Kaggle Tips For Feature Engineering and Selection

Gilberto Titericz Jr
November / 2019
Spain

Single Model Accuracy

- Quality of the dataset/ground truth
- Size of the dataset
- Feature pre-processing
- Feature Engineering
- Feature Selection
- Model selection
- Hyperparameter optimization

Multi Model Accuracy

- Accuracy of Single Models
- Diversity of Models
- Model Selection
- Post Processings

Exploratory Data Analysis

- EDA is part of Data Science Pipeline
  - Explore the data using aggregation and plot tools:
    - Eg. R: data.table, ggplot2, DataExplorer
      Python: pandas, matplotlib, seaborn
      MS Excel

Kaggle – Santander Value Prediction Challenge

- Train set: 4459 x 4993

Kaggle – Santander Value Prediction Challenge

ID	target	48df886f9	0deb4b6a8	34b15f335	a8cb14b00	2f0771a37	30347e683	d08d1fbe3	6ee66e115	20aa07010	dc5a8f1d8	11d86fa6a	77c9823f2	8d6c2a0b2	4681de4fd
000d6aaf2	38000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
000fbd867	600000	0	0	0	0	0	0	0	0	2200000	0	0	0	0	0
0027d6b71	10000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0028cbf45	2000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
002a68644	14400000	0	0	0	0	0	0	0	0	2000000	0	0	0	0	0
002dbeb22	2800000	0	0	0	0	0	0	0	0	17020000	0	8000	0	0	0
003925ac6	164000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
003eb0261	600000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004b92275	979000	0	0	0	0	0	0	0	0	58000	0	0	0	0	22000
0067b4fef	460000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00689ee2c	1100000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0069007ac	16000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
006b60dd7	354000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8057126	7000000	0	0	0	0	0	0	0	0	0	0	0	0	0	5000000
8825875	100000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0096e207e	800000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00c2deb75	200000	0	0	0	0	0	0	0	0	7000	0	0	0	0	0
00ce2134f	3600000	0	0	0	0	0	0	0	0	0	0	0	0	0	1200000

EDA

ID	target	f190486d6	58e2e02e6	eeb9cd3aa	9fd594eec	6eef030c1	15ace8c9f	fb0f5dbfe	58e056e12
7862786dc	3513333.3	0	1477600	1586889	75000	3147200	466461.5	1600000.0	0.0
c95732596	160000.0	310000	0	1477600	1586889	75000	3147200.0	466461.5	1600000.0
16a02e67a	2352551.7	3513333	310000	0	1477600	1586889	75000.0	3147200.0	466461.5
ad960f947	280000.0	160000	3513333	310000	0	1477600	1586888.9	75000.0	3147200.0
8adafbb52	5450500.0	2352552	160000	3513333	310000	0	1477600.0	1586888.9	75000.0
fd0c7cfc2	1359000.0	280000	2352552	160000	3513333	310000	0.0	1477600.0	1586888.9
a36b78ff7	60000.0	5450500	280000	2352552	160000	3513333	310000.0	0.0	1477600.0
e42aae1b8	12000000.0	1359000	5450500	280000	2352552	160000	3513333.3	310000.0	0.0
0b132f2c6	500000.0	60000	1359000	5450500	280000	2352552	160000.0	3513333.3	310000.0
448efbb28	1878571.4	12000000	60000	1359000	5450500	280000	2352551.7	160000.0	3513333.3
ca98b17ca	814800.0	500000	12000000	60000	1359000	5450500	280000.0	2352551.7	160000.0
2e57ec99f	307000.0	1878571	500000	12000000	60000	1359000	5450500.0	280000.0	2352551.7
fef33cb02	528666.7	814800	1878571	500000	12000000	60000	1359000.0	5450500.0	280000.0

kaggle DAYS

EDA

ID	target	f190486d6	58e2e02e6	eeb9cd3aa	9fd594eec	6eef030c1	15ace8c9f	fb0f5dbfe	58e056e12
7862786dc	3513333.3	0	1477600	1586889	75000	3147200	466461.5	1600000.0	0.0
c95732596	160000.0	310000	0	1477600	1586889	75000	3147200.0	466461.5	1600000.0
16a02e67a	2352551.7	3513333	310000	0	1477600	1586889	75000.0	3147200.0	466461.5
ad960f947	280000.0	160000	3513333	310000	0	1477600	1586888.9	75000.0	3147200.0
8adafbb52	5450500.0	2352552	160000	3513333	310000	0	1477600.0	1586888.9	75000.0
fd0c7cfc2	1359000.0	280000	2352552	160000	3513333	310000	0.0	1477600.0	1586888.9
a36b78ff7	60000.0	5450500	280000	2352552	160000	3513333	310000.0	0.0	1477600.0
e42aae1b8	12000000.0	1359000	5450500	280000	2352552	160000	3513333.3	310000.0	0.0
0b132f2c6	500000.0	60000	1359000	5450500	280000	2352552	160000.0	3513333.3	310000.0
448efbb28	1878571.4	12000000	60000	1359000	5450500	280000	2352551.7	160000.0	3513333.3
ca98b17ca	814800.0	500000	12000000	60000	1359000	5450500	280000.0	2352551.7	160000.0
2e57ec99f	307000.0	1878571	500000	12000000	60000	1359000	5450500.0	280000.0	2352551.7
fef33cb02	528666.7	814800	1878571	500000	12000000	60000	1359000.0	5450500.0	280000.0

