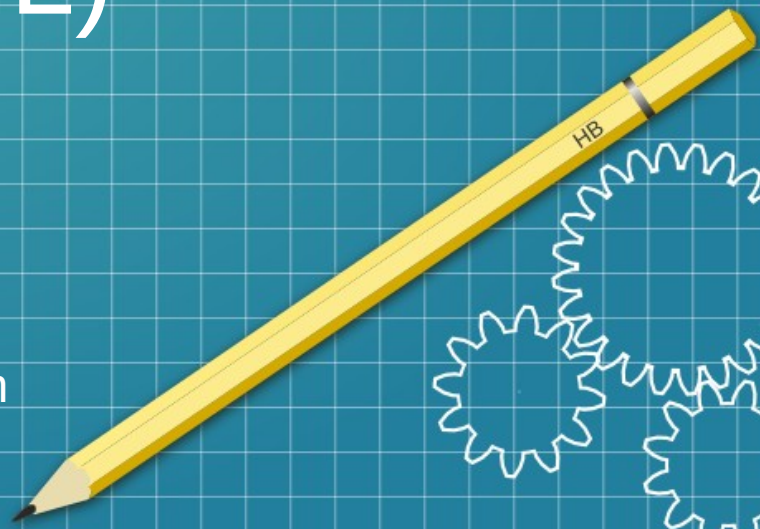
Predicting asymmetrical, aperiodic sequences (ML)

https://github.com/mtswenivh/Pseudo-random-prediction



Aim

- Propose model for predicting future outcome of asymmetric, pseudo-random generated number sequences.
    1) Pseudo random generated number analysis
    2) Stock market analysis

