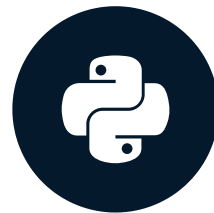


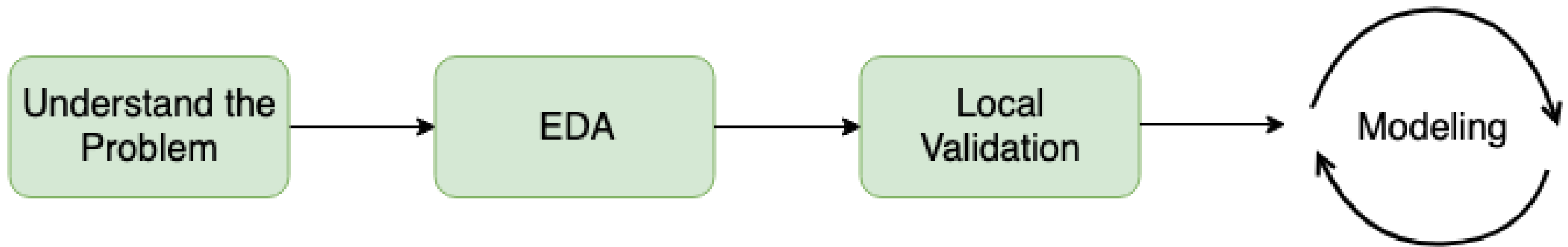
# Feature engineering

WINNING A KAGGLE COMPETITION IN PYTHON

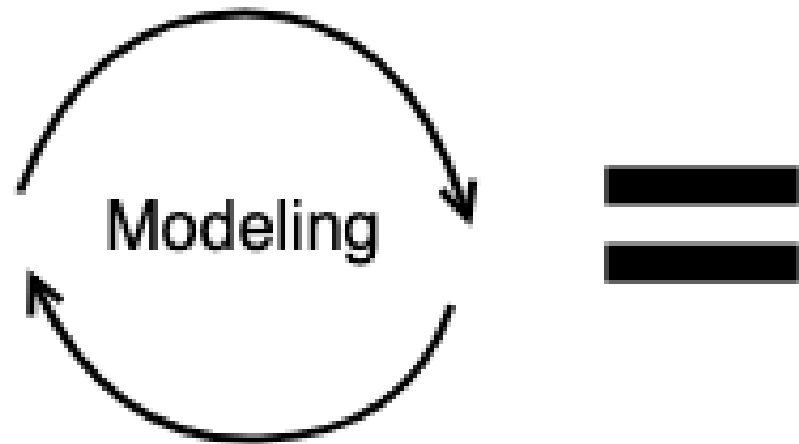


**Yauhen Babakhin**  
Kaggle Grandmaster

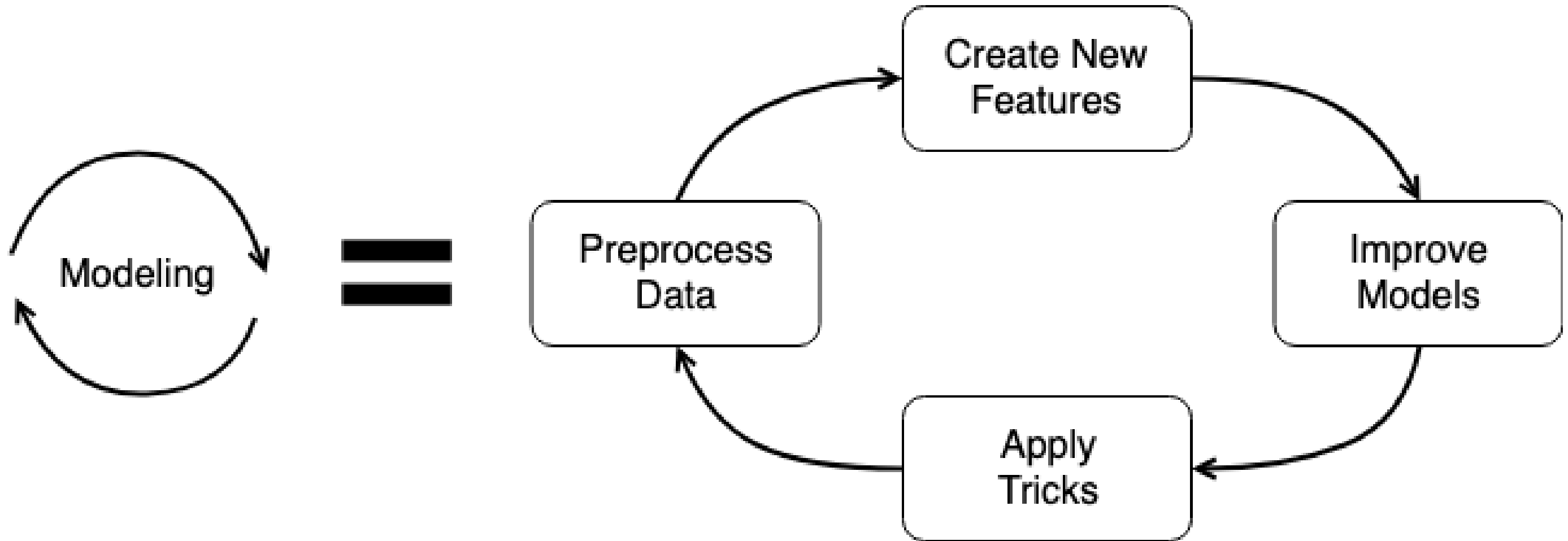
# Solution workflow



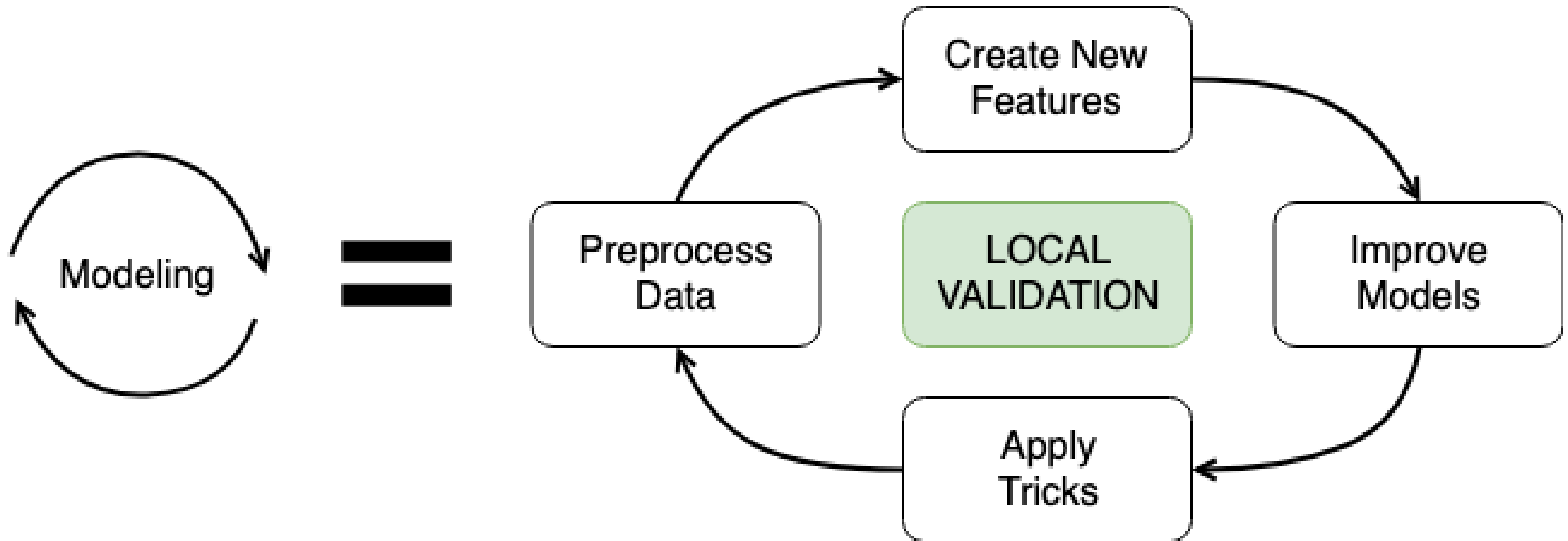
# Modeling stage



# Modeling stage

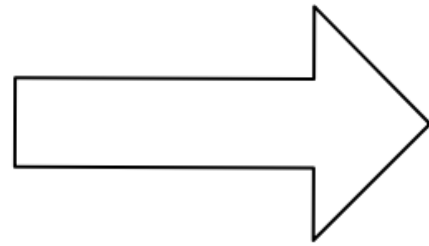


# Modeling stage



# Feature engineering

Initial data

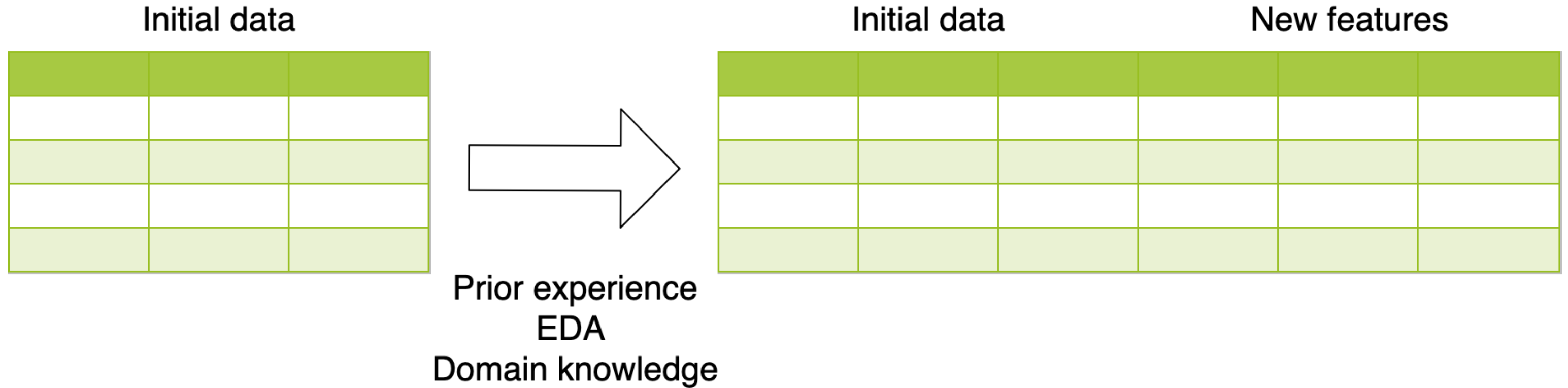


Initial data



New features

# Feature engineering



# Feature types

- Numerical
- Categorical
- Datetime
- Coordinates
- Text
- Images



# Creating features

```
# Concatenate the train and test data  
data = pd.concat([train, test])
```

```
# Create new features for the data DataFrame...
```

```
# Get the train and test back  
train = data[data.id.isin(train.id)]  
test = data[data.id.isin(test.id)]
```

# Arithmetical features

```
# Two sigma connect competition  
two_sigma.head(1)
```

```
   id  bathrooms  bedrooms  price  interest_level  
0   10         1.5         3    3000         medium
```

```
# Arithmetical features  
two_sigma['price_per_bedroom'] = two_sigma.price / two_sigma.bedrooms  
two_sigma['rooms_number'] = two_sigma.bedrooms + two_sigma.bathrooms
```

# Datetime features

```
# Demand forecasting challenge  
dem.head(1)
```

```
   id      date store  item  sales  
0  100000 2017-12-01    1     1    19
```

```
# Convert date to the datetime object  
dem['date'] = pd.to_datetime(dem['date'])
```

