In [57]:

```python
import pandas as pd
data = pd.read_csv('data.csv')
```

In [58]:

```python
data.head(4)
```

Out[58]:

	DateTime	BANKBARODA_0	BANKBARODA_1	BANKBARODA_2	BANKBARODA_3	BANKBARODA_4	CANARABANK_0	CANARABA
0	04/01/17 11:06	148.90	148.90	148.90	148.90	3500	262.90	2
1	04/01/17 11:07	148.80	148.90	148.80	148.90	7000	262.75	2
2	04/01/17 11:08	148.95	149.00	148.80	148.85	21000	262.90	2
3	04/01/17 11:09	148.95	148.95	148.95	148.95	7000	262.60	2

4 rows × 26 columns

In [59]:

```python
data.tail(4)
```

Out[59]:

	DateTime	BANKBARODA_0	BANKBARODA_1	BANKBARODA_2	BANKBARODA_3	BANKBARODA_4	CANARABANK_0	CANA
114371	28/03/18 15:26	142.35	142.35	142.25	142.25	92000	263.90	
114372	28/03/18 15:27	142.25	142.35	142.25	142.30	64000	263.80	
114373	28/03/18 15:28	142.30	142.35	142.25	142.30	132000	263.90	
114374	28/03/18 15:29	142.25	142.35	142.25	142.30	124000	263.85	

4 rows × 26 columns

In [37]:

```python
len(data.columns)
```

Out[37]:

```
26
```

In [60]:

```python
data.columns =
'DateTime,BANKBARODA_OPEN,BANKBARODA_HIGH,BANKBARODA_LOW,BANKBARODA_CLOSE,BANKBARODA_VOLUME,CANARAI
OPEN,CANARABANK_HIGH,CANARABANK_LOW,CANARABANK_CLOSE,CANARABANK_VOLUME,AXISBANK,AXISBANK_OPEN,AXISI
LOW,AXISBANK_CLOSE,AXISBANK_VOLUME,SBI_OPEN,SBI_HIGH,SBI_LOW,SBI_CLOSE,SBI_VOLUME,PNB_OPEN,PNB_HIGI
_LOW,PNB_CLOSE,PNB_VOLUME'.split(sep = ',')
```

In [41]:

```python
import pandas_profiling
import numpy as np
```

In [42]:

```
pandas_profiling.ProfileReport(data)
```

Out[42]:

Overview

Dataset info

Number of variables	26
Number of observations	114375
Total Missing (%)	0.0%
Total size in memory	22.7 MiB
Average record size in memory	208.0 B

Variables types

Numeric	10
Categorical	0
Boolean	0
Date	0
Text (Unique)	1
Rejected	15
Unsupported	0

Warnings

- `AXISBANK_CLOSE` is highly correlated with `AXISBANK_LOW` (ρ = 0.99994) Rejected
- `AXISBANK_LOW` is highly correlated with `AXISBANK_OPEN` (ρ = 0.9999) Rejected
- `AXISBANK_OPEN` is highly correlated with `AXISBANK` (ρ = 0.99994) Rejected
- `BANKBARODA_CLOSE` is highly correlated with `BANKBARODA_LOW` (ρ = 0.99996) Rejected
- `BANKBARODA_HIGH` is highly correlated with `BANKBARODA_OPEN` (ρ = 0.99995) Rejected
- `BANKBARODA_LOW` is highly correlated with `BANKBARODA_HIGH` (ρ = 0.99992) Rejected
- `BANKBARODA_VOLUME` has 1886 / 1.6% zeros Zeros
- `CANARABANK_CLOSE` is highly correlated with `CANARABANK_LOW` (ρ = 0.99997) Rejected
- `CANARABANK_HIGH` is highly correlated with `CANARABANK_OPEN` (ρ = 0.99997) Rejected
- `CANARABANK_LOW` is highly correlated with `CANARABANK_HIGH` (ρ = 0.99995) Rejected
- `CANARABANK_VOLUME` has 4184 / 3.7% zeros Zeros
- `PNB_CLOSE` is highly correlated with `PNB_LOW` (ρ = 1) Rejected
- `PNB_HIGH` is highly correlated with `PNB_OPEN` (ρ = 1) Rejected
- `PNB_LOW` is highly correlated with `PNB_HIGH` (ρ = 1) Rejected
- `SBI_CLOSE` is highly correlated with `SBI_LOW` (ρ = 0.99997) Rejected
- `SBI_HIGH` is highly correlated with `SBI_OPEN` (ρ = 0.99997) Rejected
- `SBI_LOW` is highly correlated with `SBI_HIGH` (ρ = 0.99994) Rejected

Variables

AXISBANK

Numeric

Distinct count	3451
Unique (%)	3.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	517.95
Minimum	426.05
Maximum	625.35
Zeros (%)	0.0%

Toggle details

~~AXISBANK_CLOSE~~

Highly correlated

This variable is highly correlated with `AXISBANK_LOW` *and should be ignored for analysis*

Correlation	0.99994

~~AXISBANK_LOW~~

Highly correlated

This variable is highly correlated with `AXISBANK_OPEN` *and should be ignored for analysis*

