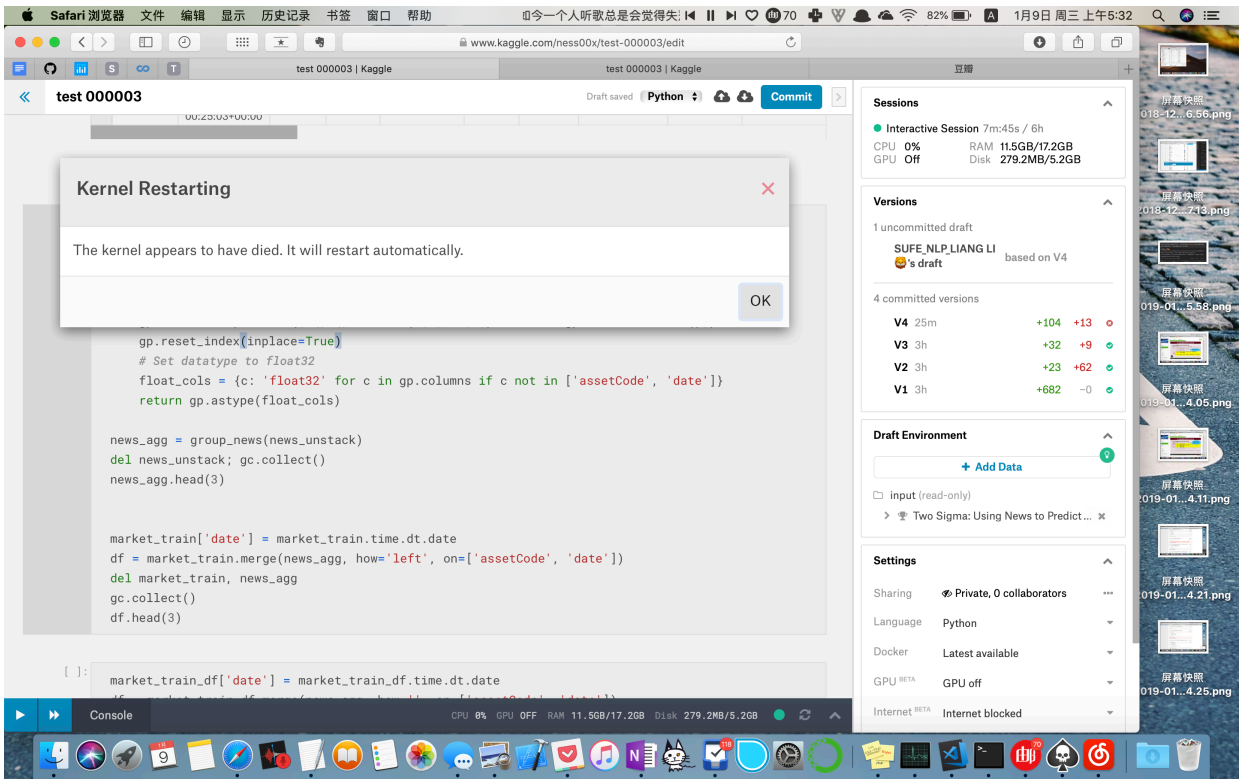
```
market_train_df.head()
```

news data预处理

在处理新闻数据的时候，kernel不停死亡，似乎是因为内存溢出的原因。 即使使用少部分数据运行，到了合并新闻数据与市场数据的时候，kernel依旧会狗带，所以最终没有把新闻数据纳入框架中。

```
import gc

def preprocess_news(news_train_df):
    drop_list = [
        'audiences', 'subjects', 'assetName',
        'headline', 'firstCreated', 'sourceTimestamp',
    ]
    news_train_df.drop(drop_list, axis=1, inplace=True)

    # Factorize categorical columns
    for col in ['headlineTag', 'provider', 'sourceId']:
        news_train_df[col], uniques = pd.factorize(news_train_df[col])
        del uniques

    # Remove {} and '' from assetCodes column
    news_train_df['assetCodes'] = news_train_df['assetCodes'].apply(lambda
x: x[1:-1].replace("'", ""))
    return news_train_df

news_train_df = preprocess_news(news_train_df)

def unstack_asset_codes(news_train_df):
    codes = []
    indexes = []
    for i, values in news_train_df['assetCodes'].iteritems():
        explode = values.split(", ")
        codes.extend(explode)
        repeat_index = [int(i)]*len(explode)
        indexes.extend(repeat_index)
    index_df = pd.DataFrame({'news_index': indexes, 'assetCode': codes})
    del codes, indexes
    gc.collect()
    return index_df

index_df = unstack_asset_codes(news_train_df)
index_df.head()

def merge_news_on_index(news_train_df, index_df):
    news_train_df['news_index'] = news_train_df.index.copy()

    # Merge news on unstacked assets
    news_unstack = index_df.merge(news_train_df, how='left',
on='news_index')
    news_unstack.drop(['news_index', 'assetCodes'], axis=1, inplace=True)
    return news_unstack
```

```
news_unstack = merge_news_on_index(news_train_df, index_df)
del news_train_df, index_df
gc.collect()
news_unstack.head(3)

def group_news(news_frame):
    news_frame['date'] = news_frame.time.dt.date  # Add date column

    aggregations = ['mean']
    gp = news_frame.groupby(['assetCode', 'date']).agg(aggregations)
    gp.columns = pd.Index(["{}_{}".format(e[0], e[1]) for e in
gp.columns.tolist()])
    gp.reset_index(inplace=True)
    # Set datatype to float32
    float_cols = {c: 'float32' for c in gp.columns if c not in
['assetCode', 'date']}
    return gp.astype(float_cols)

news_agg = group_news(news_unstack)
del news_unstack; gc.collect()
news_agg.head(3)

#合并news和market数据
market_train['date'] = market_train.time.dt.date
df = market_train.merge(news_agg, how='left', on=['assetCode', 'date'])
del market_train, news_agg
gc.collect()
df.head(3)
```



4. 神经网路

```
from keras.models import Model
