Problem Statement

The first part of the project worked on included implementing of the research papers by Khaidem Luckyson, Snehanshu Saha, and Sudeepa Roy Dey on "Predicting the direction of stock market prices using random forest." arXiv preprint arXiv:1605.00003 (2016).

We implemented the paper on stocks in same sector in Indian Stock markets.

In the later stages of the project after implementing the paper, we implemented several other variations of the model to improve the accuracy of our predictions of the stock prices.

What is Stock Market?

The stock market refers to public markets that exist for issuing, buying, and selling stocks that trade on a stock exchange. Stocks, also known as equities, represent fractional ownership in a company, and the stock market is a place where investors can buy and sell ownership of such investable assets.

Our motivation for this project:

Stock market is very volatile and various strategies are being used to predict the stock prices in the future by many analysts. Predicting the trends in stock market prices is a very challenging task due to the many uncertainties involved and many variables that influence the market value in a particular day. Correct prediction of stock prices has great intellectual and monetary value

We decided to use Machine Learning for the prediction of the stock prices to get obtain better accuracy compared to other analysts.

Paper Interpretation

- In this paper the authors preprocessed the data for stocks by:
  1. Dropping the null values in the data set which were obtained from the source of data.
  2. Smoothening the dataset by giving more weight to recent data than the past data
- The authors used various however, a limited list of technical indicators for feature extraction in order to build the feature space for training purposes.
- A random forest was constructed to predict the stock price direction in the future
- The results obtained by the authors were quite good

Additions done by us

1. We used more technical indicators available to us to extend our feature space to increase the accuracy
2. We incorporated stocks of the same sector (Pharmaceutical Industry's Equity on the Indian Market) in feature set to observe correlation in the results obtained and identify the general trend of the sector
3. We implemented an SVM model as well to improve the accuracy and compare capabilities of both the Random Forest model and the Support Vector Machines model

Methodology and Analysis

Data Pre-Processing

Exponential smoothing : It is applied on time series stock data, which applies more weightage to the recent observation and exponentially decreasing weights to past observations. The exponentially smoothed statistic of a series Y can be recursively calculated as:

$S_0 = Y_0$
for $t > 0$
$S_t = \alpha * Y_t + (1 - \alpha) * S_{t-1}$

where α is the smoothing factor and $0 < \alpha < 1$. Larger values of α reduce the level of smoothing. When $\alpha = 1$, the smoothed statistic becomes equal to the actual observation.

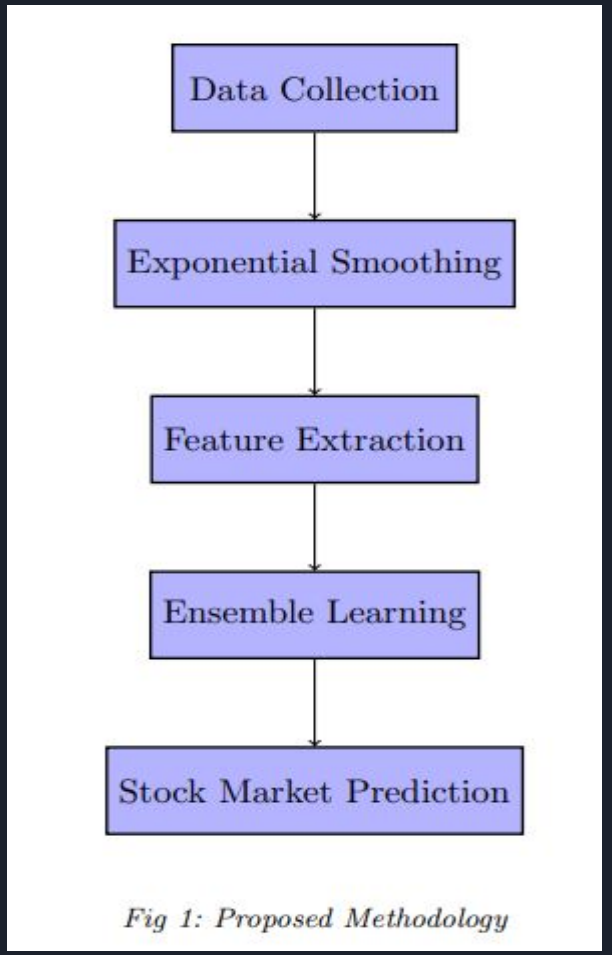

Fig 1: Proposed Methodology

Snippet of the Code for PreProcessing of Data

Exponential Smoothing

As indicated by the authors of the paper, this is done for the purpose of putting more importance on recent data and exponentially decreasing weightage to past data