kaggle DAYS

EDA

ID	target	f190486d6	58e2e02e6	eeb9cd3aa	9fd594eec	6eef030c1	15ace8c9f	fb0f5dbfe	58e056e12
7862786dc	3513333.3	0	1477600	1586889	75000	3147200	466461.5	1600000.0	0.0
c95732596	160000.0	310000	0	1477600	1586889	75000	3147200.0	466461.5	1600000.0
16a02e67a	2352551.7	3513333	310000	0	1477600	1586889	75000.0	3147200.0	466461.5
ad960f947	280000.0	160000	3513333	310000	0	1477600	1586888.9	75000.0	3147200.0
8adafbb52	5450500.0	2352552	160000	3513333	310000	0	1477600.0	1586888.9	75000.0
fd0c7cfc2	1359000.0	280000	2352552	160000	3513333	310000	0.0	1477600.0	1586888.9
a36b78ff7	60000.0	5450500	280000	2352552	160000	3513333	310000.0	0.0	1477600.0
e42aae1b8	12000000.0	1359000	5450500	280000	2352552	160000	3513333.3	310000.0	0.0
0b132f2c6	500000.0	60000	1359000	5450500	280000	2352552	160000.0	3513333.3	310000.0
448efbb28	1878571.4	12000000	60000	1359000	5450500	280000	2352551.7	160000.0	3513333.3
ca98b17ca	814800.0	500000	12000000	60000	1359000	5450500	280000.0	2352551.7	160000.0
2e57ec99f	307000.0	1878571	500000	12000000	60000	1359000	5450500.0	280000.0	2352551.7
fef33cb02	528666.7	814800	1878571	500000	12000000	60000	1359000.0	5450500.0	280000.0

Numerical Correlation

- Before building features calculate the correlation of numerical features and target.

```
train = pd.read_csv( 'train-titanic.csv' )
train.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

kaggle DAYS

Numerical Correlation

- Before building features calculate the correlation of numerical features and target.

```python
train = pd.read_csv( 'train-titanic.csv' )
train.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

Feature Engineering

- **Wiki: "Feature engineering** is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive."

- ***Andrew Ng****:* *"*Coming up with **features** is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering."

Combining Numerical Features

- Explore Linear Combinations (2-way, 3-way, …) :
  - A + B
  - A - B
  - A * B
  - A / B
  - A ^ B
  - Log(A) * Log(B)
  - A * Exp(B)
  - Rank(A) + Rank(B)
  - sin(A) + cos(B)
  - w1 * A + w2 * B
  - Etc…

Imputing Nan

Var 1	Var 2
1	NA
4	7
2	NA
NA	6
5	8
8	2

kaggle DAYS

Imputing Nan – For Linear Models

Add NA Flags

Var 1	Var 2	Var1_flag	Var2_flag
1	NA	0	1
4	7	0	0
2	NA	0	1
NA	6	1	0
5	8	0	0
8	2	0	0

Imputing Nan – For Linear Models

Scale Variables

Var 1	Var 2	Var1_flag	Var2_flag
-0.5	**0**	0	1
0	0.24	0	0
-0.333	**0**	0	1
0	0.048	1	0
0.167	0.43	0	0
0.667	-0.72	0	0

kaggle DAYS

Categorical Features

- Simple LabelEncoder.
- Use Categorical to group and calculate statistics of other variables
- Frequency(count) encoding.
- One Hot Encoder (OHE).
- Hash Encoder + OHE (hash(value) % 100).
- Mean Target Encoder.

Mea Target Encoding

- The idea of using target label information to encode variables.
- Works for both classification and regressions problems.
- Target Encoding works better for high cardinality variables.
- Must be processed using cross-validation or out-of-fold encoding.

Target Encoding

Color	Target	Target Encoder
Red	0	0.33
Red	0	0.33
Red	1	0.33
Blue	0	0.67
Blue	1	0.67
Blue	1	0.67
Green	0	0.00

Target Encoding

Color	Target	Target Encoder
Red	0	0.33
Red	0	0.33
Red	1	0.33
Blue	0	0.67
Blue	1	0.67
Blue	1	0.67
Green	0	0.00

$$\frac{sum([0,0,1])}{3}$$

$$\frac{sum([0,1,1])}{3}$$

$$\frac{sum([0])}{1}$$

Target Encoding

```
train = pd.read_csv( 'amazon-employee-access-challenge/train.csv' )
```

Feature	Direct Mean Target Encoder (Leak) AUC	5-Fold Mean Target Encoder AUC
RESOURCE	0.92	0.61
MGR_ID	0.95	0.79
ROLE_ROLLUP_1	0.62	0.59
ROLE_ROLLUP_2	0.68	0.65
ROLE_DEPTNAME	0.78	0.72
ROLE_TITLE	0.72	0.68