# Datetime features

```
# Year features
dem['year'] = dem['date'].dt.year

# Month features
dem['month'] = dem['date'].dt.month

# Week features
dem['week'] = dem['date'].dt.weekofyear
```

```
# Day features
dem['dayofyear'] = dem['date'].dt.dayofyear
dem['dayofmonth'] = dem['date'].dt.day
dem['dayofweek'] = dem['date'].dt.dayofweek
```

date	year	month	week
2017-12-01	2017	12	48
2017-12-02	2017	12	48
2017-12-03	2017	12	48
2017-12-04	2017	12	49

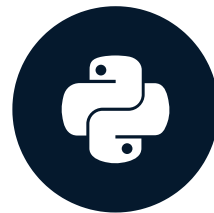
date	dayofyear	dayofmonth	dayofweek
2017-12-01	335	1	4
2017-12-02	336	2	5
2017-12-03	337	3	6
2017-12-04	338	4	0

# Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON

# Categorical features

WINNING A KAGGLE COMPETITION IN PYTHON



**Yauhen Babakhin**  
Kaggle Grandmaster

# Label encoding

ID	Categorical feature
1	A
2	B
3	C
4	A
5	D
6	A

ID	Label-encoded
1	0
2	1
3	2
4	0
5	3
6	0

# Label encoding

```
# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder
# Create a LabelEncoder object
le = LabelEncoder()
# Encode a categorical feature
df['cat_encoded'] = le.fit_transform(df['cat'])
```

	ID	cat	cat_encoded
0	1	A	0
1	2	B	1
2	3	C	2
3	4	A	0



# One-Hot encoding

ID	Categorical feature
1	A
2	B
3	C
4	A
5	D
6	A

ID	Cat == A	Cat == B	Cat == C	Cat == D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	1	0	0	0
5	0	0	0	1
6	1	0	0	0

# One-Hot encoding

```
# Create One-Hot encoded features
ohe = pd.get_dummies(df['cat'], prefix='ohe_cat')
# Drop the initial feature
df.drop('cat', axis=1, inplace=True)
# Concatenate OHE features to the dataframe
df = pd.concat([df, ohe], axis=1)
```

	ID	ohe_cat_A	ohe_cat_B	ohe_cat_C	ohe_cat_D
0	1	1	0	0	0
1	2	0	1	0	0
2	3	0	0	1	0
3	4	1	0	0	0

# Binary Features

```
# DataFrame with a binary feature  
binary_feature
```

```
   binary_feat  
0      Yes  
1      No
```

```
le = LabelEncoder()  
binary_feature['binary_encoded'] = le.fit_transform(binary_feature['binary_feat'])
```

```
   binary_feat  binary_encoded  
0      Yes          1  
1      No           0
```

# Other encoding approaches

- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder
- Hashing
- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- Target Encoder
- Weight of Evidence

# Other encoding approaches

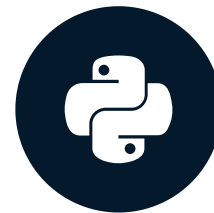
- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder
- Hashing
- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- **Target Encoder**
- Weight of Evidence

# Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON

# Target encoding

WINNING A KAGGLE COMPETITION IN PYTHON



**Yauhen Babakhin**  
Kaggle Grandmaster

# High cardinality categorical features

- Label encoder provides distinct number for each category
- One-hot encoder creates new feature for each category value
- **Target encoding to the rescue!**



# Mean target encoding

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

Test ID	Categorical	Target
10	A	?
11	A	?
12	B	?
13	A	?