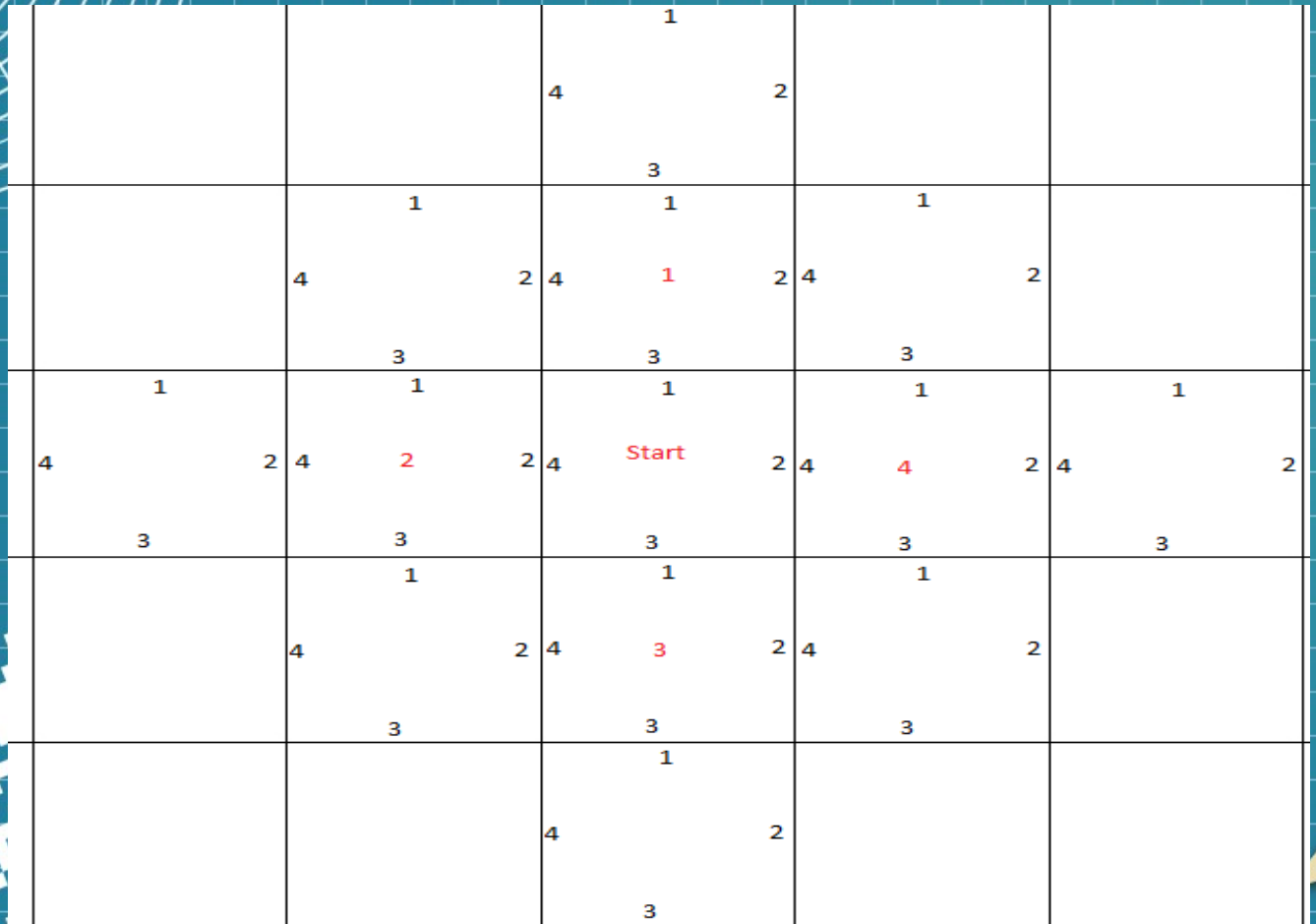
Content

- **Background**
  - Periodic square tiling geometry
  - Periodic square tiling sequence
  - Aperiodic tridecagon tiling geometry
  - Aperiodic trigecagon tiling sequence
  - Generated Data
- **Pseudo random generated number analysis**
  - Pseudo random generated vs Template sequence
  - Preprocessed data-Model variables
- **Stock market analysis (Nvidia)**
  - Stock market analysis graph
  - Preprocessed data-Model variables
- **Interim results**
  - Pseudo random number generation
  - Stock market

Background

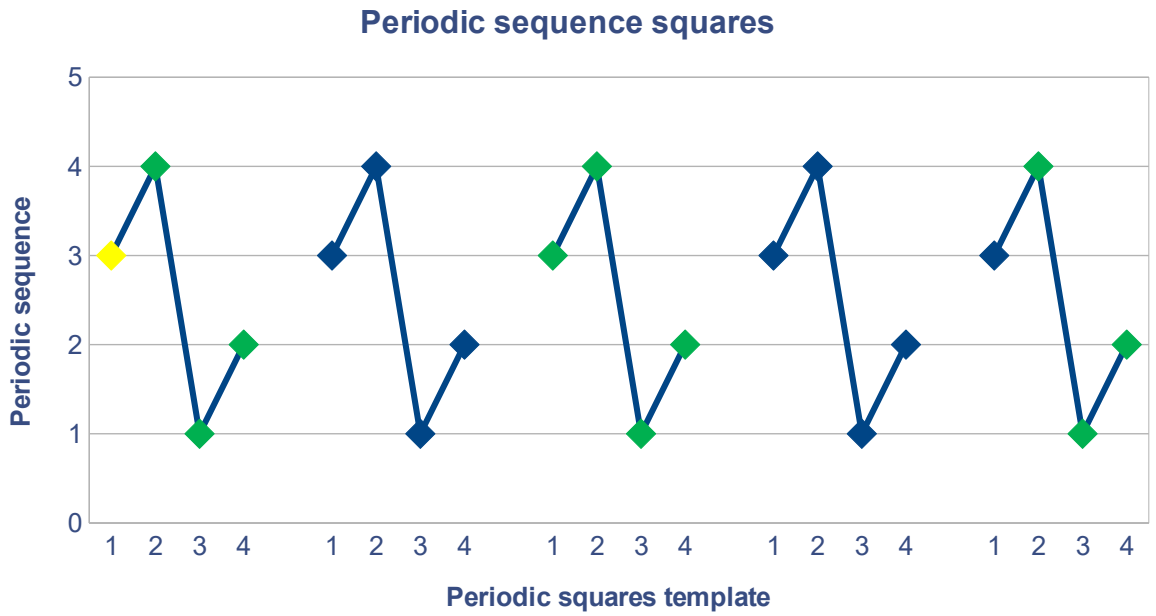
- Model based on tiling of different shapes in 2D.
  - Some shapes produce Symmetric & Periodic (e.g. squares)
  - Some shapes produce Asymmetric & Aperiodic (e.g. pentagon)
- By creating template of the tiling shape corresponding to tiling sequence of the shape predicting model is generated.

