1

1

Goal of the project

Reproduce the results from the paper "Predicting the direction of stock market prices using random forest."

Import Libraries

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (7,4.5) # Make the default figures a
bit bigger

import numpy as np
import random

#Let's make this notebook reproducible
np.random.seed(42)
random.seed(42)

import pandas_techinal_indicators as ta #https://github.com/Crypto-
toolbox/pandas-technical-indicators/blob/master/
technical_indicators.py

import pandas as pd
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import f1_score, precision_score,
confusion_matrix, recall_score, accuracy_score
from sklearn.model_selection import train_test_split
```

Data

In order to reproduce the same results as the authors, we try to use the same data they used in page 18, table 15.

Since the authors state that they compare these results with other authors using the data from yahoo finance in the period [2010-01-04 to 2014-12-10] we will use the same data. It is not clear which periods they used as traning set and testing set.

```
aapl = pd.read_csv('AAPL.csv')
del(aapl['Date'])
del(aapl['Adj Close'])
aapl.head()
```

	Open	High	Low	Close	Volume
0	33.641430	33.801430	33.494286	33.571430	107664900
1	33.915714	34.104286	33.250000	33.709999	150786300
2	33.568573	34.072857	33.538570	34.070000	171126900
3	34.028572	34.320000	33.857143	34.220001	111754300
4	34.221428	34.560001	34.094284	34.371429	157125500

Exponential smoothing

The authors don't give any guideline for alpha, so let's assume it is 0.9

```
def get_exp_preprocessing(df, alpha=0.9):
    edata = df.ewm(alpha=alpha).mean()
    return edata

saapl = get_exp_preprocessing(aapl)
saapl.head() #saapl stands for smoothed aapl
```

	Open	High	Low	Close	Volume
0	33.641430	33.801430	33.494286	33.571430	1.076649e+08
1	33.890779	34.076754	33.272208	33.697402	1.468662e+08
2	33.600503	34.073243	33.512174	34.033076	1.687227e+08
3	33.985804	34.295347	33.822677	34.201325	1.174460e+08
4	34.197868	34.533538	34.067126	34.354420	1.531579e+08

Feature Extraction - Technical Indicators

It's not very clear what 'n' should be in most of the indicators, so, we are using several values of 'n'

The indicators used are taken from Peter Bakker at: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

note: the Williams %R indicator does not seem to be available in this library yet

```
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)

    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['High'])
    del(data['Low'])
    del(data['Volume'])

    return data

def compute_prediction_int(df, n):
    pred = (df.shift(-n)['Close'] >= df['Close'])
    pred = pred.iloc[:-n]
    return pred.astype(int)

def prepare_data(df, horizon):
    data = feature_extraction(df).dropna().iloc[:-horizon]
    data['pred'] = compute_prediction_int(data, n=horizon)
    del(data['Close'])
    return data.dropna()
```

Prepare the data with a prediction horizon of 10 days

```
data = prepare_data(saapl, 10)

y = data['pred']

#remove the output from the input
features = [x for x in data.columns if x not in ['gain', 'pred']]
X = data[features]
```

Make sure that future data is not used by splitting the data in first 2/3 for training and the last 1/3 for testing

```
train_size = 2*len(X) // 3

X_train = X[:train_size]
X_test = X[train_size:]
y_train = y[:train_size]
y_test = y[train_size:]

print('len X_train', len(X_train))
print('len y_train', len(y_train))
print('len X_test', len(X_test))
print('len y_test', len(y_test))
```

```
len X_train 644
len y_train 644
len X_test 323
len y_test 323
```

Random Forests

