



PREDICTING A SMALL CAP COMPANY GROWTH USING PYTHON LIBRARIES

Prof. G. Shankarlingam
Dean, Faculty of Engineering
Chaitanya Deemed to be University
Warangal, Telangana

K. Thirupathi Reddy
Assistant Professor
Chaitanya Deemed to be University
Warangal, Telangana

ABSTRACT

Predicting a small cap company growth is always a difficult task but by using some technology we can predict it successfully. First we need to collect data from various sources and create graphs for data then predict the stock growth according to the market to the market conditions and company performance.

To predict the stock growth we need to use some algorithms like linear regression algorithm, support vector machine learning algorithm and naïve bayes algorithms. These three algorithms are important to analyses the stock growth in a successful way. To predict the stock growth we need to use python coding language. The pre defined algorithms and libraries are helps to analyses the stock growth in a easier way.

In this journal we practically shown you the code of python and the successful execution images also.

Keywords: smallcap, prediction, regression algorithm, machine learning, google colab.

I. Introduction

Investing in stock market provides many investors a way to earn money in a short span of time. Many of the new investors or aspiring investors fall short in understanding the stock market and get confused and often frustrated, leading them to make mistakes and incur losses which further demotivates them from investing and as a result of this, they resort to traditional ways of investing in bank fixed deposits or such which yield them not much returns. The purpose of this study is to provide a clear vision on how to navigate through the stock market with a view to make higher profits in a short span of time, or how to have moderate profits with moderate risk factor governing the investments made by the investor. The knowledge about the financial investment is increasing amongst the investors who are looking for new ways of income other than job and to earn additional income other than salary, as they want to increase their standard of living. Investing in stock market is highly risk oriented task but high returns on the investment is obtained through vigilance on market and careful investing.

II. LITERATURE REVIEW

Swapna (2016): The study creates awareness to the investors about the stock to invest in the best sector as it calculates the risk and return of particular stocks and also to assess the performance of return and risk of those companies. The objectives of investment is to get earn maximize return. More over the investment is depend upon the investor how much to invest in the stocks or companies stock based on risk. The author suggests to invest in Wipro is best and good to invest which lowest risk and highest return. Suresh and Harshitha (2017): This research is about comparing risk and return relationship of stocks by using Markowitz and Sharpe's model. The study aims to identify the level of deviation in returns by comparing these two models and to check whether the results obtained are constant or not. The result of beta will allow the investors to know about the market risk of the particular investments. The research shows that both the models give almost the same value for both individual return and risk and also portfolio. Suresh (2018): The study was organized to analyse the risk and returns of selected logistic stocks listed on BSE and to compare their performance against the benchmark for the period of 5 years i.e. from 1st January 2013 to 31st December 2017. This paper analyses the performance of logistic sector taking BSE Sensex as benchmark. This study is based on secondary data collected from BSE. The data were collected based on monthly prices of the logistic sector and with the help of monthly prices, annual returns were calculated for a period of five years.

III. OBJECTIVES OF THE STUDY

To analysis the return and risk of the selected Small cap stocks To compare the return and risk of selected small cap stocks against benchmark To rank all the stocks based on return and risk To construct portfolio and to provide necessary suggestions based on the study.

IV. RESEARCH METHODOLOGY:

The study mainly focuses on the price movement of selected small cap stocks. The study is descriptive in nature. For the equity analysis, the data of monthly share price are collected from the BSE Sensex. The data were collected for past the past 5 year i.e. JAN-2014 to DEC-2018. Various tools such as Return, Standard deviation, Beta, Alpha etc., the data are collected from BSE Sensex, journals and websites.

A. Duration of the study:

The data is collected for a period of 5 years starting from 1st Jan 2014 to 31st Dec 2018

B. Data collection

The study is based on secondary data collected from BSE website. Data is collected for a period of 5 years (i.e. from 1 st January 2014 to 31st December 2018) from Media and Entertainment sector listed in BSE. Additionally the data are also collected from newspaper, websites, journals, books reports by researchers and scholars

C. Selection of sample companies

DCB Bank - RSS software - TATA Elxsi - Aarti Drugs - Wintac Pharmaceuticals

D. Analytical tools for data analysis

1. Return

2. Standard deviation
3. Variance
4. Beta
5. Alpha
6. Correlation

E. Limitations of the study

The study is limited to only one sectors from BSE→ Sensex. Analysis is based on secondary data collected from→ BSE website, published literature etc. Only five companies have been selected for→ conducting this study.

Alpha and Beta:

Alpha represents how much an investment's actual return exceeded its expected return, based on its risk level. Alpha is used to evaluate whether an investment outperformed a certain benchmark. Beta, on the other hand, measures how volatile an asset is compared to the overall market.

DATA ANALYSIS AND INTERPRETATION

Interpretation:

The above graph shows that all the small cap stocks has given the positive returns during the study period. Aarti drugs gave the highest return of 84.51% for the period 2014 to 2018 followed by RSS Software, TATA Elxsi and Wintac Pharmaceuticals. DCB bank has given the lowest return of 31.82% during the same period.

Interpretation:

From the above table it can be found that DCB bank has the lowest risk followed by Wintac Pharmaceuticals, TATA Elxsi and Aarti Drugs. The stock which has highest risk is RSS Software.