Periodic square tiling geometry



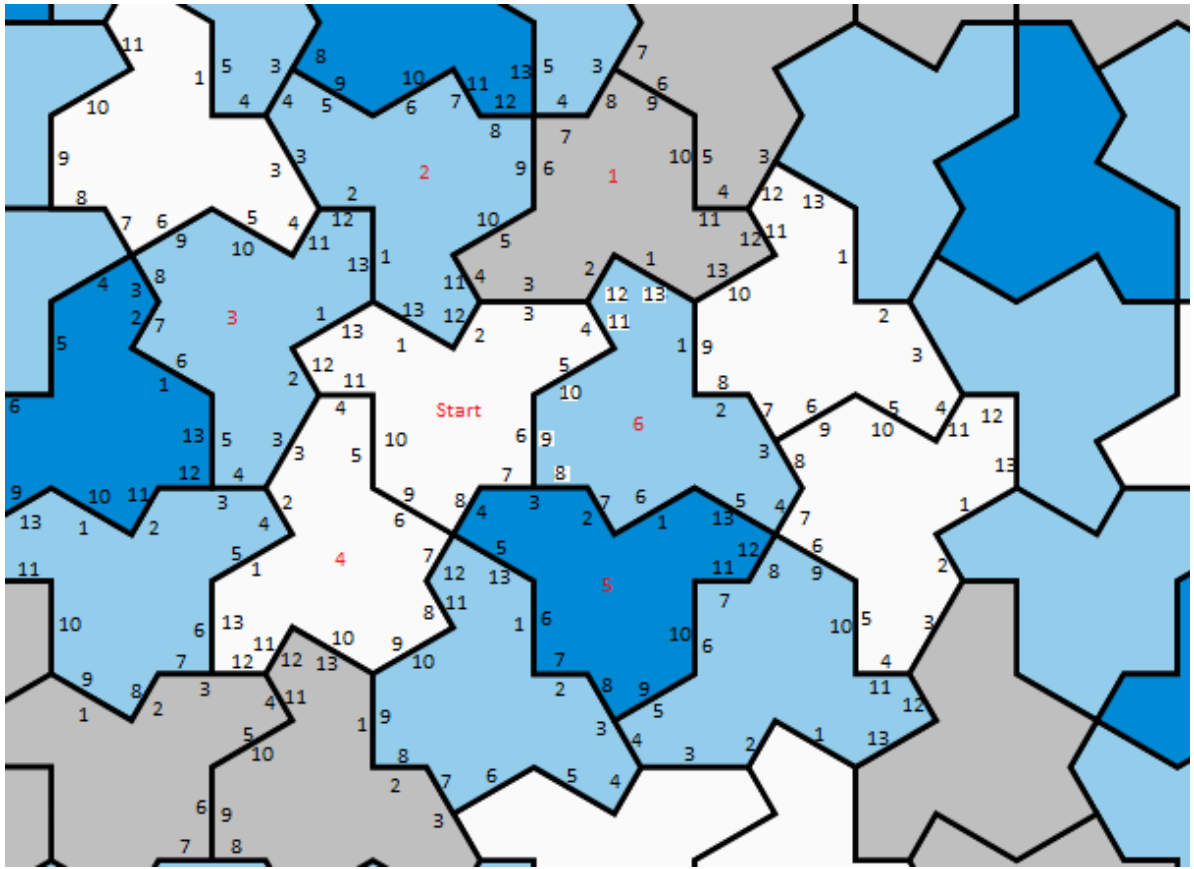
- Regular tiling of squares
- Symmetrical, periodic
- Each side of triangle is assigned a number (template 1-4)
- Starting from the middle square each number on the side referenced with adjacent squares