```
rf = RandomForestClassifier(n_jobs=-1, n_estimators=65,
random_state=42)
rf.fit(X_train, y_train.values.ravel());
```

The expected results for a 10 days prediction according to the paper in table 15 for Apple stock should be around 92%

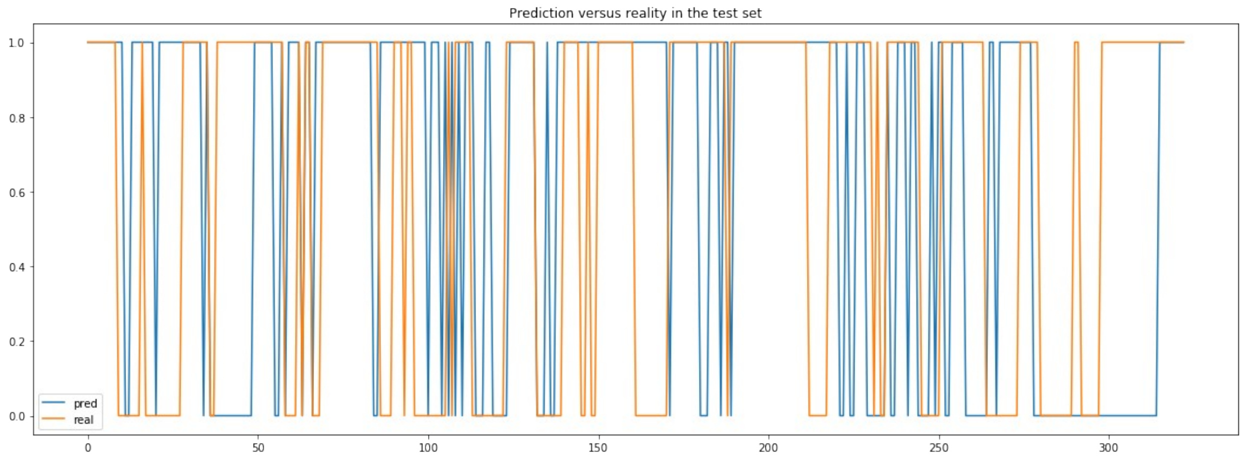
```
pred = rf.predict(X_test)
precision = precision_score(y_pred=pred, y_true=y_test)
recall = recall_score(y_pred=pred, y_true=y_test)
f1 = f1_score(y_pred=pred, y_true=y_test)
accuracy = accuracy_score(y_pred=pred, y_true=y_test)
confusion = confusion_matrix(y_pred=pred, y_true=y_test)
print('precision: {0:1.2f}, recall: {1:1.2f}, f1: {2:1.2f}, accuracy:
{3:1.2f}'.format(precision, recall, f1, accuracy))
print('Confusion Matrix')
print(confusion)
```

```
precision: 0.66, recall: 0.68, f1: 0.67, accuracy: 0.58
Confusion Matrix
[[ 47  71]
 [ 66 139]]
```

However, the resulting accuracy is 58% !

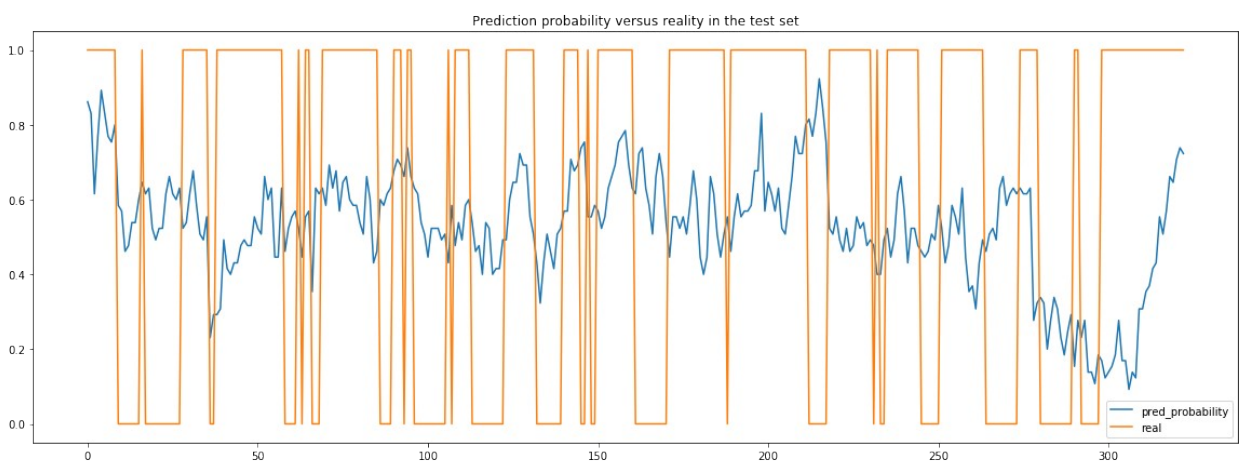
Some plots for intuition of what is going on