Correlation	0.9999

~~AXISBANK_OPEN~~

Highly correlated

This variable is highly correlated with `AXISBANK` *and should be ignored for analysis*

Correlation	0.99994

AXISBANK_VOLUME

Numeric

Distinct count	690
Unique (%)	0.6%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	40951
Minimum	0
Maximum	4509600
Zeros (%)	0.7%

Toggle details

~~BANKBARODA_CLOSE~~

Highly correlated

This variable is highly correlated with `BANKBARODA_LOW` *and should be ignored for analysis*

Correlation	0.99996

~~BANKBARODA_HIGH~~

Highly correlated

This variable is highly correlated with `BANKBARODA_OPEN` *and should be ignored for analysis*

Correlation	0.99995

~~BANKBARODA_LOW~~

Highly correlated

This variable is highly correlated with `BANKBARODA_HIGH` *and should be ignored for analysis*

Correlation	0.99992

BANKBARODA_OPEN

Numeric

Distinct count	1448
Unique (%)	1.3%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	162.46
Minimum	128.35
Maximum	204.2
Zeros (%)	0.0%

Toggle details

BANKBARODA_VOLUME

Numeric

Distinct count	675
Unique (%)	0.6%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	81296
Minimum	0
Maximum	4403000
Zeros (%)	1.6%

Toggle details

~~CANARABANK_CLOSE~~

Highly correlated

This variable is highly correlated with `CANARABANK_LOW` *and should be ignored for analysis*

Correlation	0.99997

~~CANARABANK_HIGH~~

Highly correlated

This variable is highly correlated with `CANARABANK_OPEN` *and should be ignored for analysis*

Correlation	0.99997

~~CANARABANK_LOW~~

Highly correlated

This variable is highly correlated with `CANARABANK_HIGH` *and should be ignored for analysis*

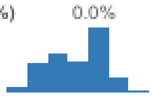
Correlation	0.99995

CANARABANK_OPEN

Numeric

Distinct count	4028
Unique (%)	3.5%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	331.77
Minimum	226.55
Maximum	462.1
Zeros (%)	0.0%



Toggle details

CANARABANK_VOLUME

Numeric

Distinct count	759
Unique (%)	0.7%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	44442
Minimum	0
Maximum	3121008
Zeros (%)	3.7%

Toggle details

DateTime

Categorical, Unique

First 3 values
22/03/18 12:26
10/08/17 15:22
27/03/17 12:08

Last 3 values
15/09/17 11:35
03/08/17 9:41
25/01/17 12:38

Toggle details

~~PNB_CLOSE~~

Highly correlated

This variable is highly correlated with `PNB_LOW` *and should be ignored for analysis*

Correlation	1

~~PNB_HIGH~~

Highly correlated

This variable is highly correlated with `PNB_OPEN` *and should be ignored for analysis*

Correlation	1

~~PNB_LOW~~

Highly correlated

This variable is highly correlated with `PNB_HIGH` *and should be ignored for analysis*

Correlation	1

PNB_OPEN

Numeric

Distinct count	32025
Unique (%)	28.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	7751.7
Minimum	91.85
Maximum	11085
Zeros (%)	0.0%

Toggle details

PNB_VOLUME

Numeric

Distinct count	3050
Unique (%)	2.7%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	54428
Minimum	0
Maximum	8888000
Zeros (%)	0.1%

Toggle details

~~SBI_CLOSE~~

Highly correlated

This variable is highly correlated with `SBI_LOW` *and should be ignored for analysis*

Correlation	0.99997

~~SBI_HIGH~~

Highly correlated

This variable is highly correlated with `SBI_OPEN` *and should be ignored for analysis*

Correlation	0.99997

~~SBI_LOW~~

Highly correlated

This variable is highly correlated with `SBI_HIGH` *and should be ignored for analysis*

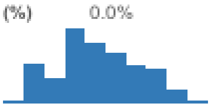
Correlation	0.99994

SBI_OPEN

Numeric

Distinct count	2220
Unique (%)	1.9%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	286.39
Minimum	232.45
Maximum	348.6
Zeros (%)	0.0%