Portfolio Return and Risk:

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

from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))

test.csv
train.csv
```

```
import nltk
import numpy as np # Linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from nltk import word_tokenize, ngrams
from sklearn import ensemble
from sklearn.model_selection import KFold
```

```

from sklearn.metrics import log_loss
import xgboost as xgb

eng_stopwords = set(stopwords.words('english'))
color = sns.color_palette()

%matplotlib inline

pd.options.mode.chained_assignment = None # default='warn'

```

```

In [3]:
train_df = pd.read_csv('../input/train.csv')
test_df = pd.read_csv('../input/test.csv')
print(train_df.shape)
print(test_df.shape)

```

```

(2142, 6)
(516, 4)

```

```

In [4]:
train_df.head()

```

	Unnamed: 0	description_x	description_y	ticker_x	ticker_y	same_security
0	0	first trust dow jones internet	first trust dj internet idx	FDN	FDN	True
1	1	schwab intl large company index etf	schwab strategic tr fundamental intl large co ...	FNDF	FNDF	True
2	2	vanguard small cap index adm	vanguard small-cap index fund inst	VSMAX	VSCIX	False
3	3	duke energy corp new com new isin #us4 sedol #...	duke energy corp new com new isin #us26441c204...	DUK	DUK	True
4	4	visa inc class a	visa inc.	V	V	True

```

In [5]:

```

```

train_df.rename(columns
= {'description_x': 'question1', 'description_y': 'question2', 'same_security': 'is_similar'}, inplace=True)

```

```

In [6]: train_df.head()

```

	Unnamed: 0	question1	question2	ticker_x	ticker_y	is_similar
0	0	first trust dow jones internet	first trust dj internet idx	FDN	FDN	True
1	1	schwab intl large company index etf	schwab strategic tr fundamental intl large co ...	FNDF	FNDF	True
2	2	vanguard small cap index adm	vanguard small-cap index fund inst	VSMAX	VSCIX	False
3	3	duke energy corp new com new isin #us4 sedol #...	duke energy corp new com new isin #us26441c204...	DUK	DUK	True
4	4	visa inc class a	visa inc.	V	V	True

```

In [7]:
test_df.head()

```

```

Out[7]:

```

	test_id	description_x	description_y	same_security
0	0	semtech corp	semtech corporation	NaN
1	1	vanguard mid cap index	vanguard midcap index - a	NaN
2	2	spdr gold trust gold shares	spdr gold trust spdr gold shares	NaN
3	3	vanguard total bond index adm	vanguard total bond market index	NaN
4	4	oakmark international fund class i	oakmark international cl i	NaN

In [8]:

```
test_df.rename(columns={'description_x':'question1','description_y':'question2','same_security':'is_similar'},inplace=True)
```

In [9]:

```
test_df.head()
```

Out[9]:

	test_id	question1	question2	is_similar
0	0	semtech corp	semtech corporation	NaN
1	1	vanguard mid cap index	vanguard midcap index - a	NaN
2	2	spdr gold trust gold shares	spdr gold trust spdr gold shares	NaN
3	3	vanguard total bond index adm	vanguard total bond market index	NaN
4	4	oakmark international fund class i	oakmark international cl i	NaN

In [10]:

```
## Target Exploration
```

```
is_sim = train_df['is_similar'].value_counts()
```

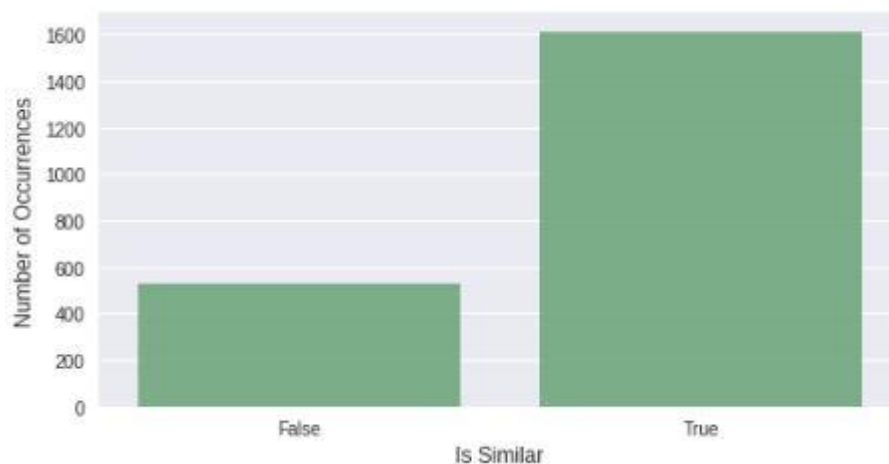
```
plt.figure(figsize=(8,4))
```

```
sns.barplot(is_sim.index, is_sim.values, alpha=0.8, color=color[1])
```

```
plt.ylabel('Number of Occurrences', fontsize=12)
```

```
plt.xlabel('Is Similar', fontsize=12)
```

```
plt.show()
```



In [11]:

```
is_sim/is_sim.sum()
```

Out[11]:

```
True      0.753035
```

```
False     0.246965
```

```
Name: is_similar, dtype: float64
```

In [12]:

```
all ques_df = pd.DataFrame(pd.concat([train_df['question1'], train_df['question2']]))
```

```
all ques_df.columns = ["questions"]
```

```
all ques_df["num_of_words"] = all ques_df["questions"].apply(lambda x: len(str(x).split()))
```

In [13]:

```
count_str = all ques_df["num_of_words"].value_counts()
```

```
plt.figure(figsize=(12,6))
```

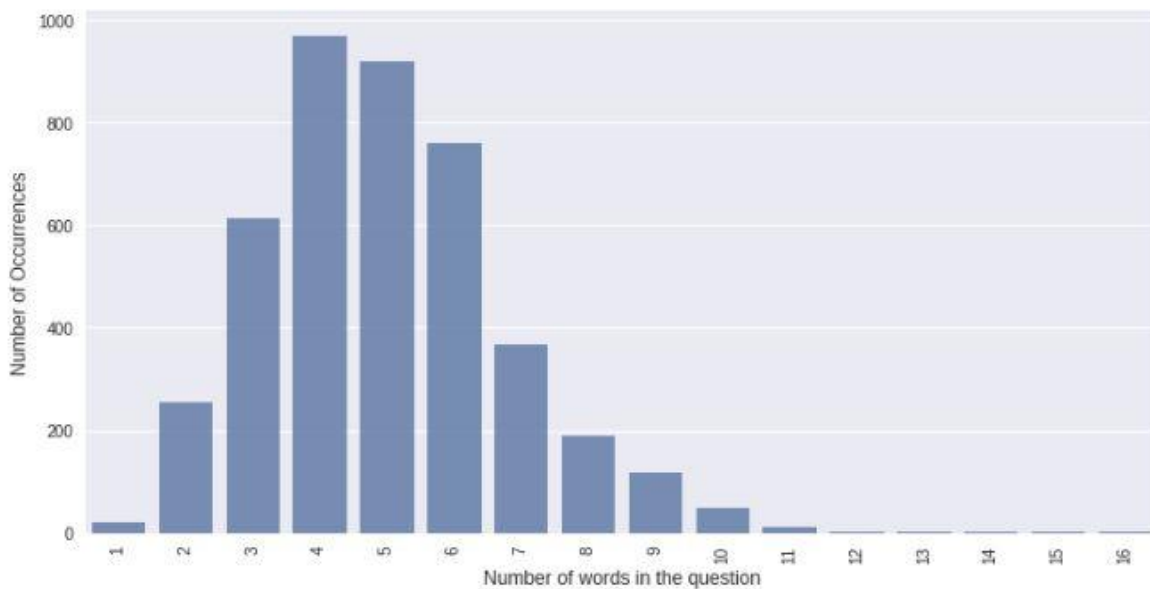
```
sns.barplot(count_str.index, count_str.values, alpha=0.8, color=color[0])
```

```
plt.ylabel('Number of Occurrences', fontsize=12)
```

```
plt.xlabel('Number of words in the question', fontsize=12)
```

```
plt.xticks(rotation='vertical')
```

```
plt.show()
```



In [14]:

```
all ques_df["num_of_chars"] = all ques_df["questions"].apply(lambda x: len(str(x)))
```

```
count_str = all ques_df["num_of_chars"].value_counts()
```

```
plt.figure(figsize=(12,6))
```

```
sns.barplot(count_str.index, count_str.values, alpha=0.8, color=color[3])
```

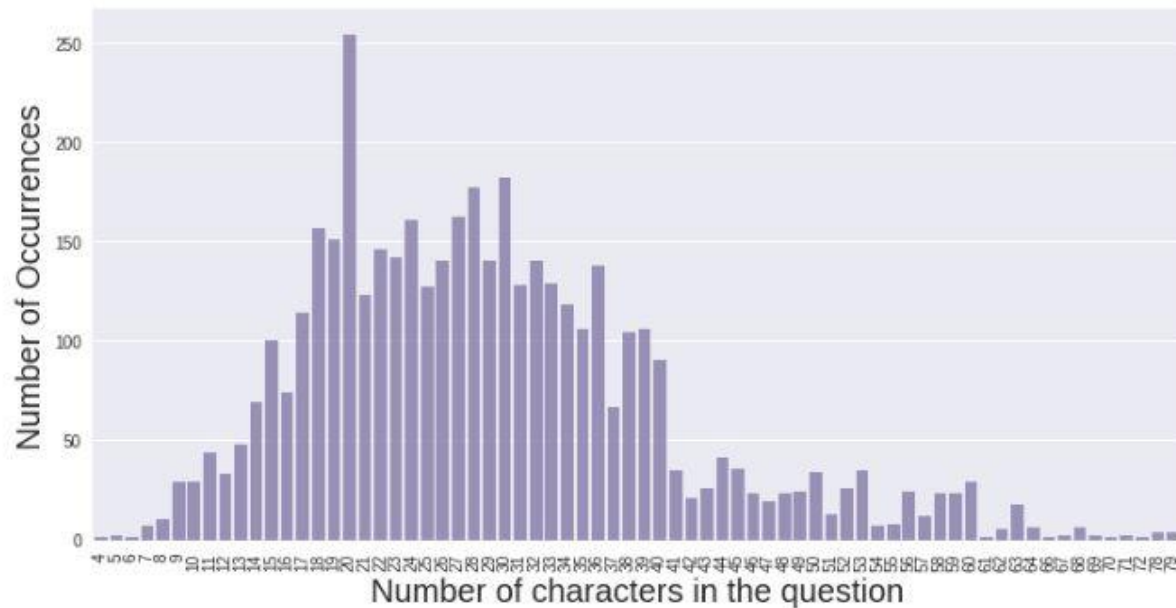
```
plt.ylabel('Number of Occurrences', fontsize=20)
```

```
plt.xlabel('Number of characters in the question', fontsize=20)
```

```
plt.xticks(rotation='vertical')
```

```
plt.show()
```

```
# del all ques_df
```



In [15]:

```
def get_unigrams(que):
    return [word for word in word_tokenize(que.lower()) if word not in eng_stopwords]

## Finding the intersection between two series in pandas and return Len.
def get_common_unigrams(row):
    return len( set(row["unigrams_ques1"]).intersection(set(row["unigrams_ques2"])) )