```
plt.figure(figsize=(20,7))
plt.plot(np.arange(len(pred)), pred, label='pred')
plt.plot(np.arange(len(y_test)), y_test, label='real' );
plt.title('Prediction versus reality in the test set')
plt.legend();
```




```
plt.figure(figsize=(20,7))
proba = rf.predict_proba(X_test)[:,1]
plt.figure(figsize=(20,7))
plt.plot(np.arange(len(proba)), proba, label='pred_probability')
plt.plot(np.arange(len(y_test)), y_test, label='real' );
plt.title('Prediction probability versus reality in the test set');
plt.legend();
plt.show();

<matplotlib.figure.Figure at 0x1fec5d7a908>
```



Let's now duplicate the analysis for the case where the test set is shuffled

This means that there is **data leakage** in the training set, as the future and the past are together in the training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =
2*len(X) // 3)

print('len X_train', len(X_train))
print('len y_train', len(y_train))
print('len X_test', len(X_test))
print('len y_test', len(y_test))
```

```
len X_train 644
len y_train 644
len X_test 323
len y_test 323
```

Let's use Random Forests with data leaked data set

```
rf = RandomForestClassifier(n_jobs=-1, n_estimators=65,
random_state=42)
rf.fit(X_train, y_train.values.ravel());
```

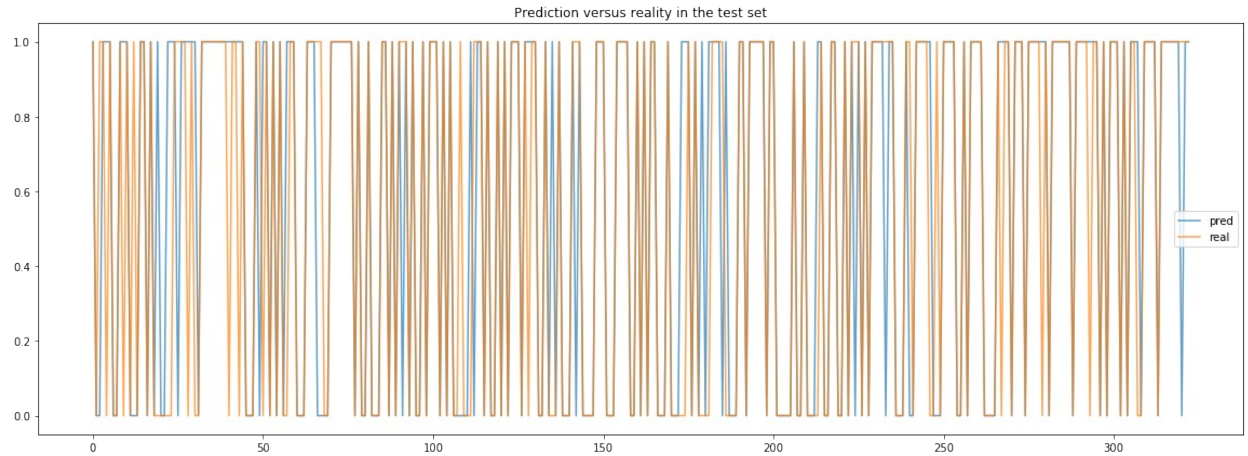
The expected results for a 10 days prediction according to the paper in table 15 for Apple stock should be around 92%

```
pred = rf.predict(X_test)
precision = precision_score(y_pred=pred, y_true=y_test)
recall = recall_score(y_pred=pred, y_true=y_test)
f1 = f1_score(y_pred=pred, y_true=y_test)
accuracy = accuracy_score(y_pred=pred, y_true=y_test)
confusion = confusion_matrix(y_pred=pred, y_true=y_test)
print('precision: {0:1.2f}, recall: {1:1.2f}, f1: {2:1.2f}, accuracy:
{3:1.2f}'.format(precision, recall, f1, accuracy))
print('Confusion Matrix')
print(confusion)
```

```
precision: 0.87, recall: 0.91, f1: 0.89, accuracy: 0.87
Confusion Matrix
[[117  25]
 [ 16 165]]
```