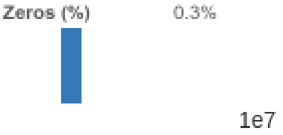
Toggle details

SBI_VOLUME

Numeric

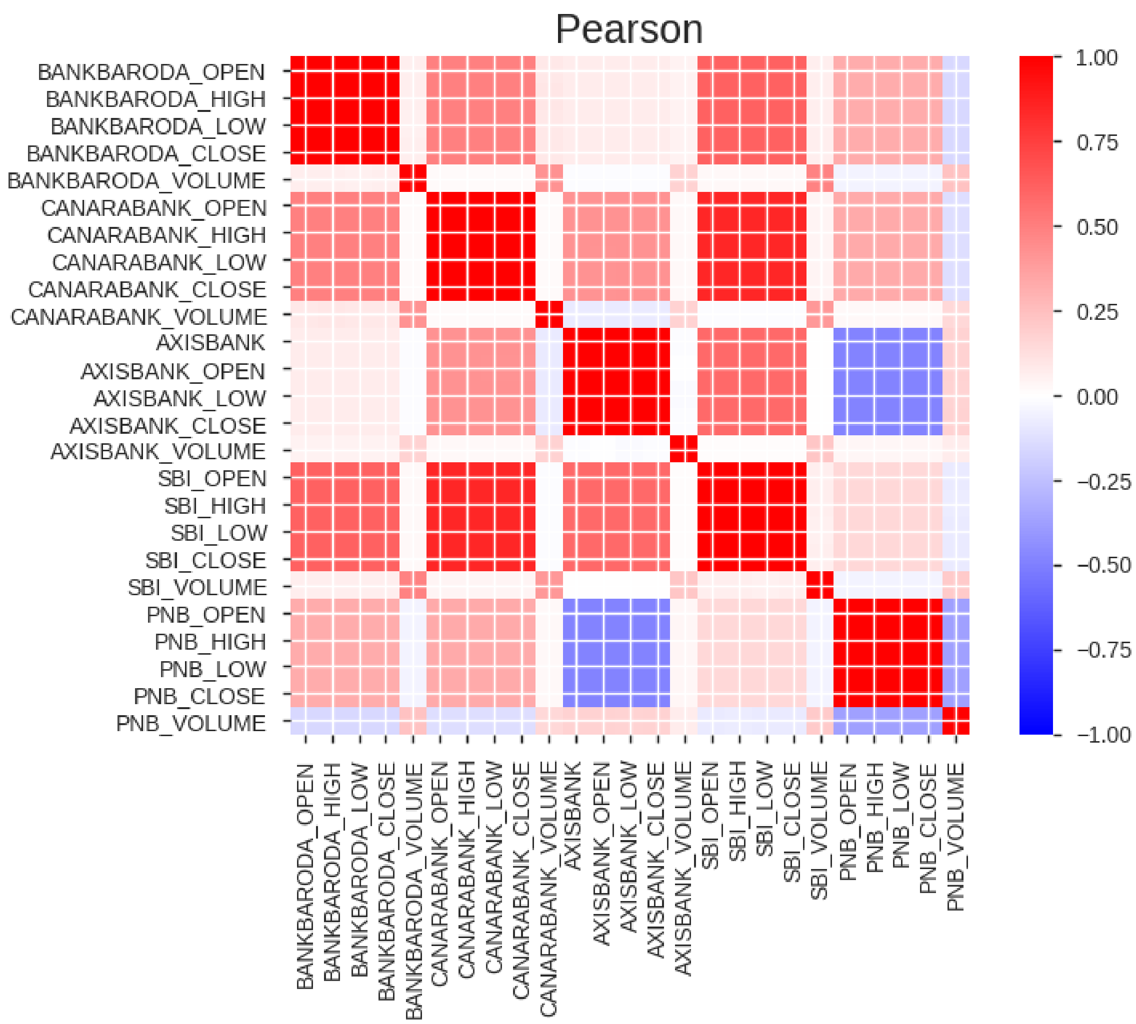
Distinct count	746

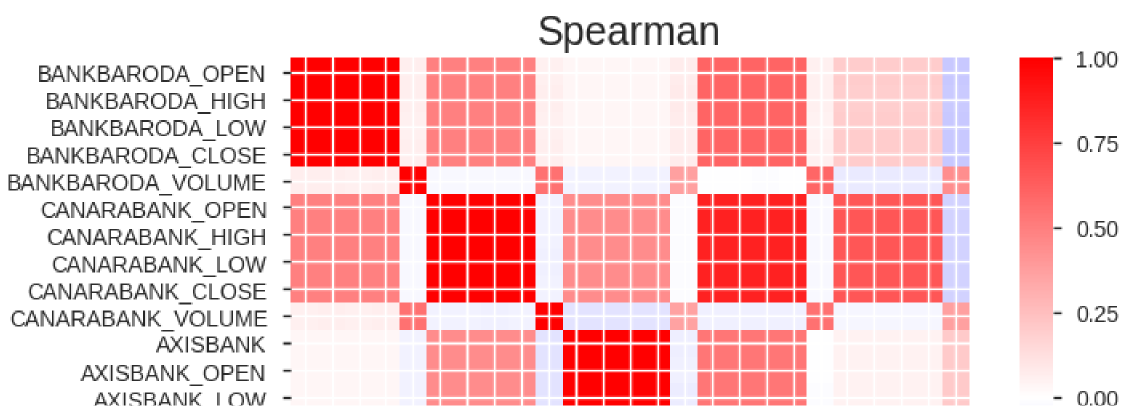
Unique (%)	0.7%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	130380
Minimum	0
Maximum	11187000
Zeros (%)	0.3%