from keras.layers import Input, Dense, Activation,Embedding, Concatenate,
Flatten, BatchNormalization
from keras.losses import binary_crossentropy, mse
# 分类变量处理
categorical_inputs = []
for cat in cat_cols:
    categorical_inputs.append(Input(shape=[1], name=cat))

categorical_embeddings = []
for i, cat in enumerate(cat_cols):
    categorical_embeddings.append(Embedding(embed_sizes[i], 10)
(categorical_inputs[i]))

categorical_logits = Flatten()(categorical_embeddings[0])
categorical_logits = Dense(32,activation='relu')(categorical_logits)
categorical_logits =Dropout(0.5)(categorical_logits)
categorical_logits =BatchNormalization()(categorical_logits)
categorical_logits = Dense(32,activation='relu')(categorical_logits)

#数值变量处理
numerical_inputs = Input(shape=(11,), name='num')
numerical_logits = numerical_inputs
numerical_logits = BatchNormalization()(numerical_logits)

numerical_logits = Dense(128,activation='relu')(numerical_logits)
numerical_logits=Dropout(0.5)(numerical_logits)
numerical_logits = BatchNormalization()(numerical_logits)
numerical_logits = Dense(128,activation='relu')(numerical_logits)
numerical_logits = Dense(64,activation='relu')(numerical_logits)

logits = Concatenate()([numerical_logits,categorical_logits])
logits = Dense(64,activation='relu')(logits)
out = Dense(1, activation='sigmoid')(logits)

model = Model(inputs = categorical_inputs + [numerical_inputs],
outputs=out)
model.compile(optimizer='adam',loss=binary_crossentropy)
```

```
#对自变量做筛选
def get_input(market_train, indices):
    X_num = market_train.loc[indices, num_cols].values
    X = {'num':X_num}
    for cat in cat_cols:
        X[cat] = market_train.loc[indices, cat_cols].values
    y = (market_train.loc[indices,'returnsOpenNextMktres10'] >= 0).values
```

```
    r = market_train.loc[indices,'returnsOpenNextMktres10'].values
    u = market_train.loc[indices, 'universe']
    d = market_train.loc[indices, 'time'].dt.date
    return X,y,r,u,d

# r, u and d are used to calculate the scoring metric
X_train,y_train,r_train,u_train,d_train = get_input(market_train_df1,
train_indices)
X_valid,y_valid,r_valid,u_valid,d_valid = get_input(market_train_df1,
val_indices)
```

```
market_train.head()
```

```
class SWA(keras.callbacks.Callback):

    def __init__(self, filepath, swa_epoch):
        super(SWA, self).__init__()
        self.filepath = filepath
        self.swa_epoch = swa_epoch

    def on_train_begin(self, logs=None):
        self.nb_epoch = self.params['epochs']
        print('Stochastic weight averaging selected for last {} epochs.'