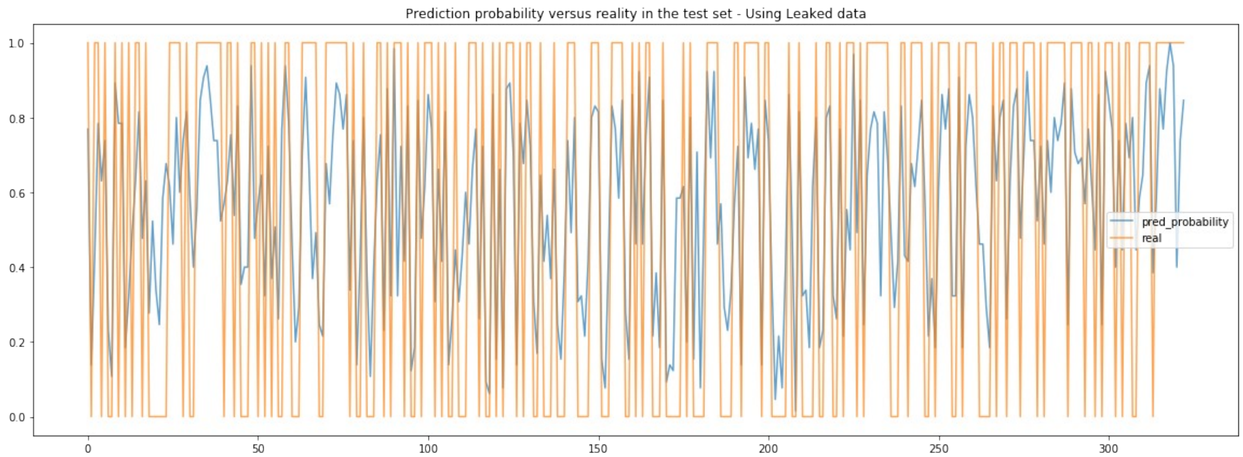
The accuracy results almost match those expected from the paper 87% vs the expected 92%

```
plt.figure(figsize=(20,7))
plt.plot(np.arange(len(pred)), pred, alpha=0.7, label='pred')
plt.plot(np.arange(len(y_test)), y_test, alpha=0.7, label='real' );
plt.title('Prediction versus reality in the test set - Using Leaked
data')
plt.legend();
```




```
plt.figure(figsize=(20,7))
proba = rf.predict_proba(X_test)[:,1]
plt.figure(figsize=(20,7))
plt.plot(np.arange(len(proba)), proba, alpha = 0.7,
label='pred_probability')
plt.plot(np.arange(len(y_test)), y_test, alpha = 0.7, label='real' );
plt.title('Prediction probability versus reality in the test set -
Using Leaked data');
plt.legend();
plt.show();

<matplotlib.figure.Figure at 0x1fec591be10>
```



Comments on Data Leaked results

The results with the data leakage approach are much more in line with those reported by the paper. In the paper it was reported a 92% accuracy for Apple, while this analysis yields 87% instead. This could be for a number of reasons including:

- the feature Williams %R was not computed
- the value for alpha was set to 0.9, but we don't know what alpha was used
- the value n for several of the technical indicators was not known, and we opted for try several values for 'n'
- more technical indicators were used in the hope that they could contribute to even better results

This analysis seems to indicate that the results from the original paper suffer from data leakage. Please, let me know if you notice any mistake in the analysis / code or if you feel there is something I misunderstood.