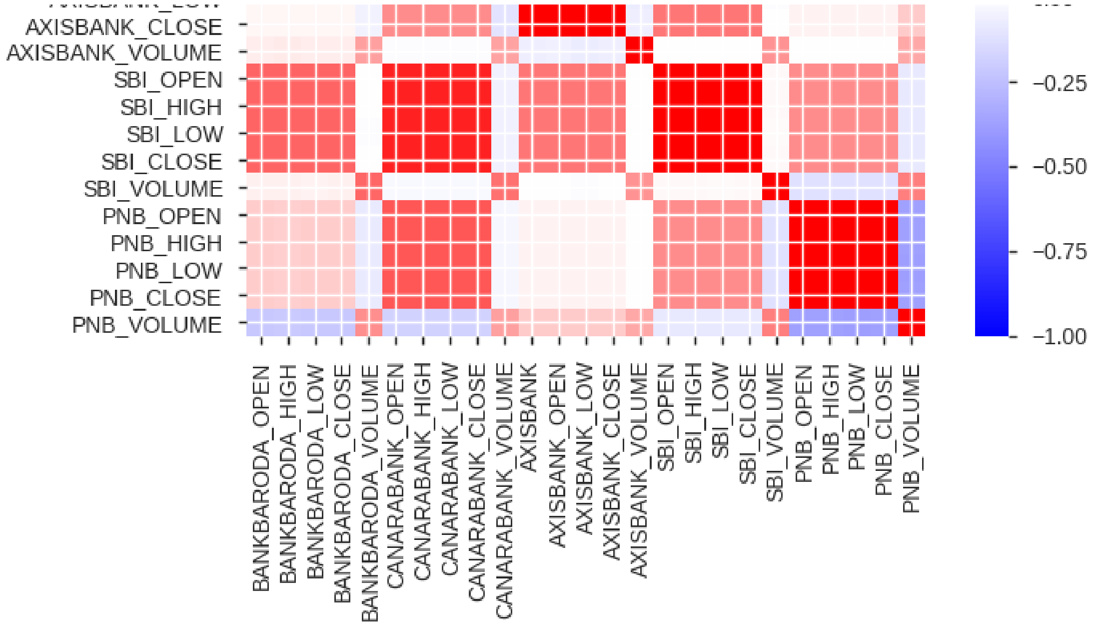


Toggle details

Correlations







Sample

	DateTime	BANKBARODA_OPEN	BANKBARODA_HIGH	BANKBARODA_LOW	BANKBARODA_CLOSE	BANKBARODA_\
0	04/01/17 11:06	148.90	148.90	148.90	148.90	
1	04/01/17 11:07	148.80	148.90	148.80	148.90	
2	04/01/17 11:08	148.95	149.00	148.80	148.85	
3	04/01/17 11:09	148.95	148.95	148.95	148.95	
4	04/01/17 11:10	148.85	148.85	148.75	148.85	

In [61]:

```
data[data.BANKBARODA_VOLUME == 0]    # bank_of_baroda4 and canara_bank4 has some zero values...(cor
```

```
responding
#volume is zero..i'm not sure if that data is right..)
```

Out[61]:

	DateTime	BANKBARODA_OPEN	BANKBARODA_HIGH	BANKBARODA_LOW	BANKBARODA_CLOSE	BANKBARODA_VOLUME
44	04/01/17 11:50	149.00	149.00	149.00	149.00	0
74	04/01/17 12:20	148.70	148.70	148.70	148.70	0
75	04/01/17 12:21	148.70	148.70	148.70	148.70	0
81	04/01/17 12:27	148.95	148.95	148.95	148.95	0
103	04/01/17 12:49	148.65	148.65	148.65	148.65	0
108	04/01/17 12:54	148.70	148.70	148.70	148.70	0
148	04/01/17 13:34	148.55	148.55	148.55	148.55	0
203	04/01/17 14:29	148.40	148.40	148.40	148.40	0
264	04/01/17 15:30	148.50	148.50	148.50	148.50	0
400	05/01/17 11:30	150.90	150.90	150.90	150.90	0
421	05/01/17 11:51	150.65	150.65	150.65	150.65	0
433	05/01/17 12:03	150.80	150.80	150.80	150.80	0
480	05/01/17 12:50	150.80	150.80	150.80	150.80	0
557	05/01/17 14:07	151.20	151.20	151.20	151.20	0
807	06/01/17 12:01	153.30	153.30	153.30	153.30	0
832	06/01/17 12:26	153.25	153.25	153.25	153.25	0
837	06/01/17 12:31	153.35	153.35	153.35	153.35	0
856	06/01/17 12:50	153.30	153.30	153.30	153.30	0
898	06/01/17 13:32	153.65	153.65	153.65	153.65	0
952	06/01/17 14:26	153.85	153.85	153.85	153.85	0
1111	09/01/17 10:49	153.90	153.90	153.90	153.90	0
1151	09/01/17 11:29	153.80	153.80	153.80	153.80	0
1162	09/01/17 11:40	153.60	153.60	153.60	153.60	0
1165	09/01/17 11:43	153.55	153.55	153.55	153.55	0
1187	09/01/17 12:05	153.45	153.45	153.45	153.45	0
1190	09/01/17 12:08	153.40	153.40	153.40	153.40	0
1217	09/01/17 12:35	153.15	153.15	153.15	153.15	0
1222	09/01/17 12:40	153.20	153.20	153.20	153.20	0
1231	09/01/17 12:49	153.10	153.10	153.10	153.10	0
1232	09/01/17	153.10	153.10	153.10	153.10	0