              .format(self.nb_epoch - self.swa_epoch))

    def on_epoch_end(self, epoch, logs=None):

        if epoch == self.swa_epoch:
            self.swa_weights = self.model.get_weights()

        elif epoch > self.swa_epoch:
            for i in range(len(self.swa_weights)):
                self.swa_weights[i] = (self.swa_weights[i] *
                    (epoch - self.swa_epoch) + self.model.get_weights()
[i])/((epoch - self.swa_epoch)  + 1)

        else:
            pass

    def on_train_end(self, logs=None):
        self.model.set_weights(self.swa_weights)
        print('Final model parameters set to stochastic weight average.')
        self.model.save_weights(self.filepath)
        print('Final stochastic averaged weights saved to file.')

class SnapshotCallbackBuilder:
    def __init__(self, nb_epochs, nb_snapshots, init_lr=0.1):
        self.T = nb_epochs
```

```
        self.M = nb_snapshots
        self.alpha_zero = init_lr

    def get_callbacks(self, model_prefix='Model'):

        callback_list = [

 callbacks.ModelCheckpoint("model.hdf5",monitor='val_my_iou_metric',
                                   mode = 'max', save_best_only=True,
verbose=1),
            swa,

 callbacks.LearningRateScheduler(schedule=self._cosine_anneal_schedule)
        ]

        return callback_list

    def _cosine_anneal_schedule(self, t):
        cos_inner = np.pi * (t % (self.T // self.M))  # t - 1 is used when
t has 1-based indexing.
        cos_inner /= self.T // self.M
        cos_out = np.cos(cos_inner) + 1
        return float(self.alpha_zero / 2 * cos_out)
```

```
#避免过拟合
from keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau
check_point = ModelCheckpoint('model.hdf5',verbose=True,
save_best_only=True)
early_stop = EarlyStopping(patience=5,verbose=True)
```

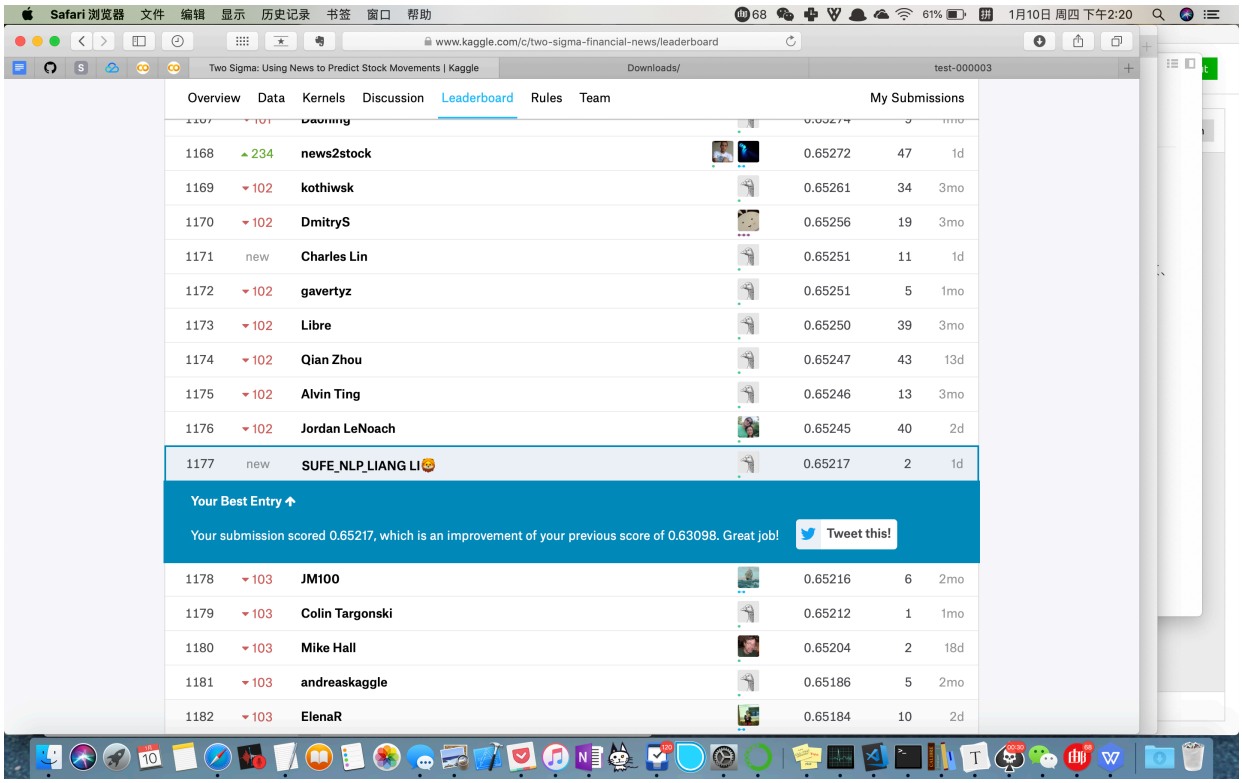
```
#模型训练
model.fit(X_train,y_train.astype(int),
          validation_data=(X_valid,y_valid.astype(int)),
          epochs=5,
          verbose=True,
          callbacks=[early_stop,check_point])
```

5. 最终预测

```
days = env.get_prediction_days()
n_days = 0
prep_time = 0
prediction_time = 0
packaging_time = 0
predicted_confidences = np.array([])
for (market_obs_df, news_obs_df, predictions_template_df) in days:
```

```