Target Encoding N-Way

```python
train[ '3WAY' ] = (
    train['MGR_ID'].apply(str) +'_'+
    train['ROLE_DEPTNAME'].apply(str) +'_'+
    train['ROLE_FAMILY_DESC'].apply(str) )

train[['MGR_ID', 'ROLE_DEPTNAME', 'ROLE_FAMILY_DESC','3WAY']].head()
```

	MGR_ID	ROLE_DEPTNAME	ROLE_FAMILY_DESC	3WAY
0	85475	123472	117906	85475_123472_117906
1	1540	123125	118536	1540_123125_118536
2	14457	117884	267952	14457_117884_267952
3	5396	119993	240983	5396_119993_240983
4	5905	119569	123932	5905_119569_123932

Target Encoding

train = pd.read_csv('amazon-employee-access-challenge/train.csv')

Feature	Direct Mean Target Encoder (Leak) AUC	5-Fold Mean Target Encoder AUC
RESOURCE	0.92	0.61
MGR_ID	0.95	0.79
ROLE_ROLLUP_1	0.62	0.59
ROLE_ROLLUP_2	0.68	0.65
ROLE_DEPTNAME	0.78	0.72
ROLE_TITLE	0.72	0.68
3WAY	0.98	0.82

Target Encoding

```
def target_encode_simple( df_train, df_valid, col, target ):

    dt  = df_train.groupby(col)[target].agg(['mean']).reset_index(drop=False)

    tmp = df_valid.merge( dt, on=col, how='left' )['mean'].values

    return tmp
```