	DateTime	BANKBARODA_OPEN	BANKBARODA_HIGH	BANKBARODA_LOW	BANKBARODA_CLOSE	BANKBARODA_VOLUME
...	...	...	...	...	...	...
109395	09/03/18 13:45	133.35	133.35	133.35	133.35	0
109396	09/03/18 13:46	133.35	133.35	133.35	133.35	0
109714	12/03/18 12:49	129.75	129.75	129.75	129.75	0
109728	12/03/18 13:03	129.35	129.35	129.35	129.35	0
109803	12/03/18 14:18	129.55	129.55	129.55	129.55	0
110711	15/03/18 10:41	144.65	144.65	144.65	144.65	0
110750	15/03/18 11:20	144.35	144.35	144.35	144.35	0
110829	15/03/18 12:39	145.50	145.50	145.50	145.50	0
110834	15/03/18 12:44	145.25	145.25	145.25	145.25	0
110851	15/03/18 13:01	144.85	144.85	144.85	144.85	0
110856	15/03/18 13:06	145.00	145.00	145.00	145.00	0
110860	15/03/18 13:10	145.20	145.20	145.20	145.20	0
110934	15/03/18 14:24	144.35	144.35	144.35	144.35	0
111143	16/03/18 11:38	146.25	146.25	146.25	146.25	0
111148	16/03/18 11:43	146.25	146.25	146.25	146.25	0
111241	16/03/18 13:15	146.40	146.40	146.40	146.40	0
111854	20/03/18 10:59	133.60	133.60	133.60	133.60	0
112032	20/03/18 13:57	134.90	134.90	134.90	134.90	0
112207	21/03/18 10:37	137.85	137.85	137.85	137.85	0
112264	21/03/18 11:34	137.60	137.60	137.60	137.60	0
112592	22/03/18 10:47	136.00	136.00	136.00	136.00	0
112621	22/03/18 11:16	135.85	135.85	135.85	135.85	0
112727	22/03/18 13:02	136.40	136.40	136.40	136.40	0
112728	22/03/18 13:03	136.40	136.40	136.40	136.40	0
113084	23/03/18 12:44	131.85	131.85	131.85	131.85	0
113356	26/03/18 11:01	133.70	133.70	133.70	133.70	0
113383	26/03/18 11:28	134.65	134.65	134.65	134.65	0
113725	27/03/18 10:55	143.15	143.15	143.15	143.15	0
113791	27/03/18 12:01	143.30	143.30	143.30	143.30	0
114280	28/03/18 13:55	142.80	142.80	142.80	142.80	0

1886 rows × 26 columns

In [62]:

```
data_unmodified = data.copy() #keep one copy untouched
```

Data Augmentation - by means of adding more information to the state...

In [63]:

```
#let's convert datetime to pandas data-time object
import re
def is_date(x): return np.issubdtype(x.dtype, np.datetime64)


def add_datepart(df, fldname, drop=True, time=False):
    """add_datepart converts a column of df from a datetime64 to many columns containing
    the information from the date. This applies changes inplace.

    Parameters:
    -----------
    df: A pandas data frame. df gain several new columns.
    fldname: A string that is the name of the date column you wish to expand.
        If it is not a datetime64 series, it will be converted to one with pd.to_datetime.
    drop: If true then the original date column will be removed.
    time: If true time features: Hour, Minute, Second will be added.

    Examples:
    ---------

    >>> df = pd.DataFrame({ 'A' : pd.to_datetime(['3/11/2000', '3/12/2000', '3/13/2000'],
infer_datetime_format=False) })
    >>> df

        A
    0   2000-03-11
    1   2000-03-12
    2   2000-03-13

    >>> add_datepart(df, 'A')
    >>> df

        AYear AMonth AWeek ADay ADayofweek ADayofyear AIs_month_end AIs_month_start AIs_quarter_enc
AIs_quarter_start AIs_year_end AIs_year_start AElapsed
    0   2000   3      10    11   5          71         False         False            False
alse             False        False          952732800
    1   2000   3      10    12   6          72         False         False            False
alse             False        False          952819200
    2   2000   3      11    13   0          73         False         False            False
alse             False        False          952905600
    """
    fld = df[fldname]
    fld_dtype = fld.dtype
    if isinstance(fld_dtype, pd.core.dtypes.dtypes.DatetimeTZDtype):
        fld_dtype = np.datetime64

    if not np.issubdtype(fld_dtype, np.datetime64):
        df[fldname] = fld = pd.to_datetime(fld, infer_datetime_format=True)
    targ_pre = re.sub('[Dd]ate$', '', fldname)
    attr = ['Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear',
            'Is_month_end', 'Is_month_start', 'Is_quarter_end', 'Is_quarter_start', 'Is_year_end',
'Is_year_start']
    if time: attr = attr + ['Hour', 'Minute', 'Second']

    for n in attr: df[targ_pre + n] = getattr(fld.dt, n.lower())
    df[targ_pre + 'Elapsed'] = fld.astype(np.int64) // 10 ** 9
    if drop: df.drop(fldname, axis=1, inplace=True)
```

In [64]:

```
add_datepart(data,'DateTime',drop = True,time = True)
```

In [65]:

```
data.drop(labels = ['DateTimeElapsed'],axis = 1,inplace = True) #it seems adding time elapsed is an overkill..
```