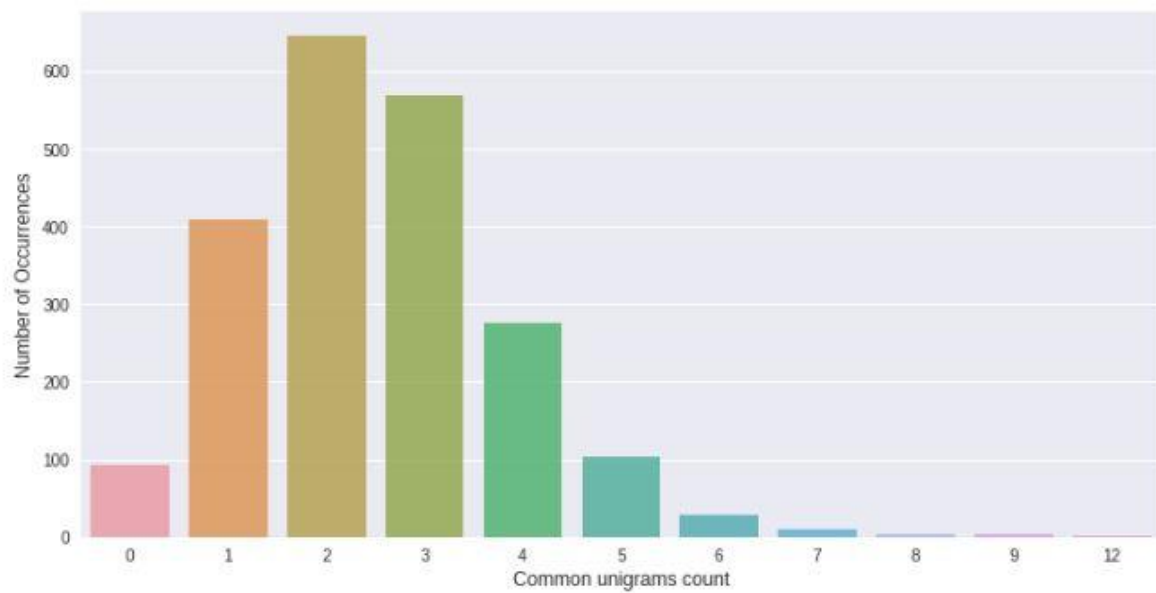
def get_common_unigram_ratio(row):
    return float(row["unigrams_common_count"]) / max(len(
set(row["unigrams_ques1"]).union(set(row["unigrams_ques2"])) ),1)

train_df["unigrams_ques1"] = train_df['question1'].apply(lambda x: get_unigrams(str(x)))
train_df["unigrams_ques2"] = train_df['question2'].apply(lambda x: get_unigrams(str(x)))
train_df["unigrams_common_count"] = train_df.apply(lambda row: get_common_unigrams(row), axis=1)
train_df["unigrams_common_ratio"] = train_df.apply(lambda row:
get_common_unigram_ratio(row),axis=1)
```

In [16]:

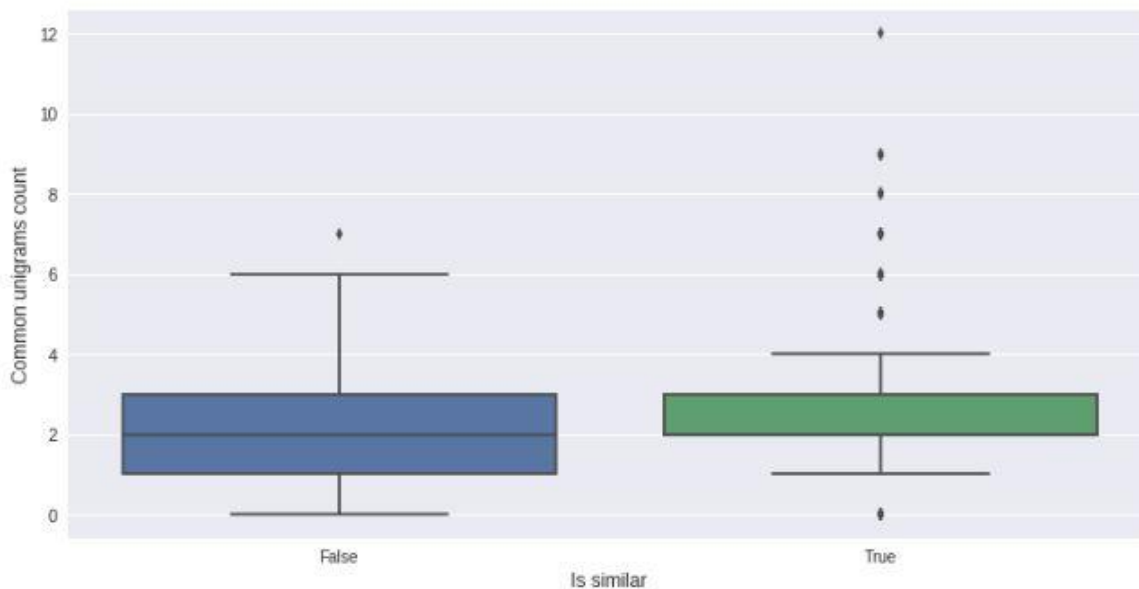
```
count_str = train_df["unigrams_common_count"].value_counts()