Target Encoding – Out-of-Fold

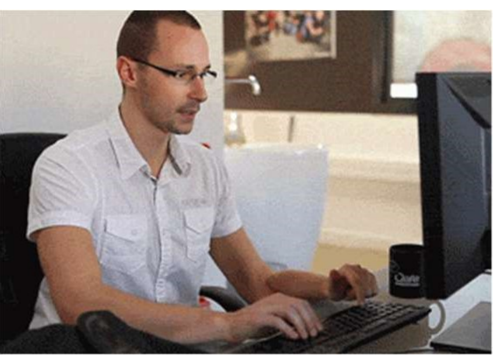
Trainset

Validset

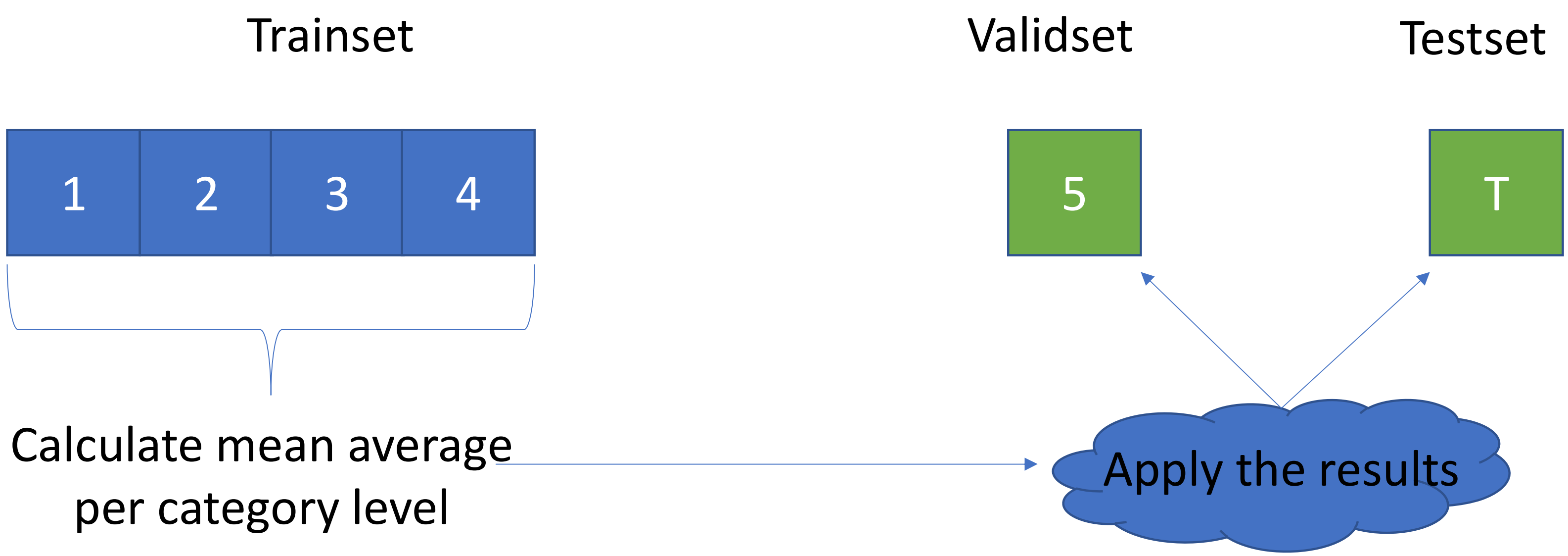
Testset

1	2	3	4	5

T



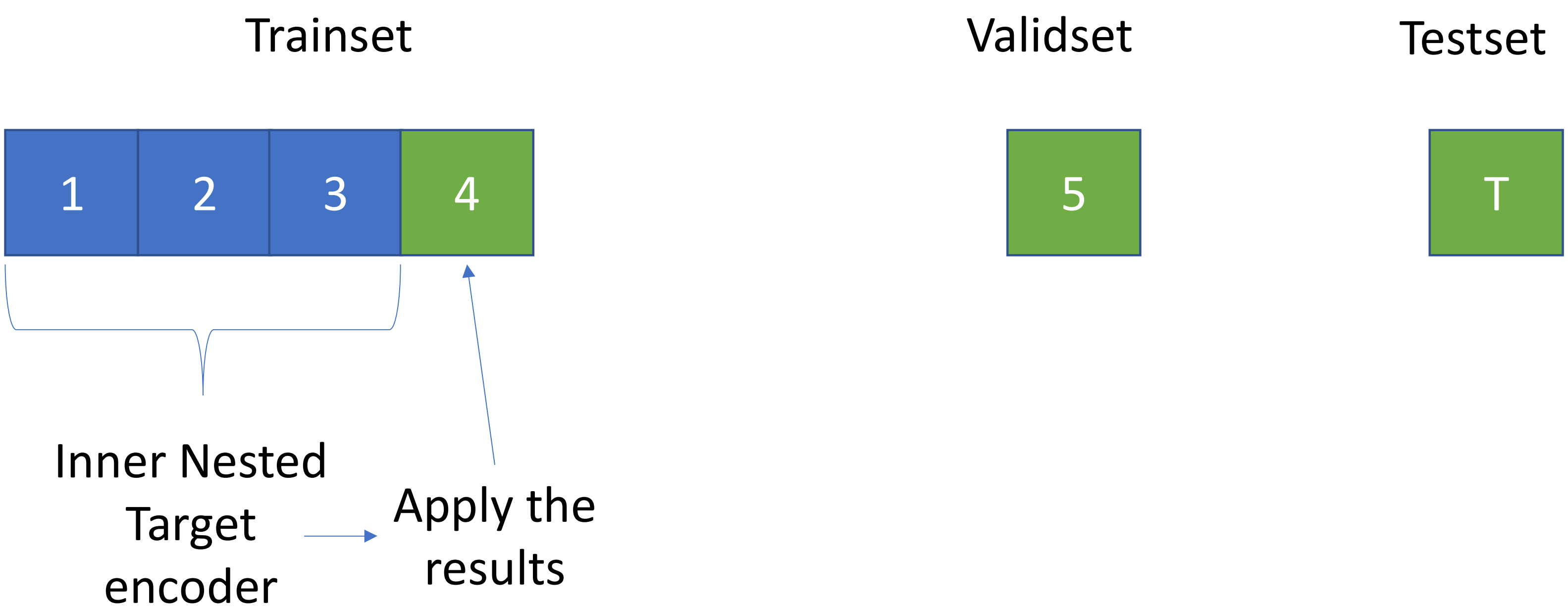
Target Encoding – Simple Out-of-Fold



Target Encoding – Nested Out-of-Fold

Run another CV inside
Folds: 1,2,3 and 4 to
calculate Target Encoding of
the Trainset.

Target Encoding – Nested Out-of-Fold



Aggregations

```
train = pd.read_csv( 'train-titanic.csv' )
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

```
df = train.groupby( ['Pclass'] )['Age'].agg( ['count','mean','max','min','std','skew'] ).reset_index()
df
```

	Pclass	count	mean	max	min	std	skew
0	1	186	38.233441	80.0	0.92	14.802856	0.119857
1	2	173	29.877630	70.0	0.67	14.001077	0.133837
2	3	355	25.140620	74.0	0.42	12.495398	0.483990

kaggle DAYS

Merging Aggregations

```
train.merge( df, on=['Pclass'], how='left' ).head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	count	mean	max	min	std	skew
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	355	25.140620	74.0	0.42	12.495398	0.483990
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	186	38.233441	80.0	0.92	14.802856	0.119857
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	355	25.140620	74.0	0.42	12.495398	0.483990
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	186	38.233441	80.0	0.92	14.802856	0.119857

Scaling

- Scaling is not necessary for Decision Tree based models.

- Scaling is necessary for Linear Models and Neural.
  - Standard Scaler
  - Min Max Scaler
  - Max Absolute Scaler
  - Signed Logp1 Scaling
  - Sklearn Robust Scaler (scales to 25% and 75% quantiles)
  - Sklearn PowerTransformer (makes data more Gaussian-like)

kaggle DAYS

Target Engineering Transformation

- Explore different viewpoints of target label
- $y_t = f(y)$ and $y = f^{-1}(y^t)$
- More useful for regression.
- Can be used for classification

Target Engineering Transformation

Direct	Inverse
$f(x) = x^{\frac{1}{n}}$	$f(x) = x^{n}$
$f(x) = \ln(x+1)$	$f^{-1}(x) = e^{x} - 1$
$f(x) = \frac{1}{x}$	$f(x) = \frac{1}{x}$

Target Transformation

```
def transform( var ):
        return np.log( 1 + var )

def inverse_transform( var ):
        return np.exp( var ) - 1

model.fit( train,   transform( y_train ) )

ypred = inverse_transform(  model.predict( test )  )
```

Target Transformation

Allstate
You're in good hands.