In [66]:

```
data.head(2)
```

Out[66]:

	BANKBARODA_OPEN	BANKBARODA_HIGH	BANKBARODA_LOW	BANKBARODA_CLOSE	BANKBARODA_VOLUME	CANARABANK_
0	148.9	148.9	148.9	148.9	3500	2
1	148.8	148.9	148.8	148.9	7000	2

2 rows × 40 columns

In [67]:

```
# Now we have lot of categorical variables...we need numbers..let's do it

def bool_to_numeric(x):
    if x == False:
        return 0
    else:
        return 1
for column in data.columns:
    data[column] = data[column].apply(bool_to_numeric)
```

In [68]:

```
data.head(2) # now we got a bunch of numbers ... here we did some data-augmentation....this makes figuring out seasonal patterns quite easy.
```

Out[68]:

	BANKBARODA_OPEN	BANKBARODA_HIGH	BANKBARODA_LOW	BANKBARODA_CLOSE	BANKBARODA_VOLUME	CANARABANK_
0	1	1	1	1	1	
1	1	1	1	1	1	

2 rows × 40 columns

MOTIVITING QUESTIONS:

1. How will we test Robustness and generalizability of the model ?

2. How are we going to address data sufficiency problem ?

Do we even have data sufficiency problem ?

- In classical games we can run through multiple episodes .. so the agent can go through the game until it masters.Why can't we do something similar with financial data ?First let's frame the problem :

`**Aside**` :

Deep nets are mere function approximators...so here are some sensible things we can say about them...

The two simple rules of machine learning:

```
1.Garbage in garbage out -> Encode and feed all the variables(even the ones that are even m
arginally important..) that are needed for the task.

2.Explicit is better than implicit as long as we don't incur too much representation cost.
-> For a given input and output pairs we can have a deep net(more than one hidden layer is
all that is needed) to approximate the function to arbitrary precision.That means
theoritically we can find a bunch of weights that satify all the input and output
pairs.Then what's all this fuss abount cnn's ,rnn's or any other specific way of connecting
the layers.
```

Why cnn's work ?

> we run a bunch of kernels over our input data (initially with random numbers..) and tweak those weights with gradient descent and hope to eventually end up measuring some useful statistic of the data .... say edge detection....But why this weird convolution....can't we just model it as classical multilayer perceptron.....

<img src = 'mlp.png'>

CNN

<img src = 'cnn.png' >

CNN as mere matrix multiplication but with lot's of zero weigths..

- let's unroll the square matrix(image or any data) into vectors from left to right, top to bottom, the convolution could be represented as a sparse matrix C where the non-zero elements are the elements wi,j of the kernel (with i and j being the row and column of the kernel respectively)

<img src = 'conv_as_mat.png'>

So,the conclusion is we have our input vector which will get multiplied with a bunch of other numbers(of which a lot of them happens to be zero in case of `CNN` just by design and it happens to work well....really well compared to initializing all the numbers randomly and hoping our gradient descent will eventually zero out the unnecessary ones..if it turns out to be an optimal design...but why is that not working ? .. `To answer this we got to look into gradient descent..our defacto method for searching the right numbers....`)

Aside:

> I find some elegance in thinking how by just multiplying a bunch of numbers(inputs) with some other numbers(weights) and throwing away the results that turn out to be negative(THEY CALL IT Rectified Linear Unit.. `such a terrible name`) is powering all that we now fancy as A.I ..from facebook automatic image tagging to siri..language translation.......

What's wrong with gradient descent...aka Backprop..?

Probably lot's of people used hill climbing analogy for gradient descent ..let's have a one more go with it but this time to see what's not so cool about it ....

At each point in our hyper space say we have a torch and we can only look at points that are only a unit away from us..we will go in the direction of the one that is steepest......Gradient Descent.

But does this lead to our destination ?

After doing all the matrix multiplications with random weights we are hoping to minize some metric...(error).......so we are looking for local or hopefully global minima...right .. `No` ?

The question of learnability(generalization..) in machine learning ?

> The whole field of theoritical machine learning is concerned with answering this one question...Long story short... we are looking for `flat regions` in our hyper space not necessarily a minima...in fact ..in practice.. we often end up in local maxima..that is just a hell lot flatter...

When we are using cross-validation...or splitting the data into train,valid and test....we are covering our handicap of visualizing this `hyperspace` and ensuring we are in a flat region.....

This is the most interesting part .... Why CNN'S work...or for that fact one model generalizes better than the other(i'am not referring to model selection through hyperparameter tuning...but entirely different model achitectures.....cnn's,rnn's,Dqnn's...) ?

When we have two different explanations for the same phenomenon ... choose the one that is simplest....or the one that makes least number of assumptions.....we call it `OCCAM'S RAZOR`(the common underlying thread of science.).If all the science has to be summarized in one line ... this is the closest we could get.

Here is the most elegant idea..

> Given all the inputs if we could predict the output by just using some of the inputs(model sparsity)..that's a better model compared to having/using a bit of all the inputs..by means of the above argument.Then how can we choose which ones to use..(the zero's and `w` of our weight matrix).Here we are cheating again...we are placing our trust in evolution...it turns out that when we look at things ....even though we perceive the world as a continuum.. we infact scurry through it..i.e we only breifly `pause at some points..while hastening our way through the others and ignoring at some others` here's a human face recognition in action....