Periodic square tiling sequence



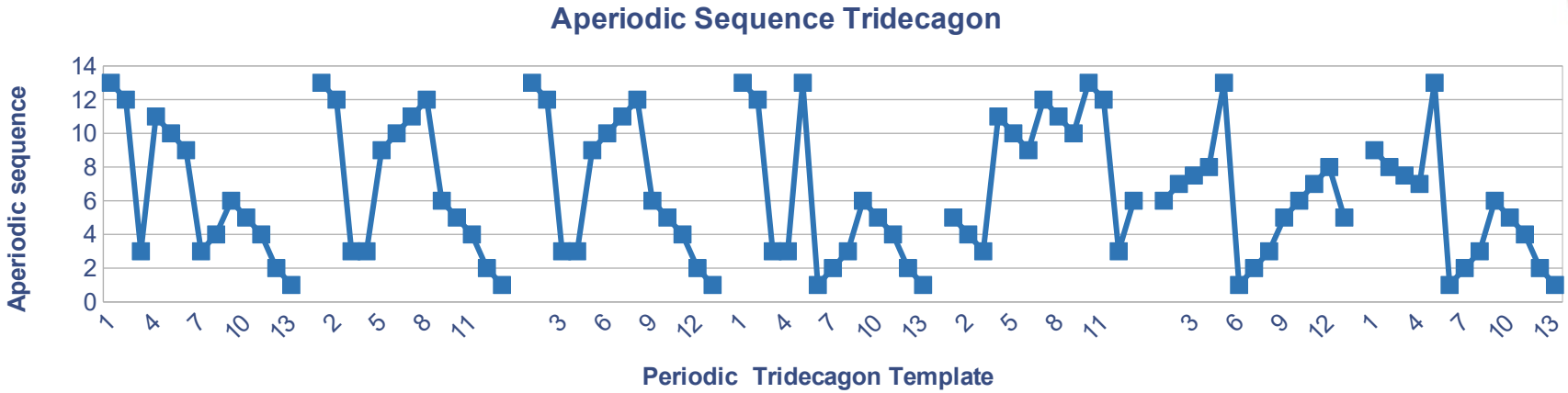
- Predictable pattern recognition
- Consistent template (1-4) matched with consistent and recurring sequence.

Aperiodic tridecagon tiling geometry



- Irregular tiling of tridecagons
- Asymmetrical, Aperiodic
- Each side of tridecagon is assigned a number (template 1-13)
- Starting from the middle square each number on the side referenced with adjacent tridecagons

Aperiodic trigecagon tiling sequence



- Asymmetric tridecagon shaped to tile without living voids
- Consistent Template (1-13) sequence, matched with inconsistent but predictable sequence