Allstate Claims Severity

How severe is an insurance claim?

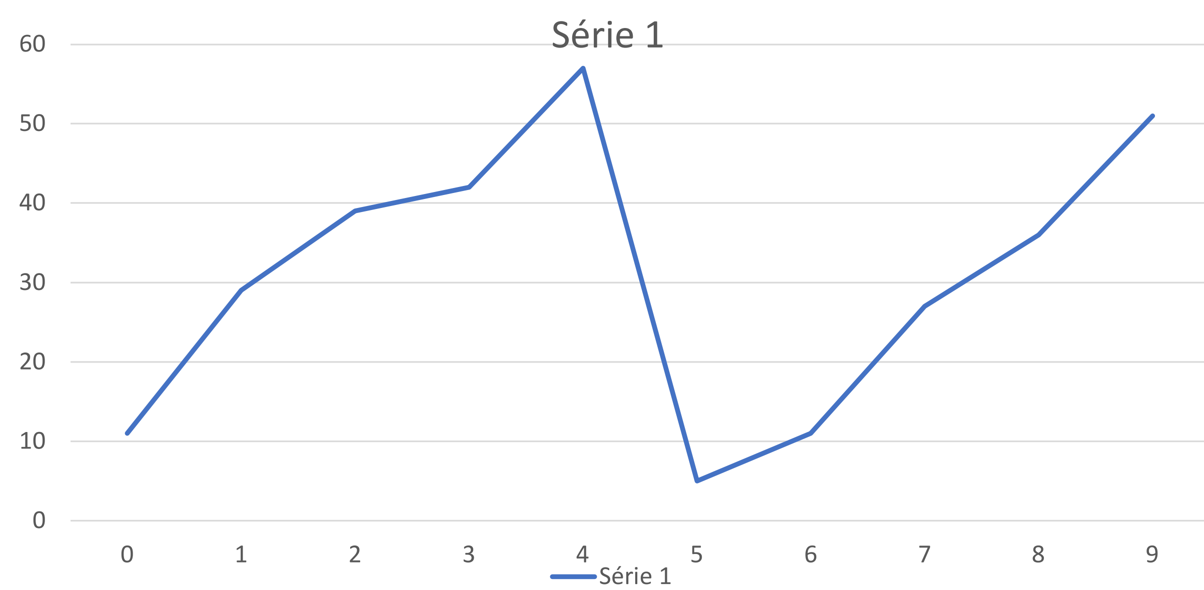
3,052 teams · 3 years ago

- $y^{1/2}$
- $y^{1/4}$
- $y^{1/8}$
- $\log(y)$
- $\log(y+100)$
- $\log(y+200)$
- $\log(y+400)$
- $10/y$

#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	▲3	Bishwarup B			1109.70772	111	3y
2	▲7	Alexey Noskov			1110.01363	91	3y
3	—	Faron			1110.04733	121	3y
4	▼2	Zach			1110.24415	205	3y
5	▲6	Eureka			1110.30015	64	3y
6	▲4	Turnin'			1110.51398	114	3y
7	▼6	BR On Vacation BR			1110.61167	125	3y

Features for Time Series

TimeStamp	Value
0	11
1	29
2	39
3	42
4	57
5	5
6	11
7	27
8	36
9	51



Features for Time Series

TimeStamp	Target	Lag 1	Lag 2
0	11	-	-
1	29	11	-
2	39	29	11
3	42	39	29
4	57	42	39
5	5	57	42
6	11	5	57
7	27	11	5
8	36	27	11
9	51	36	27

$$LagN(t) = Target(\, t - N \,)$$

kaggle DAYS

Features for Time Series

Trainset

TimeStamp	Target	Lag 1	Lag 2
0	11	-	-
1	29	11	-
2	39	29	11
3	42	39	29
4	57	42	39
5	5	57	42

Testset

6	-	5	57
7	-	-	5
8	-	-	-
9	-	-	-

Features for Time Series

Trainset

TimeStamp	Target	Lag 4	Lag 5
0	11	-	-
1	29	-	-
2	39	-	-
3	42	-	-
4	57	11	-
5	5	29	11