In [10]:
```
# Function for exponentially smoothing

def exp_smoothing(df, alpha):
    es_data = df.ewm(alpha=alpha).mean()
    return es_data
```

In [32]:
```
# For current testing purposes, value of alpha used is 0.9

sdata = exp_smoothing(raw_data, 0.9)
sdata2 = exp_smoothing(raw_data2, 0.9)
sdata3 = exp_smoothing(raw_data3, 0.9)
sdata4 = exp_smoothing(raw_data4, 0.9)
sdata5 = exp_smoothing(raw_data5, 0.9)
sdatan = exp_smoothing(raw_datan, 0.9)
```

In [8]:
```
data = prepare_data(sdata, 30)

## Identifying and extracting the label
y = data['pred']

## Extracting the input features and creating the input feature matrix
input_feature = [x for x in data.columns if x not in ['gain', 'pred']]
X = data[input_feature]
data
```

Methodology and Analysis

Data Processing

Technical Indicators :

1. These are important parameters that are calculated from time series stock data that aim to forecast financial market direction. They are tools which are widely used by investors to check for bearish or bullish signals.
2. Technical indicators are calculated from the exponentially smoothed time series data which are later organized into feature matrix. The target to be predicted in the i th day is calculated as follows:

$$target_i = sign(close_{i+d} - close_i)$$

where d is the number of days after which the prediction is to be made. When the value of target-i is +1, it indicates that there is a positive shift in the price after d days and -1 indicates that there is a negative shift after d days. The target-i values are assigned as labels to the i th row in the feature matrix.

Methodology and Analysis

Feature Extraction

The technical indicators which we took **from Research paper** are listed below :

Name of Technical Indicator	Name used in Code Snippet
Relative Strength Index	relative_strength_index
Stochastic Oscillator	stochastic_oscillator_d
Moving Average Convergence Divergence (MACD)	macd
Price Rate of Change	rate_of_change
On Balance Volume	on_balance_volume

Methodology and Analysis

Feature Extraction

Extra technical indicators which we used in our project are listed below :

Name of Technical Indicator	Name used in Code Snippet
Moving average	moving_average
Standard deviation	standard_deviation
Donchian channel	donchian_channel
Ultimate oscillator	ultimate_oscillator
Keltner channel	keltner_channel
Coppock momentum indicator	coppock_curve

Methodology and Analysis

Feature Extraction

Extra technical indicators which we used in our project are listed below :

Name of Technical Indicator	Name used in Code Snippet
Commodity channel index	commodity_channel_index
Ease of movement	ease_of_movement
Force index	force_index
Average true range	average_true_range
Money flow index	money_flow_index
Chaikin oscillator	chaikin_oscillator
Exponential moving average	exponential_moving_average

Methodology and Analysis

Feature Extraction

Extra technical indicators which we used in our project are listed below :

Name of Technical Indicator	Name used in Code Snippet
Accumulation distribution	accumulation_distribution
True strength index	true_strength_index
Relative strength index	frelative_strength_index
Vortex indicator	vortex_indicator
Know sure thing oscillator	kst_oscillator
Mass index	mass_index

Methodology and Analysis

Feature Extraction

Extra technical indicators which we used in our project are listed below :

Name of Technical Indicator	Name used in Code Snippet
Stochastic oscillator d	stochastic_oscillator_d
Stochastic oscillator k	stochastic_oscillator_k
Personal Property Securities Register	ppsr
Bollinger bands	bollinger_bands
Average directional movement index	average_directional_movement_index
The triple exponential average	trix