Generated data

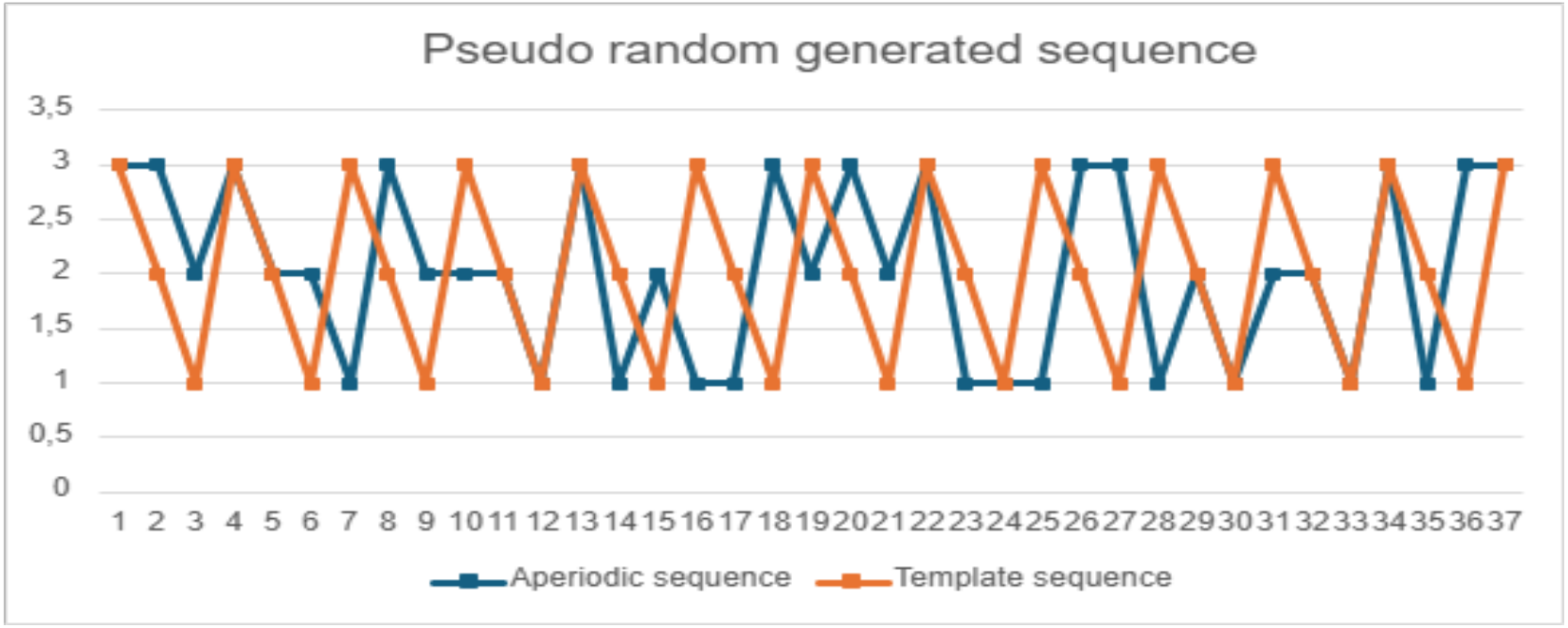
Square data									
Start		1		2		3		4	
Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence
1	3	1	3	1	3	1	3	1	3
2	4	2	4	2	4	2	4	2	4
3	1	3	1	3	1	3	1	3	1
4	2	4	2	4	2	4	2	4	2

Trigecagon data															
Start		1		2		3		4		5		6		7	
Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence	Template	Generated Sequence
1	13	1	13	1	13	1	13	1	5	1	6	1	9	1	13
2	12	2	12	2	12	2	12	2	4	2	7	2	8	2	12
3	3	3	3	3	3	3	3	3	3	3	7,5	3	7,5	3	3
4	11	4	3	4	3	4	3	4	11	4	8	4	7	4	11
5	10	5	9	5	9	5	13	5	10	5	13	5	13	5	10
6	9	6	10	6	10	6	1	6	9	6	1	6	1	6	9
7	3	7	11	7	11	7	2	7	12	7	2	7	2	7	3
8	4	8	12	8	12	8	3	8	11	8	3	8	3	8	4
9	6	9	6	9	6	9	6	9	10	9	5	9	6	9	6
10	5	10	5	10	5	10	5	10	13	10	6	10	5	10	5
11	4	11	4	11	4	11	4	11	12	11	7	11	4	11	4
12	2	12	2	12	2	12	2	12	3	12	8	12	2	12	2
13	1	13	1	13	1	13	1	13	6	13	5	13	1	13	1