Testset

6	-	39	29
7	-	42	39
8	-	57	42
9	-	5	57

Features for Time Series

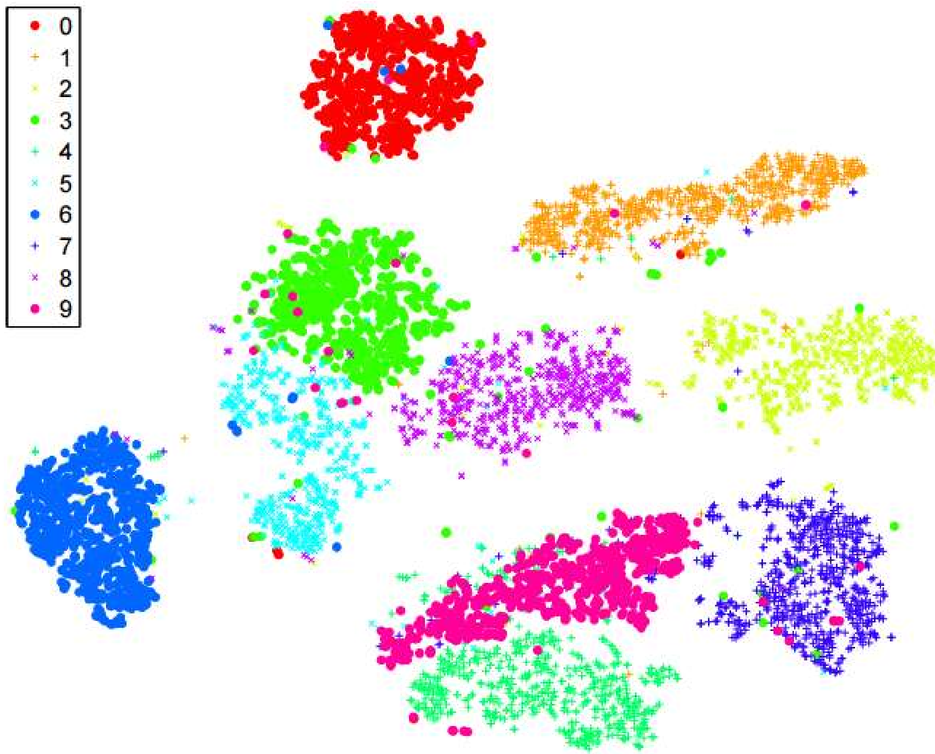
```
df = pd.DataFrame( {
     'ts': np.arange( df.shape[0] ),
    'target':np.array([11,29,39,42,57,5,11,27,36,51])}
)
df['lag1'] = df['target'].shift(1)
df['lag2'] = df['target'].shift(2)
df['hist_mean'] = df['target'].shift(1).rolling(window=9999, min_periods=1 ).mean()
df
```

	ts	target	lag1	lag2	hist_mean
0	0	11	NaN	NaN	NaN
1	1	29	11.0	NaN	11.000000
2	2	39	29.0	11.0	20.000000
3	3	42	39.0	29.0	26.333333
4	4	57	42.0	39.0	30.250000
5	5	5	57.0	42.0	35.600000
6	6	11	5.0	57.0	30.500000
7	7	27	11.0	5.0	27.714286
8	8	36	27.0	11.0	27.625000
9	9	51	36.0	27.0	28.555556

kaggle DAYS

Dimension Reduction

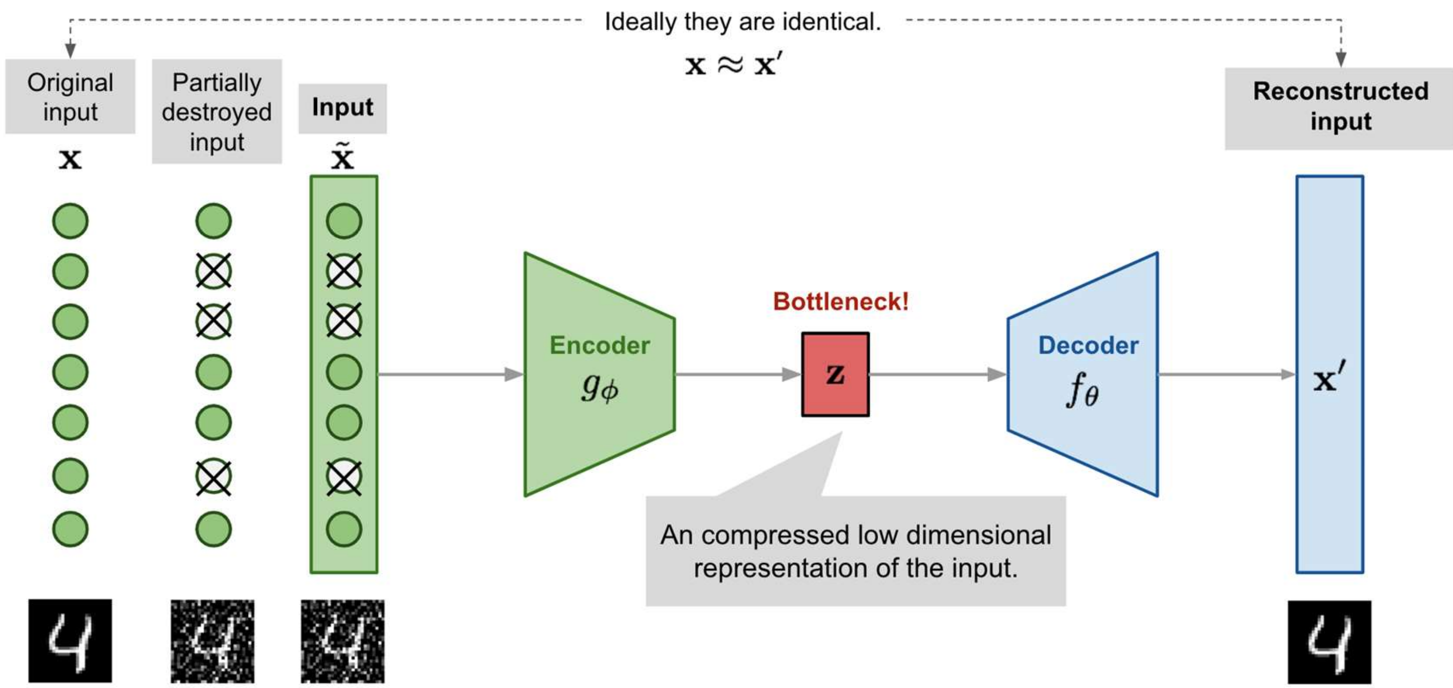
- PCA
- LDA
- SVD
- t-SNE
- Nearest Neighbor
- Autoencoder