<img src = 'sac.png'>

Think on these things:

> The light from some of the regions doesn't ever hit the retina.....and yet we navigate through the world pretty comfortably.....our brain is running `interpolation` or sensible guessing continuously to form continuous images...where we fixate differs from image to image(which regions encode the most informating...think shannons' information theory).

So we try out kernels of various sizes [(3,3),(5,5)..)] as some of them might look at important regions for the task at hand.....

> So cnn or any model will generalize if it can make predictions by using a smaller subset of data..in the light of `rule 2` .... in case of cnn's we are explicitly enforcing weight sparsity thereby easing out the work of gradient descent....why/when does a model work better than the other? (reffering to all the models and architectures...cnn is used only as a running example...if we encode our bias ....albeit a correct one into the architecture...it will work better than the one left to the whims of gradient descent.)

Let's generate some data..... From First Principles.

NOTE: Here i will explore data generation methods in general with some comments in regard to financial data....but i do not consider this a valid approach....a bit apprehensive about using them in the reinforcement

learning setting...towards the end i would consider `explicit modeling of the environment itself` ....neverthless i thought there is no harm in playing around a bit..

GENERATIVE MODELS : From variational autoencoders to Generative Adversarial Networks

In all of modeling we assume that our data exists in some high-dimensional manifold and it has some structure to it which we can model leaving out the noise.All the algorithms can be seen as an attempt to estimate the parameters of this underlying distribution(mean,variance...) by Expectation Maximization(for what parameters of my `assumed` family of distribution[gaussian,exponetial.....] is the data i have seen most probable..).

But what if we have a distribution like this with a gap in between .. no matter what distribution we assume we cannot model the data...

<img src = 'mix.png'>

Mixture of gaussians (probablistic instantiation of fourier series..): We can say that for the above example data is generated from a bernouli+gaussian...i.e we will assume a hidden variable which we are not able to observe that is generating different clusters and the data within the cluster is sampled from gaussian...(in practice we don't get to see that nice picture..so we don't know the number of clusters...)

So now we have set some stage for the discussion on generative models.

Unless i pursue academics i 'm unlikely to have heard about `generative moment matching networks` :

Maximum Mean Discrepancy(just a metric for measuring similarities between two distributions...like kl divergance) : Here i find using some math symbols easier than writing in plain english..say we have two random variables `x` and `y` sampled from distributions `p` and `q`: then MMD claims that the distance between these distributions can be seen as :

<img src = 'mmd.jpg'>

NOTE: similar distributions have similar moments

Now that is super counter-intuitive as how can the difference between two ditributions be equated to just the first moment(mean) of the data....but here's the kicker it's not any transformation....but only the ones that can capture all the moments of the data in orginal space into the mean in new transformed space....here's a naive example:

say `x` is one dimensional : `p` = (x1,x2,x3,.........)

Now consider `phi` our transformation : phi(x1) = (x1,x1**2) now we can see that in the new transformed space comparing mean discrepancy equates to comparing both mean and variance of the distributions in orginal space....now i think with one more leap of faith we can assume there be functions that can take care of all the moments(mean,variance,.....).[here's a much cooler idea ....we even don't need to have `phi` for estimating the mean discrepancy...]

> Basically in '90s people studied a class of functions called kernel functions...this gave rise to the popularity of svm's which are useless now ...but the kernel function idea can be used with abandon...MMD(maximum mean discrepancy) is one good use...where without knowing about the function that can `approximate all the moments of the distribution` we can calculate the squared difference between them.

Now we can take `z` (say a know gaussian distribution with mean 0,var 1) that can be fed into a neural network with random weights the output `x` will have some distribution `p` ....we already have some data ... in our case time series data of (open,high,...) `y`

<img src = 'mmd2.jpg'>

Now let's squint at the idea of kernel a bit : let's take two black and white images coming from two distributions `p` and `q`:

<img src = '1.jpg' >

Now visually we find both the images close/similar but the euclidean distance(rbf kernel..)

Now visually we find both the images close/similar but the euclidean distance(rbf kernel..) between them is large....so you have to choose a kernel which matches with your perception of similarity if you want so...this is where GAN'S fare well as we are explicitly doing that .

NOTE: generating distribution that is closer to target distribution(mathematically...euclidean,cosine...any distance metric) is straight forward as we have seen earlier but they were not popular because the outputs don't look similar `perceptually` ......here's where gan's fare well ...they don't produce outputs that are similar in strict sense but only feel similar...so ,the only applications they are widely used are to aid artists(painters and musicians) to provide some inspiration or train doctors(the paper linked in the repo).

GAN'S

Imagine at the output end there is a human looking at the outputs generated by the generator and comparing with orginal data to say whether he finds them similar or not....now if we could take derivative of his judgement we can backprop it to finally get better outputs that matches his intution...goodfellow thought what if i put a classifier instead of human judgement(it is after all human judgement as the labels are provided by us...)..