Pseudo random generated number analysis

- Numbers generated in a controlled system with equal chance of selecting each number.
- Outcome Range 1-3, Template 1,2,3

Pseudo random generated vs Template sequence



Processed data-model variables

Out[131]:

	Target	Current code	Position code	Prev position code	D	E	Position number	Temp pos	Prev pos	Temp	Class1	Class 2	Class 3
0	1	16	16	13	10	100	3	1	1	6	500	100	25
1	2	12	19	16	10	100	1	1	1	9	500	100	25
2	2	15	13	19	100	10	2	1	1	3	75	50	75
3	2	18	16	13	100	100	1	1	1	6	500	25	100
4	3	11	16	13	10	100	3	1	1	6	100	100	25
5	3	11	16	13	10	100	1	1	1	6	50	75	50
6	3	14	19	16	100	100	1	1	1	9	500	25	100
7	3	14	19	16	100	100	3	1	1	9	50	100	25
8	1	19	19	15	100	100	4	1	2	9	500	25	100
9	2	12	19	15	10	100	5	1	2	9	200	75	50

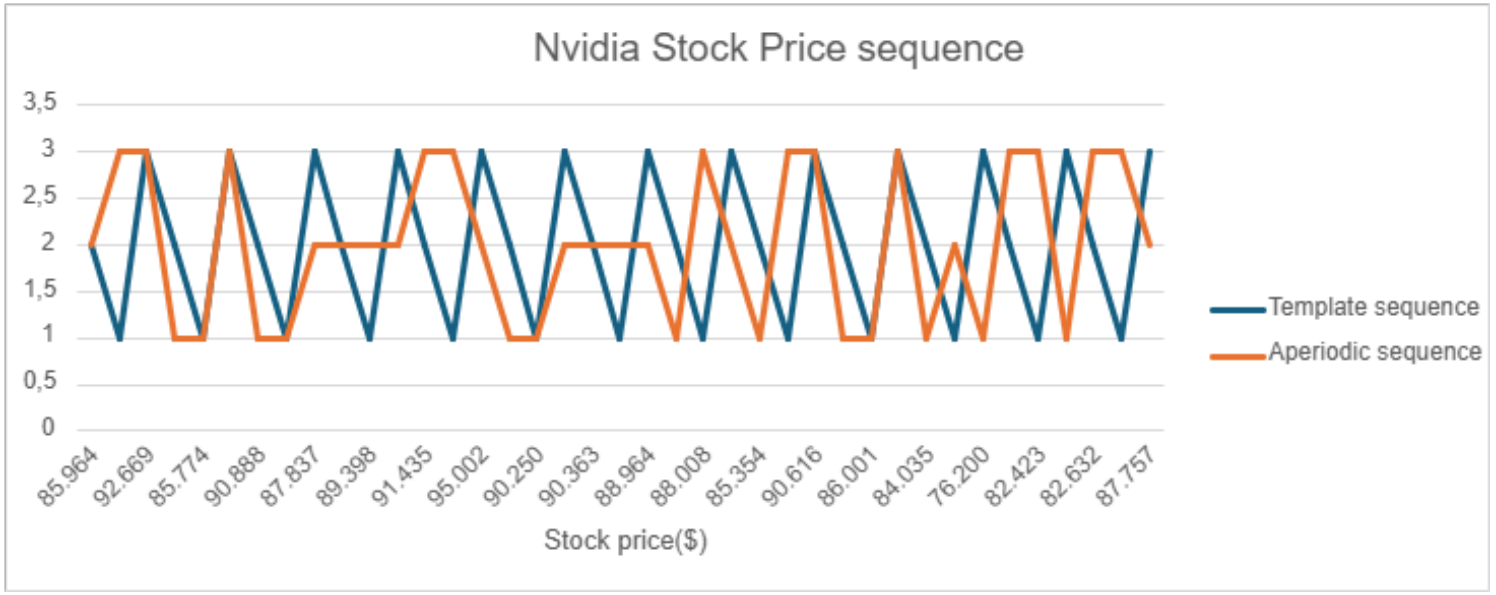
- **Current code**-current position in spectrum of probabilities
- **Position code**-Template position
- **Previous position code**-previous position in spectrum of probabilities
- **D &E**- lower and upper limit on scale 0-100
- **Position number**-position on spectrum of Template-Target combinations (1-9)
- **Temp position**-Template outcome
- **Prev pos**- Previous actual outcome
- **Class 1**-magnitude of difference between estimated previous & current values
- **Class 2**-magnitude of difference between actual previous & estimated current values
- **Class 3**-magnitude of difference between lower and upper limit values