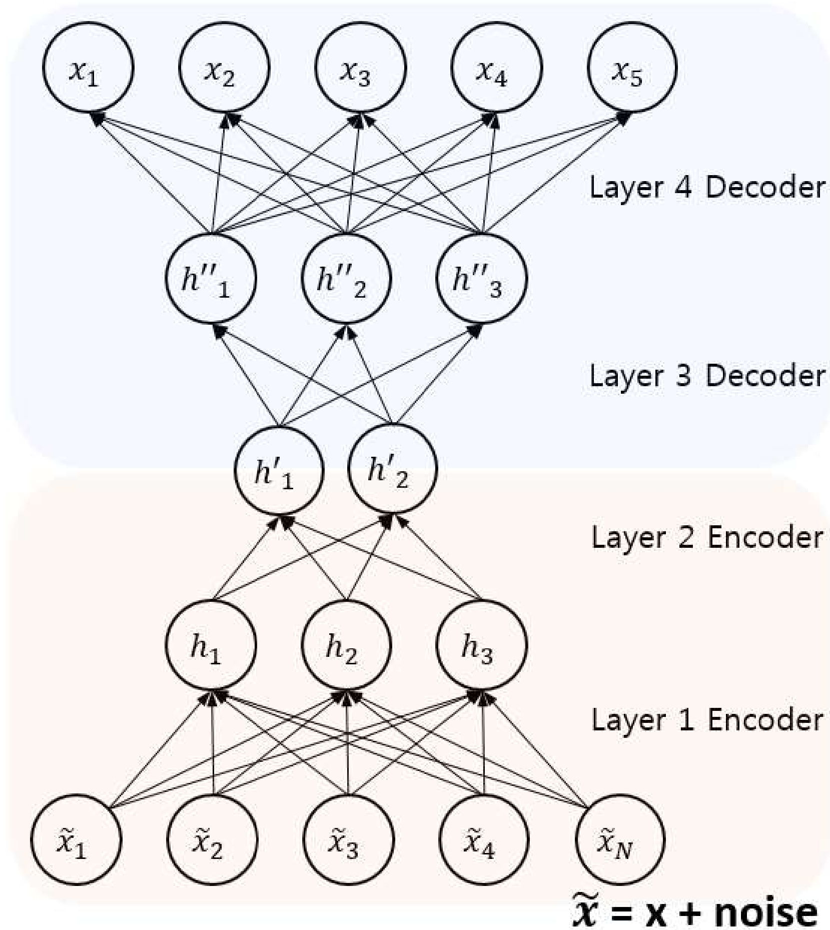
t-SNE on MNIST dataset

Denoising AutoEncoder

- It's a Deep Learning architecture to remove noise from data.



Denoising AutoEncoder



Denoising AutoEncoder

- How to add noise to the input ?
  - Random row and columns permutation in input data. Keep output.