plt.figure(figsize=(12,6))
sns.barplot(count_str.index, count_str.values, alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xlabel('Common unigrams count', fontsize=12)
plt.show()
```

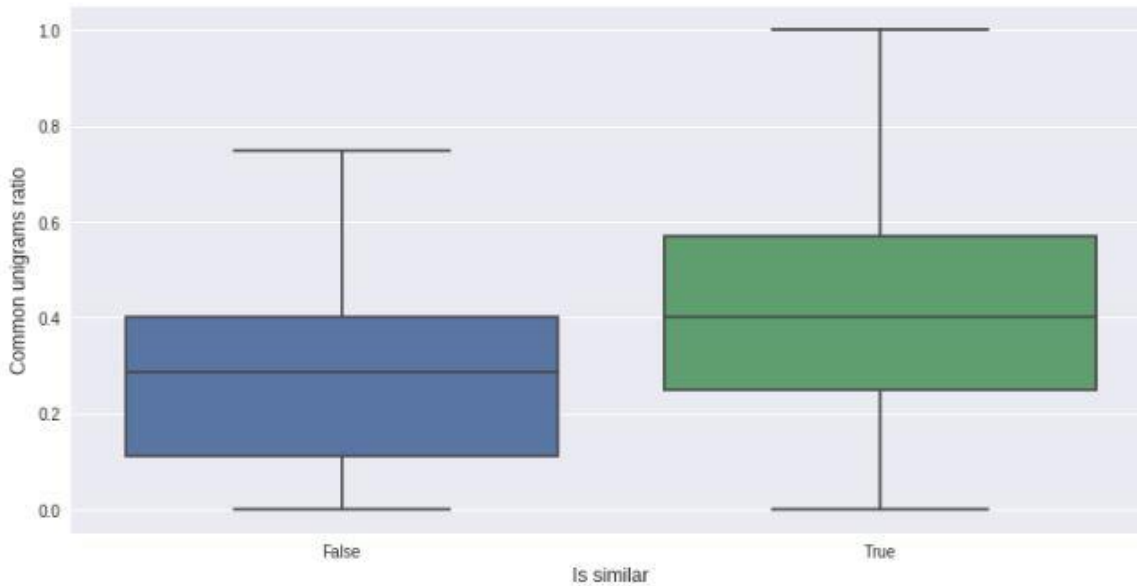
In [17]:

```
plt.figure(figsize=(12,6))
sns.boxplot(x="is_similar", y="unigrams_common_count", data=train_df)
plt.xlabel('Is similar', fontsize=12)
plt.ylabel('Common unigrams count', fontsize=12)
plt.show()
```



In [18]:

```
plt.figure(figsize=(12,6))
sns.boxplot(x="is_similar", y="unigrams_common_ratio", data=train_df)
plt.xlabel('Is similar', fontsize=12)
plt.ylabel('Common unigrams ratio', fontsize=12)
plt.show()
```

In [19]:

```
def get_bigrams(que):
    return [ i for i in ngrams(que,2)]

def get_common_bigrams(row):
    return len( set(row['bigrams_ques1']).intersection(set(row['bigrams_ques2'])) ) )

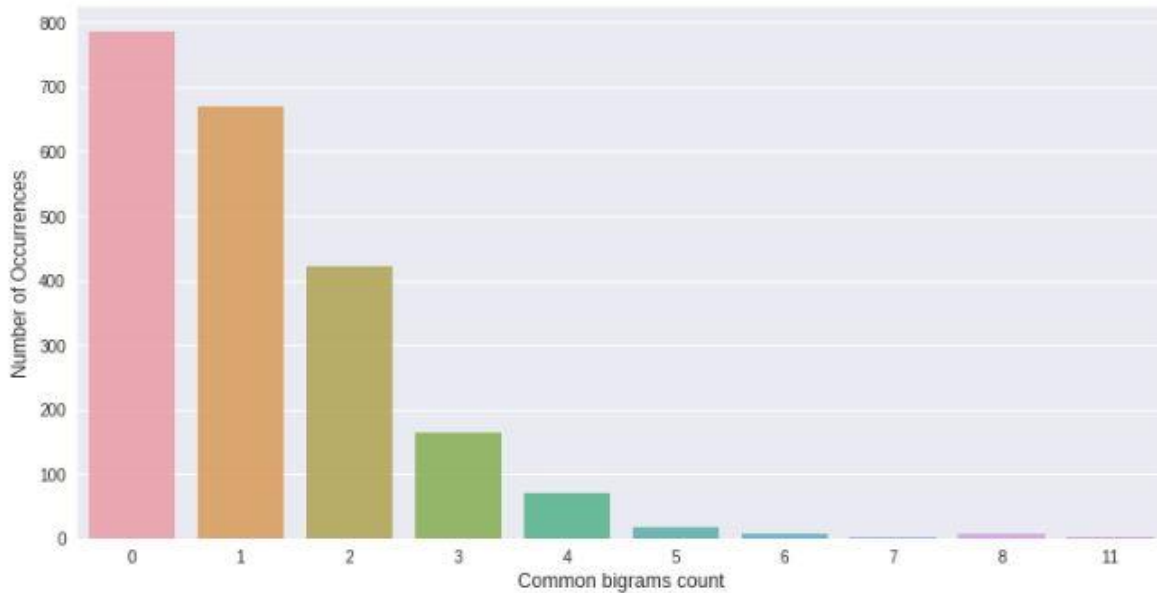
def get_common_bigram_ratio(row):
    return float(row["bigrams_common_count"]) / max(len(
set(row["bigrams_ques1"]).union(set(row["bigrams_ques2"])) ),1)

train_df["bigrams_ques1"] = train_df["unigrams_ques1"].apply(lambda x: get_bigrams(x))
train_df["bigrams_ques2"] = train_df["unigrams_ques2"].apply(lambda x: get_bigrams(x))
train_df["bigrams_common_count"] = train_df.apply(lambda row: get_common_bigrams(row), axis=1)
train_df["bigrams_common_ratio"] = train_df.apply(lambda row: get_common_bigram_ratio(row), axis=1)
```

In [20]:

```
count_str = train_df['bigrams_common_count'].value_counts()