Stock market analysis (Nvidia)

- Stock price tracking
- Three movements negative(1), lateral(2), positive(3) range, Template 1-3

Stock market analysis (Nvidia)

- **Stock price movement**: 1=negative movement(<2%), 2=neutral movement(>2%), 3=positive movement((<2%)



Processed data-model variables

:1]:

	Target	Template	Price	Position code	Prev position code	Position number	Temp pos	Prev pos	Class 1	Class 2	Class 3	Lo-Hi limit	Trend_limit	Trend direction	Template_limit
0	2	6	85.96	22	33	9	2	3	200	200	200	10	1000	1	0
1	1	3	88.70	11	22	6	1	1	30	30	200	0	-1000	-1	1
2	3	9	92.67	33	31	1	3	2	30	30	30	100	1000	1	1
3	1	6	87.53	22	33	8	2	3	200	200	30	10	1000	1	1
4	3	3	85.77	13	11	3	3	1	30	30	200	100	1000	-1	0
5	2	9	91.91	32	13	7	2	3	200	200	30	30	5000	1	1
6	1	6	90.89	22	32	6	2	2	200	45	200	100	3000	2	0
7	3	3	87.94	13	11	2	3	1	30	30	45	50	1000	-1	0
8	1	9	87.84	32	13	7	2	3	200	200	30	10	1000	1	1
9	2	6	88.46	22	21	3	2	1	45	30	200	100	3000	2	0

- **Template**
- **Price-stock**
- **Position code**-Template position
- **Previous position code**-previous position in spectrum of probabilities
- **Position number**-position on spectrum of Template/Target combinations (1-9)
- **Temp position-Template outcome**
- **Prev pos- Previous actual outcome**
- **Class 1**-magnitude of difference between estimated previous & current values
- **Class 2**-actual previous
- **Class 3**-prior to actual previous
- **Lo-Hi limit-** magnitude lower and upper limit scaled
- **Trend_limit**-Class sequence compared to lower-upper limit
- **Trend direction**- positive vs negative trend
- **Template limit: compare lower**-upper limt vs template

Interim Results (Pseudo random number generation)

Hypertuned parameters Accuracy

- **Logistic regression** - Accuracy 47%
- **SVM** - Accuracy 57%
- **Decision tree** - Accuracy 98%
- **Knn** - Accuracy 59%

- **Future improvement**
  - Decision tree yields high accuracy
  - Test repeatability of results with other datasets
  - Explore/Create more tailored models

Interim results (Stock market)

Hypertuned parameters Accuracy

- **Logistic regression** - Accuracy 53%
- **SVM**-Accuracy 60%
- **Decision tree**-Accuracy 63%
- **Knn**-Accuracy 59%

- **Future improvement**
  1. Weighted variable importance
  2. More distinct values for different variable
  3. Explore/Create more tailored models

https://github.com/mtswenivh/Pseudo-random-prediction