    n_days +=1
    print(n_days,end=' ')

    t = time.time()

    market_obs_df['assetCode_encoded'] =
market_obs_df[cat].astype(str).apply(lambda x: encode(encoders[i], x))

    market_obs_df[num_cols] = market_obs_df[num_cols].fillna(0)
    market_obs_df[num_cols] = scaler.transform(market_obs_df[num_cols])
    X_num_test = market_obs_df[num_cols].values
    X_test = {'num':X_num_test}
    X_test['assetCode'] = market_obs_df['assetCode_encoded'].values

    prep_time += time.time() - t

    t = time.time()
    market_prediction = model.predict(X_test)[:,0]*2 -1
    predicted_confidences = np.concatenate((predicted_confidences,
market_prediction))
    prediction_time += time.time() -t

    t = time.time()
    preds =
pd.DataFrame({'assetCode':market_obs_df['assetCode'],'confidence':market_pr
ediction})
    # insert predictions to template
    predictions_template_df =
predictions_template_df.merge(preds,how='left').drop('confidenceValue',axis
=1).fillna(0).rename(columns={'confidence':'confidenceValue'})
    env.predict(predictions_template_df)
    packaging_time += time.time() - t
```

```
#提交
env.write_submission_file()
total = prep_time + prediction_time + packaging_time
print(f'Preparing Data: {prep_time:.2f}s')
print(f'Making Predictions: {prediction_time:.2f}s')
print(f'Packing: {packaging_time:.2f}s')
print(f'Total: {total:.2f}s')
```


Safari 浏览器 文件 编辑 显示 历史记录 书签 窗口 帮助
1月10日 周四 下午2:20
www.kaggle.com/c/two-sigma-financial-news/leaderboard
Two Sigma: Using News to Predict Stock Movements | Kaggle
Downloads/
test-000003
Overview Data Kernels Discussion Leaderboard Rules Team My Submissions
1168 ▲234 news2stock 0.65272 47 1d
1169 ▼102 kothiwsk 0.65261 34 3mo
1170 ▼102 DmitryS 0.65256 19 3mo
1171 new Charles Lin 0.65251 11 1d
1172 ▼102 gavertyz 0.65251 5 1mo
1173 ▼102 Libre 0.65250 39 3mo
1174 ▼102 Qian Zhou 0.65247 43 13d
1175 ▼102 Alvin Ting 0.65246 13 3mo
1176 ▼102 Jordan LeNoach 0.65245 40 2d
1177 new SUFE_NLP_LIANG LI 0.65217 2 1d
Your Best Entry
Your submission scored 0.65217, which is an improvement of your previous score of 0.63098. Great job!
Tweet this!
1178 ▼103 JM100 0.65216 6 2mo
1179 ▼103 Colin Targonski 0.65212 1 1mo
1180 ▼103 Mike Hall 0.65204 2 18d
1181 ▼103 andreaskaggle 0.65186 5 2mo
1182 ▼103 ElenaR 0.65184 10 2d