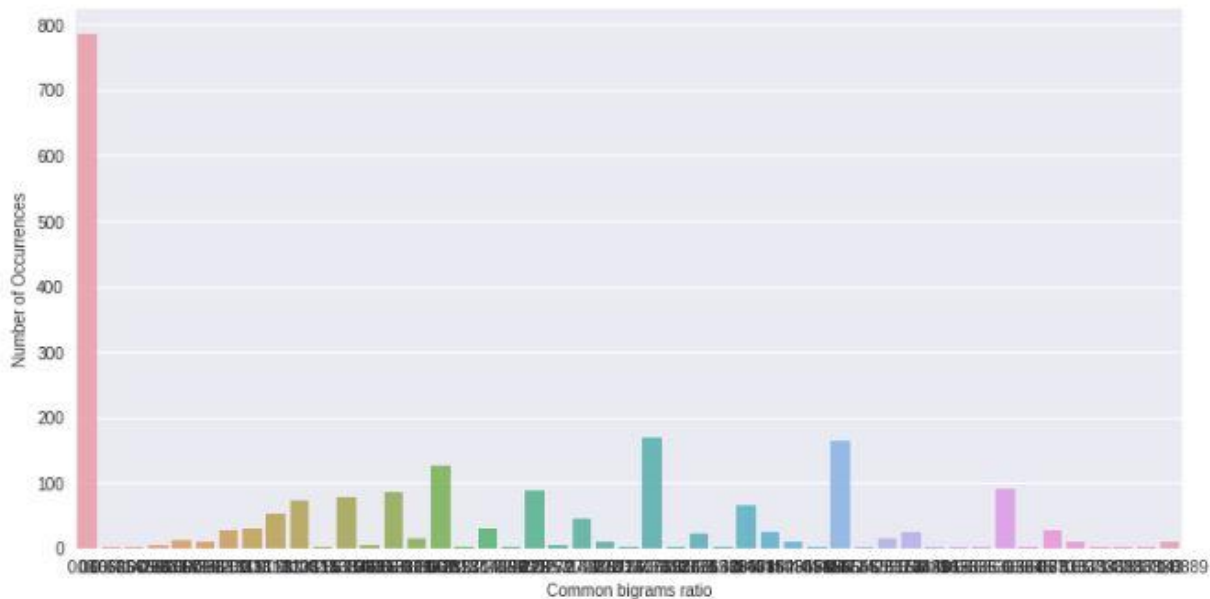
plt.figure(figsize=(12,6))
sns.barplot(count_str.index, count_str.values, alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xlabel('Common bigrams count', fontsize=12)
plt.show()
```



In [21]:

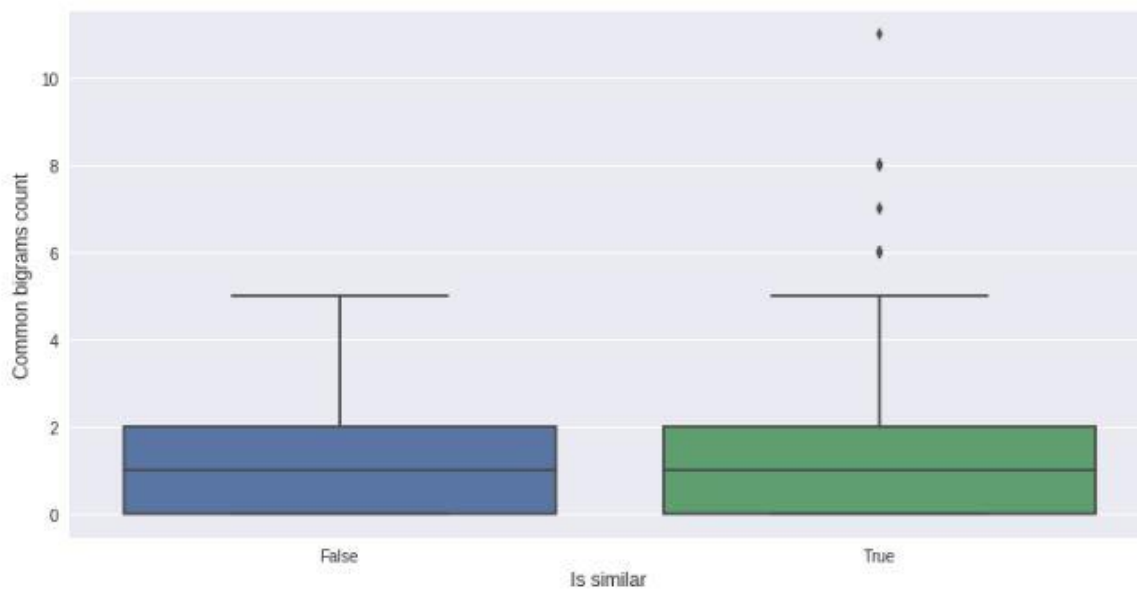
```
count_str = train_df['bigrams_common_ratio'].value_counts()
```