# Mean target encoding

1. Calculate mean on the train, apply to the test
2. Split train into  $K$  folds. Calculate mean on  $(K-1)$  folds, apply to the  $K$ -th fold
3. Add mean target encoded feature to the model

# Calculate mean on the train

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

# Calculate mean on the train

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

# Calculate mean on the train

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

# Test encoding

Test ID	Categorical	Target	Mean encoded
10	A	?	0.66
11	A	?	0.66
12	B	?	0.25
13	A	?	0.66

# Train encoding using out-of-fold

Train ID	Categorical	Target	Fold
1	A	1	1
2	B	0	1
3	B	0	1
4	A	1	1
5	B	0	2
6	A	0	2
7	B	1	2

# Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	
2	B	0	1	
3	B	0	1	
4	A	1	1	
5	B	0	2	
6	A	0	2	
7	B	1	2	



# Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	0
2	B	0	1	0.5
3	B	0	1	0.5
4	A	1	1	0
5	B	0	2	
6	A	0	2	
7	B	1	2	

# Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	0
2	B	0	1	0.5
3	B	0	1	0.5
4	A	1	1	0
5	B	0	2	
6	A	0	2	
7	B	1	2	

# Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	0
2	B	0	1	0.5
3	B	0	1	0.5
4	A	1	1	0
5	B	0	2	0
6	A	0	2	1
7	B	1	2	0

# Practical guides

# Practical guides

## Smoothing

$$\text{mean\_enc}_i = \frac{\text{target\_sum}_i}{n_i}$$

$$\text{smoothed\_mean\_enc}_i = \frac{\text{target\_sum}_i + \alpha * \text{global\_mean}}{n_i + \alpha}$$

$$\alpha \in [5; 10]$$

# Practical guides

## Smoothing

$$\text{mean\_enc}_i = \frac{\text{target\_sum}_i}{n_i}$$

$$\text{smoothed\_mean\_enc}_i = \frac{\text{target\_sum}_i + \alpha * \text{global\_mean}}{n_i + \alpha}$$

$$\alpha \in [5; 10]$$

## New categories

- Fill new categories in the test data with a *global\_mean*

# Practical guides

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	0
5	B	1

Test ID	Categorical	Target	Mean encoded
10	A	?	0.43
11	B	?	0.38
12	C	?	0.40

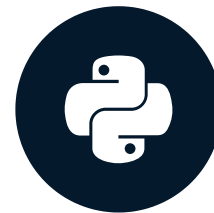
# Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON



# Missing data

WINNING A KAGGLE COMPETITION IN PYTHON



**Yauhen Babakhin**  
Kaggle Grandmaster

# Missing data

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	NaN	1
5	NaN	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	NaN	1
5	NaN	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation
- Constant value imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	<b>4.72</b>	1
5	NaN	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation
- Constant value imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	<b>-999</b>	1
5	NaN	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation
- Constant value imputation

## Categorical data

- Most frequent category imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	-999	1
5	NaN	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation
- Constant value imputation

## Categorical data

- Most frequent category imputation
- New category imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	-999	1
5	A	2.6	0
6	A	5.3	0

# Impute missing data

## Numerical data

- Mean/median imputation
- Constant value imputation

## Categorical data

- Most frequent category imputation
- New category imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	-999	1
5	<b>MISS</b>	2.6	0
6	A	5.3	0



# Find missing data

```
df.isnull().head(1)
```

```
   ID  cat  num  target
0  False False False  False
```

```
df.isnull().sum()
```

```
ID      0
cat     1
num     1
target  0
```

# Numerical missing data

```
# Import SimpleImputer
from sklearn.impute import SimpleImputer
# Different types of imputers
mean_imputer = SimpleImputer(strategy='mean')
constant_imputer = SimpleImputer(strategy='constant', fill_value=-999)
# Imputation
df[['num']] = mean_imputer.fit_transform(df[['num']])
```

# Categorical missing data

```
# Import SimpleImputer
from sklearn.impute import SimpleImputer

# Different types of imputers
frequent_imputer = SimpleImputer(strategy='most_frequent')
constant_imputer = SimpleImputer(strategy='constant', fill_value='MISS')

# Imputation
df[['cat']] = constant_imputer.fit_transform(df[['cat']])
```

# Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON