Kaggle

January 6, 2019

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
!pip install wordcloud
from wordcloud import WordCloud
import plotly.plotly as py
import plotly.graph_objs as go
import sklearn
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.6/dist-packages (1.4.1)
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from wordcloud) (5.1.0
Requirement already satisfied: matplotlib in /usr/lib/python3/dist-packages (from wordcloud) (1
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.6/dist-packages (from wor
```

In [14]:
```
market_train_df = pd.read_csv("marketdata_sample.csv")
```

In [40]:
```
display(market_train_df)
```

In [41]:
```
market_train_df.head()
```

In [42]:
```
news_train_df = pd.read_csv('news_sample.csv')
```

In [43]:
```
news_train_df.head()
```

In [44]:
```
market_train_df['price_diff'] = market_train_df['close'] - market_train_df['open']
grouped = market_train_df.groupby('time').agg({'price_diff': ['std', 'min']}).reset_in
```

In [45]:
```
print(f"Average standard deviation of price change within a day in {grouped['price_di
```

```
Average standard deviation of price change within a day in 0.5075.
```

In [46]:
```
market_train_df.sort_values('price_diff')[:10]
```

In [47]:
```
market_train_df['close_to_open'] = np.abs(market_train_df['close']/market_train_df['o
```

In [48]:
```
print(f"In {(market_train_df['close_to_open'] >= 1.2).sum()} lines price increased by
```

```
In 0 lines price increased by 20% or more.
```

In [49]:
```
print(f"In {(market_train_df['close_to_open'] <= 0.8).sum()} lines price decreased by
```

```
In 0 lines price decreased by 20% or more.
```

In [50]:
```
print(f"In {(market_train_df['close_to_open'] >= 2).sum()} lines price increased by 1
```

```
In 0 lines price increased by 100% or more.
```

In [51]:
```
print(f"In {(market_train_df['close_to_open'] <= 0.5).sum()} lines price decreased by
```

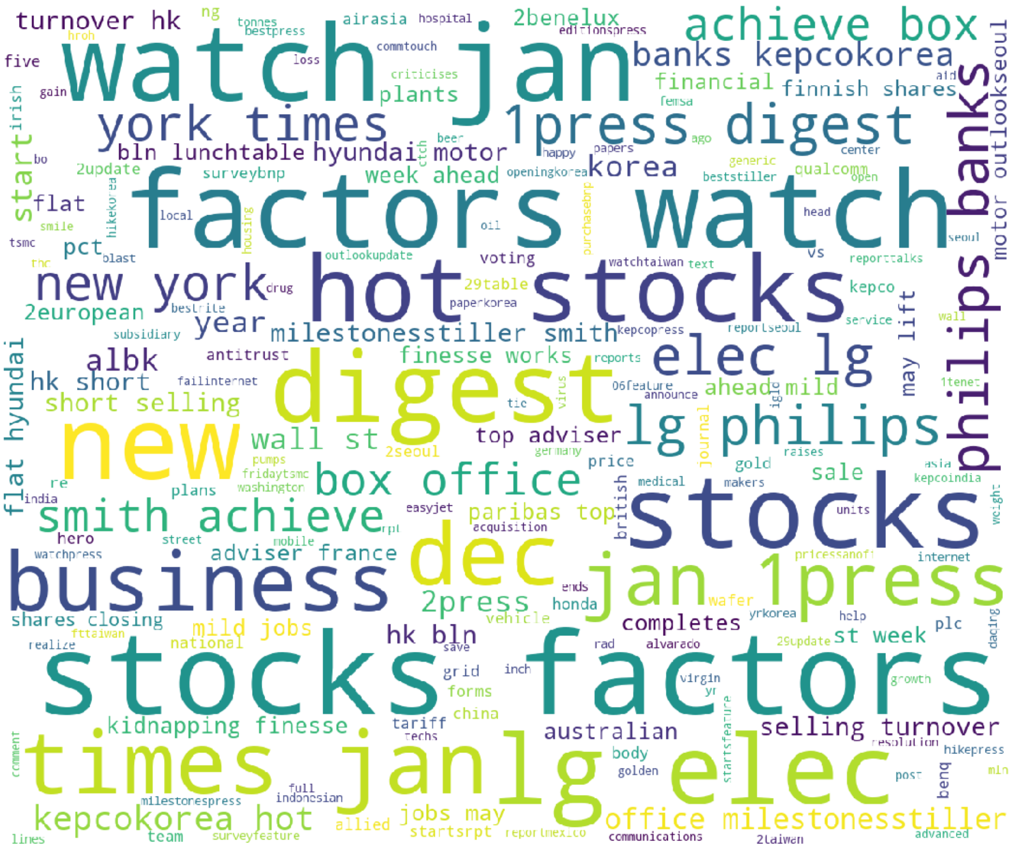
```
In 0 lines price decreased by 100% or more.
```

In [52]:
```
news_train_df.head()
```

In [53]:
```
text=''.join(news_train_df['headline'].str.lower().values[-1000000:])
wordcloud = WordCloud(max_font_size= None, 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()
```

Out[53]:

Top words in headline

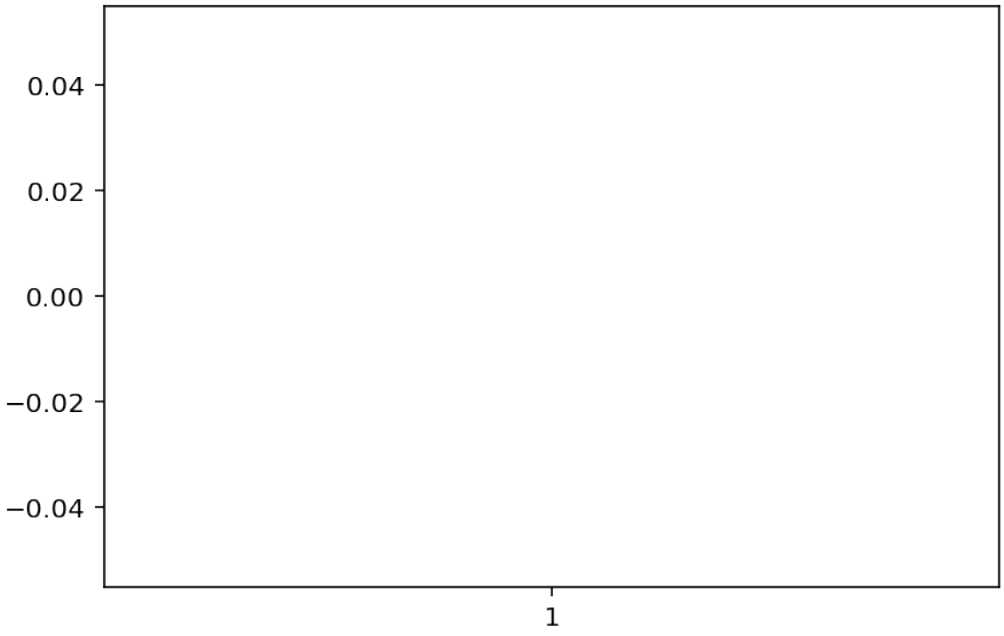


In [54]: news_train_df = news_train_df.loc[news_train_df['time'] >= '2010-01-01 22:00:00+0000']

In [55]: news_train_df['Sentence_word_count'] = news_train_df['wordCount']/news_train_df['sente

In [56]: plt.boxplot(news_train_df['Sentence_word_count'][news_train_df['Sentence_word_count']

Out[56]:

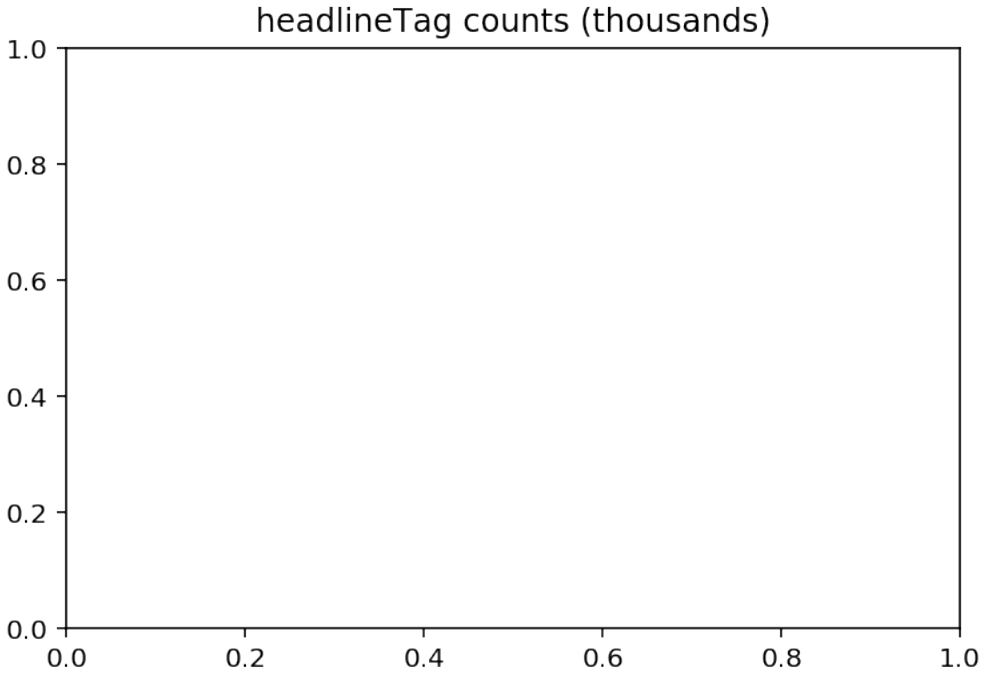


In [57]:
```
news_train_df['provider'].value_counts().head(10)
```

Out[57]:
```
Series([], Name: provider, dtype: int64)
```

In [58]:
```
(news_train_df['headlineTag'].value_counts() / 1000)[:10];
plt.title('headlineTag counts (thousands)');
```

Out[58]:



```
In [59]: for i,j in zip([-1,0,1],['Positive','Neutral','Negative']):
             df_sentiment = news_train_df.loc[news_train_df['sentimentClass']==i, 'assetName']
             print(f'Top mentioned companies for {j} sentiment are:')
             print(df_sentiment.value_counts().head(5))
             print('')

Top mentioned companies for Positive sentiment are:
Series([], Name: assetName, dtype: int64)

Top mentioned companies for Neutral sentiment are:
Series([], Name: assetName, dtype: int64)

Top mentioned companies for Negative sentiment are:
Series([], Name: assetName, dtype: int64)



In [60]: market_train_df.head()

In [19]: def dataprep(market_train_df,news_train_df):
             market_train_df['time'] = market_train_df.time.dt.date
             market_train_df['returnsOpenPrevRaw1_to_volume'] = market_train_df['returnsOpenPre
             market_train_df['close_to_open'] = market_train_df['close']/market_train_df['open
             market_train_df['volume_to_mean'] = market_train_df['volume']/market_train_df['vol
             news_train_df['Sentence_word_count'] = news_train_df['wordCount']/news_train_df['s
             news_train_df['time']  = news_train_df.time.dt.date
             news_train_df['sourceTimestamp'] = news_train_df.sourceTimestamp.dt.hour
             news_train_df['firstCreated'] = news_train_df.firstCreated.dt.date
             news_train_df['assetCodesLen'] = news_train_df['assetCodes'].map(lambda x: len(eva
             news_train_df['assetCodes'] = news_train_df['assetCodes'].map(lambda x: list(eval
             news_train_df['headlineLen'] = news_train_df['headline'].apply(lambda x: len(x))
             news_train_df['assetCodesLen'] = news_train_df['assetCodesLen'].apply(lambda x: le
             news_train_df['asset_sentiment_count'] = news_train_df.groupby(['assetName','senti
             news_train_df['asset_sentiment_mean'] = news_train_df.groupby(['assetName','sentim
             lbl = {k: v for v, k in enumerate(news_train_df['headline'].unique())}
             news_df['headlineTagT'] = news_train_df['headlineTagT'].map(lbl)
             kcol = ['firstCreated','assetCodes']
             news_train_df = news_train_df(kcol, as_index= False).mean()
             market_train_df = pd.merge(news_train_df,market_train_df, how='left', left_on=['ti
             lbl = {k: v for v, k in enumerate(market_df['assetCode'].unique())}
             market_train_df['assetCodeT'] = market_train_df['assetCode'].map(lbl)
             market_train_df = market_train_df.dropna(axis=0)
             return market_train_df
             market_train_df.drop(['price_diff','assetName_mean_open', assetName_mean_close], a
             market_train_df = dataprep(market_train_df,news_train_df)
             print(market_train_df.shape)
```

```
up = market_train.returnsOpenNextMktres10 >= 0
fcol = [c for c in market_train_df.columns if c not in ['assetCode', 'assetCodes'
                                                        'firstCreated', 'headline', 'headlineTag
                                                        'returnsOpenNextMktres10', 'sourceId', '
X = market_train_df[fcol].values
up = up.values
r = market_train_df.returnsOpenNextMktres10.values

mins = np.min(X, axis=0)
maxs = np.max(X, axis=0)
rng = maxs - mins
X = 1 - ((maxs - X) /  rng)
```