Variational auto encoders : This is the last one i 'm going to consider before i try to digress on why i think all these methods are not good for using in the context of control...i.e if we ought to use this generated data to learn optimal policy.....is a disaster.

VAE(variational auto encoder....a generative model to generate new data...) : here we are explicitly trying to reduce the distance between two distributions instead of some crazy kernel.

Variational inference : It's pretty sound area of research ... though the ideas are simpler ... finding alogrithms for optimal inference with a limiterd computational cost is a difficult problem...

VAE IN CODE:(I m not running this cell to save time...)

pytorch code: Idea is to parametrize mean and variance and reduce kl divergance btw distributions along side squared error.

Note: Here ideally the decoder should be an rnn to get us the time series .... class VariationalAutoencoder():

```
def __init__(self, num_features, num_hidden_1, num_latent):
    super(VariationalAutoencoder, self).__init__()

    ### ENCODER
    self.hidden_1 = torch.nn.Linear(num_features, num_hidden_1)
    self.z_mean = torch.nn.Linear(num_hidden_1, num_latent)

    self.z_log_var = torch.nn.Linear(num_hidden_1, num_latent)


    ### DECODER
    self.linear_3 = torch.nn.Linear(num_latent, num_hidden_1)
    self.linear_4 = torch.nn.Linear(num_hidden_1, num_features)

def reparameterize(self, z_mu, z_log_var):
    # Sample epsilon from standard normal distribution
    eps = torch.randn(z_mu.size(0), z_mu.size(1)).to(device)
    # note that log(x^2) = 2*log(x); hence divide by 2 to get std_dev
    # i.e., std_dev = exp(log(std_dev^2)/2) = exp(log(var)/2)
    z = z_mu + eps * torch.exp(z_log_var/2.)
    return z


def forward(self, x):

    ### ENCODER
    x = self.hidden_1(x)
```

```
        x = torch.relu(x)
        z_mean = self.z_mean(x)
        z_log_var = self.z_log_var(x)
        encoded = self.reparameterize(z_mean, z_log_var)

        ### DECODER
        x = self.linear_3(encoded)
        x = torch.relu(x)
        x = self.linear_4(x)
        decoded = torch.sigmoid(x)

        return z_mean, z_log_var, encoded, decoded
```

Note: The above ideas should be tested out before discarding but it's just that they don't sound useful to me in our setting.

our idea is to get new samples from the underlying distribution so that we can provide more data for our network...but ...what if we model the environment directly ... most importantly this method has been tested out thorougly in the RL setting....this blog post is quite good.[curiosity driven learning]

Schmidhuber is a swiss computer scientist who did lot of work in this area in 90's and he is also the inventor of LSTM...most people never heard his name ...largely because he is not north american..anyways ...Deep Mind tested out these ideas thoroughly

references : [Curiosity-driven Exploration by Self-supervised Prediction]

[Large-Scale Study of Curiosity-Driven Learning]

`Todo` :read and implement...

A Second look into Data Sufficency Problem for financial market data?

problem definition : a naive example(THIS IS A TOY EXAMPLE CHOSEN FOR MODELING SIMPLICITY RATHER FOR PERFORMANCE..)

GOAL : maximum expected reward

STATE SPACE : minute by minute - > (open,close,low,high,volume,...) [some technical indicators can also be added....though i have no idea if they would be useful.] - > Here we will clump past few states into one state....as we hope this encodes more information into the input

ACTION SPACE : (BUY,SELL,NOACTION,QUANTITY) -> QUANTITY : indicates how many units to buy or sell.

Here is the ideal design for the output :

Use function approximator to only predict the discrete values (BUY,SELL,NOACTION) AND FEED THIS OUTCOME ALONGSIDE THE NET account balance TO PREDICT THE QUANTITY into the next layer....[eg: $(0,0,1,1200)*(w_1,w_2,w_3,w_4) = 10$....buy ten units when the network predicts to hold on you action and you have 1200 units of capital.... `note` : This is an end-end network but in accordance with the principle 2 we are explicitly defining our bias that inorder to predict the quantity to buy/sell net outstanding capital is an important information..]

c_t = Net unrealized profits + outstanding capital

REWARD : at each time stamp - > $r_t = (c_t - c_{t-1})$

Why can't we model the above setting as an episodic game.Say we have the

data from jan-2017...jan-2018 now the game ends ... say when the drawdown reaches more than 85%(we can be more keen on this...for now just assume we have a sensible way of ending an episode..)...so our agent will go through the states..at each point trying to predict the optimal action for maximizing the reward......

Now we got to choose upon the specific instantiation of the function approximator :

Q-LEARNING

1. Take state and output quality of action(expected values of return by taking that action) -> The above continuous quantity cannot fit as predicting the quality of action for each quantity will blow up the space ....to use this setting we will remove `QUANTITY` FROM ACTION SPACE...

Implementation details:

1.Use an LSTM for the concatenated state space

2.Use a 1D convolution with an Rnn :

<img src = 'dta.png' >

Policy gradient methods

2.Take state and directly predict the output...aka policy. Here we are not predicting the expected return of each action...which is kind of downside interms of interpretability.

In [ ]:

```
# A second look at Modeling Markets as a Game : Assymetric information and
```