Snippet of the Code for Technical Indicators

Technical indicators used for Feature Extraction

In [33]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_new = scaler.fit_transform(X)

print("This is X_new: ")
print(X_new)
```

In [12]:
```
## LIST WILL BE EDITED ACCORDING TO SUREKA'S LIST
### NO CHANGE HERE AS OF NOW


def feature_extraction(data):
    for x in [5, 14, 26, 44, 66]:
        data = ta.relative_strength_index(data, n=x)
        data = ta.stochastic_oscillator_d(data, n=x)
        data = ta.accumulation_distribution(data, n=x)
        data = ta.average_true_range(data, n=x)
        data = ta.momentum(data, n=x)
        data = ta.money_flow_index(data, n=x)
        data = ta.rate_of_change(data, n=x)
        data = ta.on_balance_volume(data, n=x)
        data = ta.commodity_channel_index(data, n=x)
        data = ta.ease_of_movement(data, n=x)
        data = ta.trix(data, n=x)
        data = ta.vortex_indicator(data, n=x)
        #add %R indicator

    data['ema50'] = data['Close'] / data['Close'].ewm(50).mean()
    data['ema21'] = data['Close'] / data['Close'].ewm(21).mean()
    data['ema14'] = data['Close'] / data['Close'].ewm(14).mean()
    data['ema5'] = data['Close'] / data['Close'].ewm(5).mean()

    #williams %R is missing
    data = ta.macd(data, n_fast=12, n_slow=26)

    del(data['Open'])
    del(data['Prev Close'])
    del(data['High'])
    del(data['Low'])
    del(data['Volume'])
    del(data['Last'])
    del(data['Average'])
    del(data['Turnover'])
    del(data['No. of Trades'])
    del(data['Deliverable Qty'])
    del(data['% Dly Qt to Traded Qty'])

    return data
```

Under the function ‘feature_extraction’ and within the for loop, we have several technical indicators included from the library.

This portion of the code is for implementing the research paper.

Methodology and Analysis

Learning algorithms

1) Random forest
    a) Works by training **multiple decision trees** on **different subspaces** of the feature space, which overcomes the problem of overfitting of training data, at the cost of slightly increased bias.
    b) **None of the trees in the forest sees the entire training data**. The data is **recursively split** into various partitions. At a particular node, the split is done by asking a question on an attribute.
    c) The choice for the splitting criterion is based on some **impurity measures** such as Shannon Entropy or Gini impurity. Gini impurity is used as the function to measure the quality of split in each node.

Snippet of the Code for Random Forest

MODEL - Random Forests

In [17]:
```
model = RandomForestClassifier(n_jobs=-1, n_estimators=80, random_state=42)
```

Training and testing the model

In [18]:
```
model.fit(X_train, y_train.values.ravel());
print("model fitted")
prediction = model.predict(X_test)
print("prediction created")
print(np.size(prediction))
print(np.size(y_test))

accuracy = accuracy_score(y_pred=prediction, y_true=y_test)

print('Accuracy: {0:1.2f}'.format(accuracy))


confusion = confusion_matrix(y_pred=prediction, y_true=y_test)
print('Confusion Matrix')
print(confusion)


precision = precision_score(y_pred=prediction, y_true=y_test)
recall = recall_score(y_pred=prediction, y_true=y_test)
f1 = f1_score(y_pred=prediction, y_true=y_test)
print('Precision: {0:1.2f}, Recall: {1:1.2f}, f1: {2:1.2f}'.format(precision, recall, f1))
```

```
model fitted
prediction created
188
188
Accuracy: 0.63
Confusion Matrix
[[80 23]
 [46 39]]
Precision: 0.63, Recall: 0.46, f1: 0.53
```

Methodology and Analysis

Learning algorithms

2) Support Vector Machine

A support vector machine is a **supervised learning algorithm** that sorts data into two categories. The objective is to find a **hyperplane in an N-dimensional space** (N — number of features) that distinctly classifies the data points.

The algorithm must find a plane that has the **maximum distance between data points of both classes**, so that future data points are classified with greatest confidence.

Support vectors are data points that are **closest to the hyperplane** and influence the position and orientation of the hyperplane, and help maximising the margin

Snippet of the Code for Support Vector Machines

SVM IMPLEMENTATION

In [19]:

```
from sklearn import svm
# model = svm.SVC(gamma = 1 / (X_train.shape[-1] * X_train.var()))

model = svm.SVC()

model.fit(X_train, y_train.values.ravel());
print("model fitted")
prediction = model.predict(X_test)
print("prediction created")
print(np.size(prediction))
print(np.size(y_test))

accuracy = accuracy_score(y_pred=prediction, y_true=y_test)

print('Accuracy: {0:1.2f}'.format(accuracy))


confusion = confusion_matrix(y_pred=prediction, y_true=y_test)
print('Confusion Matrix')
print(confusion)


precision = precision_score(y_pred=prediction, y_true=y_test)
recall = recall_score(y_pred=prediction, y_true=y_test)
f1 = f1_score(y_pred=prediction, y_true=y_test)
print('Precision: {0:1.2f}, Recall: {1:1.2f}, f1: {2:1.2f}'.format(precision, recall, f1))
```

```
model fitted
prediction created
188
188
Accuracy: 0.59
Confusion Matrix
[[94  9]
 [68 17]]
Precision: 0.65, Recall: 0.20, f1: 0.31
```

Time series data

As the data was time series we couldn't do random splitting for test and train data as this causes data leakage.

Creating random split data improves the accuracy by a wide margin but the method is incorrect as there is overlap of data between training and testing data.

In practice the accuracy from TimeSeriesSplit data split is the correct estimate of accuracy compared to random split which is infeasible and overestimates the accuracy.

Out-of-bag (OOB) error

Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests utilizing bootstrap aggregating (bagging) to subsample data samples used for training. OOB is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

The Random Forest Classifier is trained using bootstrap aggregation, where each new tree is fit from a bootstrap sample of the training observations . The out-of-bag (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained.

Results

Implementation of Research Paper:

Through the direct implementation of the research paper, the following results were obtained:

1. Accuracy = 56%
2. Precision = 58%
3. Confusion matrix:

```
Confusion Matrix
[[55 46]
 [33 45]]
```

Results

Implementation of Research Paper:

Results snippet from running the code

```
Accuracy: 0.56
Confusion Matrix
[[55 46]
 [33 45]]
Precision: 0.55, Recall: 0.58, f1: 0.50
```

Results

Implementation of extension of Research Paper:

Through the direct implementation of additional features, the following results were obtained:

1. Accuracy = 63%
2. Precision = 67%
3. Confusion matrix:

```
Confusion Matrix
[[96  7]
 [62 14]]
```

Results

Implementation of extension of Research Paper:

Results snippet from running the code

```
Accuracy: 0.63
Confusion Matrix
[[96  7]
 [62 14]]
Precision: 0.67, Recall: 0.16, f1: 0.26
```

Conclusions (1)

Direct implementation of the research paper suggests that the most important features are:

	feature	importance
102	Trix_66	0.026186
76	OBV_44	0.019738
79	Trix_44	0.018579
112	BollingerB_66	0.017099
87	Copp_44	0.016733

We see that these features contribute most to the determination of direction of movement of price.

Conclusions (2)

Implementation of additional features and research paper suggests that the most important features are:

	feature	importance
306	Acc/Dist_ROC_44_5	0.007249
81	Acc/Dist_ROC_14_2	0.007183
26	Acc/Dist_ROC_26	0.006673
117	Acc/Dist_ROC_66_2	0.005808
227	Acc/Dist_ROC_26_4	0.005690
93	Acc/Dist_ROC_26_2	0.005663
171	SO%d_44_3	0.005577
274	ROC_5_5	0.005467
318	Acc/Dist_ROC_66_5	0.005403
276	CCI_5_5	0.005383
14	Acc/Dist_ROC_14	0.005358
50	Acc/Dist_ROC_66	0.005349

We also see that there is a considerable improvement in the results obtained from implementation of additional features and from including the equity market of related companies.

Here, we observe that there is a strong dependence on the stock prices of other companies highlighted through the “_n” where n is a particular company number.

Additional notes

Through refinement of parameters and alteration of horizon, the maximum possible accuracy consistently achieved is for

- Alpha = 0.9
- Horizon = 30 days

This means that the data has been smoothed with a constant 0.9

We have been most successful in predicting the stock price after 30 days, i.e. medium-term.

Thank You!