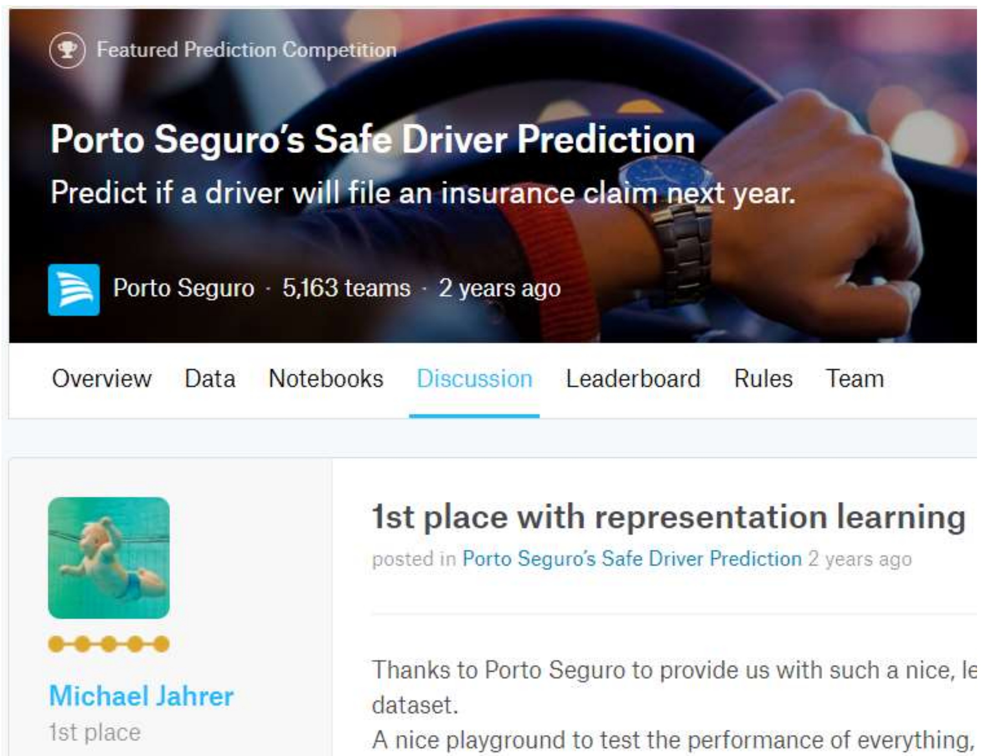
Y

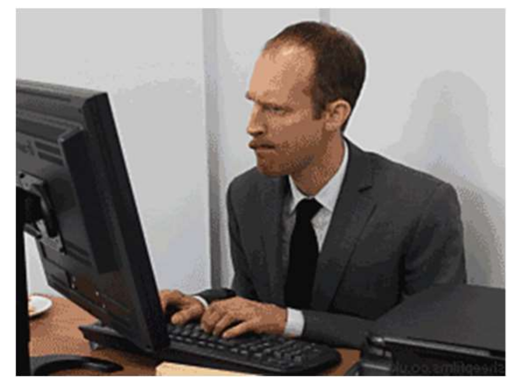
Var 1	Var 2
1	0
4	7
2	0
0	6
5	8
8	2

X

Var 1	Var 2
1	0
5	8
2	0
0	6
4	7
8	2

Denoising AutoEncoder





Clustering

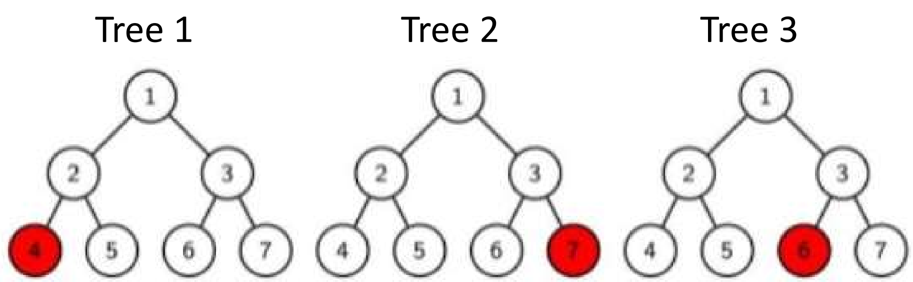
- K-means
- Affinity Propagation
- Mean-shift
- Spectral Clustering
- DBSCAN
- Agglomerative Clustering
- OPTICS
- Birch

GBDT Tree Leaves

```
predict(X, raw_score=False, num_iteration=None, pred_leaf=False, pred_contrib=False, **kwargs)    [source]
```

Return the predicted value for each sample.

- Return the index of the Leaf for every tree in the fit:
  - Use the index as a categorical feature in a linear model.



Feature set for this row: [4, 7, 6]

GBDT Tree Leaves

- Fit a LightGBM model for 100 rounds (trees) using parameters num_leaves=10.

- Leaf Predict function returns a matrix of shape: (Nrows x 100), each column have a leave index ranging from 0 to 9.

NLP – Natural Language Processing

1. Bag-of-Words / NGRAMS
2. Tf-Idf
3. Transfer Learning using DL (word embedding)
4. Double translation (removes noise/add)

   Eg: translate from Spanish to English then back to Spanish

Feature Selection

- Computer power and Time available
- Classical Algorithms: Forward Selection, Backward Elimination, Recursive Feature Elimination (sklearn RFE)

Feature Selection

- Use GBDT Feature Importance Information:
  1. Build a LightGBM model using only train fold and compute feature importance.
  2. Drop features with importance below certain threshold and train the model again.
  3. Use that model to predict the validation/test set.

Random Noise Feature Importance

- Add one or more features using a random values.
  - Eg. train[“rand”] = np.random.randn(train.shape[0])
- Train a GBDT or a Linear model and compute the feature importance.

Name	Importance
var 1	1000
var 2	100
rand	50
var 3	10

kaggle DAYS

Leave One Feature Out (LOFO)

- Remove one feature each time and compute the difference of the new metric and the initial metric using all features.

- How to remove a feature?
  1. Hard write a fixed value
  2. Hard write the mean average
  3. Shuffling

Adversarial Validation

- Build a model to classify if a row comes from the train set or from test set.
- How:
  - Concatenate train+test.
  - Target variable 0 to all train rows and 1 to all test rows.
  - Build a model.
  - If AUC metric >> 0.5 means that its easy for the model to distinguish between train and test rows.
  - Compute feature importance and have an idea which feature is responsible for the bias between train and test set.

Thank You

Find me @

www.linkedin.com/in/giba1

https://www.kaggle.com/titericz