```
plt.figure(figsize=(12,6))
sns.barplot(count_str.index, count_str.values, alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=10)
plt.xlabel('Common bigrams ratio', fontsize=10)
plt.show()
```



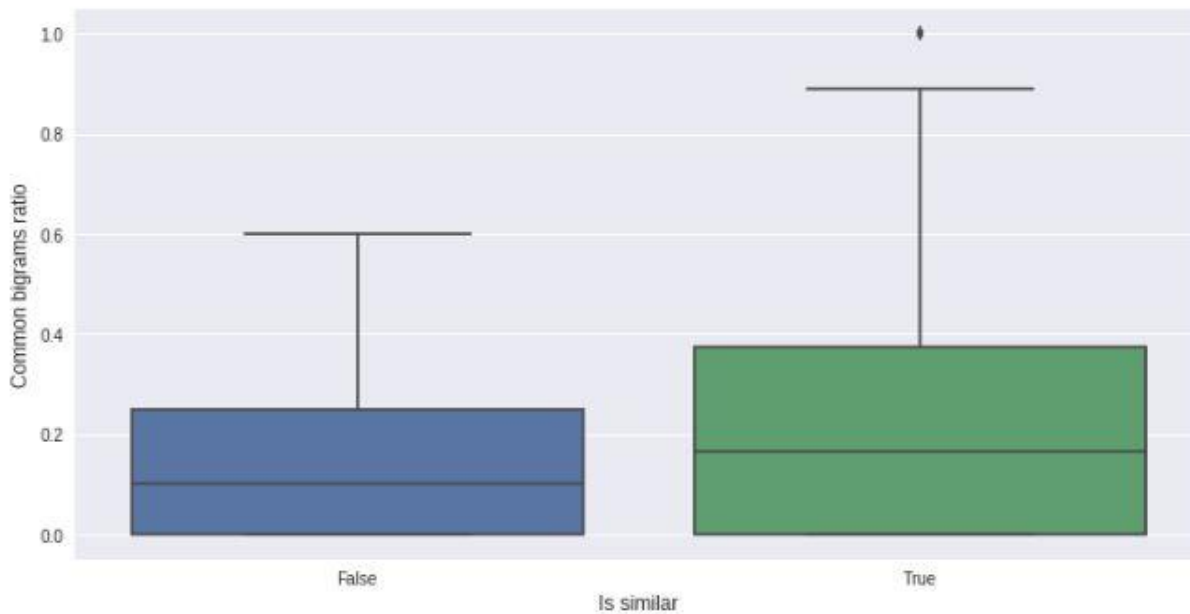
In [22]:

```
plt.figure(figsize=(12,6))
sns.boxplot(x="is_similar", y="bigrams_common_count", data=train_df)
plt.xlabel('Is similar', fontsize=12)
plt.ylabel('Common bigrams count', fontsize=12)
plt.show()
```



In [23]:

```
plt.figure(figsize=(12,6))
sns.boxplot(x="is_similar", y="bigrams_common_ratio", data=train_df)
plt.xlabel('Is similar', fontsize=12)
plt.ylabel('Common bigrams ratio', fontsize=12)
plt.show()
```



In [24]:

```
def feature_extraction(row):
    que1 = str(row['question1'])
    que2 = str(row['question2'])
    out_list = []
    # get unigram features #
    unigrams_que1 = [word for word in que1.lower().split() if word not in eng_stopwords]
    unigrams_que2 = [word for word in que2.lower().split() if word not in eng_stopwords]
    common_unigrams_len = len(set(unigrams_que1).intersection(set(unigrams_que2)))
    common_unigrams_ratio = float(common_unigrams_len) /
    max(len(set(unigrams_que1).union(set(unigrams_que2))),1)
    out_list.extend([common_unigrams_len, common_unigrams_ratio])
```

```

# get bigram features #
bigrams_que1 = [i for i in ngrams(unigrams_que1, 2)]
bigrams_que2 = [i for i in ngrams(unigrams_que2, 2)]
common_bigrams_len = len(set(bigrams_que1).intersection(set(bigrams_que2)))
common_bigrams_ratio = float(common_bigrams_len) /
max(len(set(bigrams_que1).union(set(bigrams_que2))),1)
out_list.extend([common_bigrams_len, common_bigrams_ratio])

# get trigram features #
trigrams_que1 = [i for i in ngrams(unigrams_que1, 3)]
trigrams_que2 = [i for i in ngrams(unigrams_que2, 3)]
common_trigrams_len = len(set(trigrams_que1).intersection(set(trigrams_que2)))
common_trigrams_ratio = float(common_trigrams_len) /
max(len(set(trigrams_que1).union(set(trigrams_que2))),1)
out_list.extend([common_trigrams_len, common_trigrams_ratio])
return out_list

```

In [25]:

```

def runXGB(train_X, train_y, test_X, test_y=None, feature_names=None, seed_val=0):
    params = {}
    params["objective"] = "binary:logistic"
    params['eval_metric'] = 'logloss'
    params["eta"] = 0.02
    params["subsample"] = 0.7
    params["min_child_weight"] = 1
    params["colsample_bytree"] = 0.7
    params["max_depth"] = 4
    params["silent"] = 1
    params["seed"] = seed_val
    num_rounds = 300
    plst = list(params.items())
    xgtrain = xgb.DMatrix(train_X, label=train_y)

    if test_y is not None:
        xgtest = xgb.DMatrix(test_X, label=test_y)
        watchlist = [ (xgtrain, 'train'), (xgtest, 'test') ]
        model = xgb.train(plst, xgtrain, num_rounds, watchlist, early_stopping_rounds=100,
verbose_eval=10)
    else:
        xgtest = xgb.DMatrix(test_X)
        model = xgb.train(plst, xgtrain, num_rounds)

    pred_test_y = model.predict(xgtest)

    loss = 1
    if test_y is not None:
        loss = log_loss(test_y, pred_test_y)
        return pred_test_y, loss, model
    else:
        return pred_test_y, loss, model

```

In [26]:

```

train_X = np.vstack( np.array(train_df.apply(lambda row: feature_extraction(row), axis=1)) )
test_X = np.vstack( np.array(test_df.apply(lambda row: feature_extraction(row), axis=1)) )
train_y = np.array(train_df["is_similar"])
test_id = np.array(test_df["test_id"])

```

/opt/conda/lib/python3.6/site-packages/ipykernel/__main__.py:21:

DeprecationWarning: generator 'ngrams' raised StopIteration

/opt/conda/lib/python3.6/site-packages/ipykernel/__main__.py:20:

DeprecationWarning: generator 'ngrams' raised StopIteration

In [27]:

```
train_X_similar = train_X[train_y==1]
train_X_non_similar = train_X[train_y==0]

train_X = np.vstack([train_X_non_similar, train_X_similar, train_X_non_similar,
train_X_non_similar])
train_y = np.array([0]*train_X_non_similar.shape[0] + [1]*train_X_similar.shape[0] +
[0]*train_X_non_similar.shape[0] + [0]*train_X_non_similar.shape[0])
del train_X_similar
del train_X_non_similar
print("Mean target rate : ",train_y.mean())
```

Mean target rate : 0.5040625

In [28]:

```
kf = KFold(n_splits=20, shuffle=True, random_state=2016)
for dev_index, val_index in kf.split(range(train_X.shape[0])):
    dev_X, val_X = train_X[dev_index:], train_X[val_index:]
    dev_y, val_y = train_y[dev_index], train_y[val_index]
    preds, lloss, model = runXGB(dev_X, dev_y, val_X, val_y)
    break
```

[0] train-logloss:0.689675 test-logloss:0.690481

Multiple eval metrics have been passed: 'test-logloss' will be used for early stopping.

Will train until test-logloss hasn't improved in 100 rounds.

```
[10] train-logloss:0.662903 test-logloss:0.668619
[20] train-logloss:0.642654 test-logloss:0.652351
[30] train-logloss:0.630103 test-logloss:0.642319
[40] train-logloss:0.619505 test-logloss:0.633653
[50] train-logloss:0.60543 test-logloss:0.622782
[60] train-logloss:0.594623 test-logloss:0.615234
[70] train-logloss:0.586651 test-logloss:0.609191
[80] train-logloss:0.579887 test-logloss:0.605302
[90] train-logloss:0.573774 test-logloss:0.601843
[100] train-logloss:0.568991 test-logloss:0.59923
[110] train-logloss:0.564178 test-logloss:0.595606
[120] train-logloss:0.559203 test-logloss:0.593106
[130] train-logloss:0.555807 test-logloss:0.591623
[140] train-logloss:0.552882 test-logloss:0.590005
[150] train-logloss:0.549983 test-logloss:0.588719
[160] train-logloss:0.547705 test-logloss:0.587347
[170] train-logloss:0.545736 test-logloss:0.586605
[180] train-logloss:0.543875 test-logloss:0.585491
[190] train-logloss:0.541798 test-logloss:0.584386
[200] train-logloss:0.539833 test-logloss:0.583067
[210] train-logloss:0.538559 test-logloss:0.582426
[220] train-logloss:0.537128 test-logloss:0.582368
[230] train-logloss:0.535771 test-logloss:0.581501
[240] train-logloss:0.534428 test-logloss:0.581033
[250] train-logloss:0.533441 test-logloss:0.580278
[260] train-logloss:0.532361 test-logloss:0.579552
[270] train-logloss:0.531435 test-logloss:0.578678
```

```
[280] train-logloss:0.530075    test-logloss:0.577884
[290] train-logloss:0.529124    test-logloss:0.577756
[299] train-logloss:0.528408    test-logloss:0.577879
```

In [29]:

```
xgtest = xgb.DMatrix(test_X)
preds = model.predict(xgtest)
```

```
out_df = pd.DataFrame({"test_id":test_id, "is_similar":preds})
out_df.to_csv("issimilar_predicted.csv", index=False)
```

INDIAVIX

India VIX is a volatility index based on the NIFTY Index Option prices. Volatility Index is a measure of market's expectation of volatility over the near term. Volatility is often described as the "rate and magnitude of changes in prices" and in finance often referred to as risk. Volatility Index is a measure, of the amount by which an underlying Index is expected to fluctuate, in the near term.

V. CONCLUSION:

The study is based on the analysis of small cap stocks listed on BSE. This study helps the investor in identifying the performance of selected small cap stocks for a given period. It can be concluded that stock with higher beta value is not preferred as it is exposed to higher market risk.

All the small cap stocks has higher standard deviation, therefore it is advised to construct a portfolio to avoid higher risk. This study is an attempt to evaluate the performance of selected small cap stocks listed on BSE and to identify the best stock to invest and the worst stock to be ignored. If the investors are ready to take higher risk, then the investors are suggested to invest in Aarti drugs and RSS software, where the returns are highest but risk is also high. The investors who are looking for moderate returns are suggested to invest in Tata Elxsi and Wintac pharmaceuticals. The investors who are looking for low risk are suggested to invest in DCB bank. However by constructing a portfolio an investor can minimise the risk.

REFERENCE:

1. Suresh A.S and Harshitha N, 2017, "Comparison of Returns and Risk.1 using Markowitz and Sharpe's Model", International Journal of Management and Commerce Innovations, Vol. 5, Issue 1, pp: 806- 813.
2. Markowitz, H. M. (1959). "The History of Portfolio Theory : Portfolio Theory : 1600 - 1960". Financial Analysts Journal, 55(4), pp 5-16.
3. J. F Affleck - Graves and A. H Money. (1976). "A comparison of two portfolio selection models". The Investment Analysts Journal, 7(4), pp 35-40. Suresh A.S, 2018, Equity Analysis of Selected Logistics Sectors.

4 Stocks, Asian Journal of Management, Volume-9, Issue-4, pp: 1301- 1304.

5. RO. Michaud. (1989). "The Markowitz optimization enigma: Is 'optimized' optimal?" Financial Analysts Journal, 45(1), pp 31-42.

6. W. F. Sharpe. (1963). "A Simplified Model for Portfolio Analysis". Management Science, 9(2), pp 277-293 Suresh A.S, 2018, Study on Comparison of Risk-Return Analysis of.

7 Public and Private Sector Banks listed on Bank Nifty, Journal of Business Management and Economic Research, Vol.2, Issue.1, 2018, pp.1-8