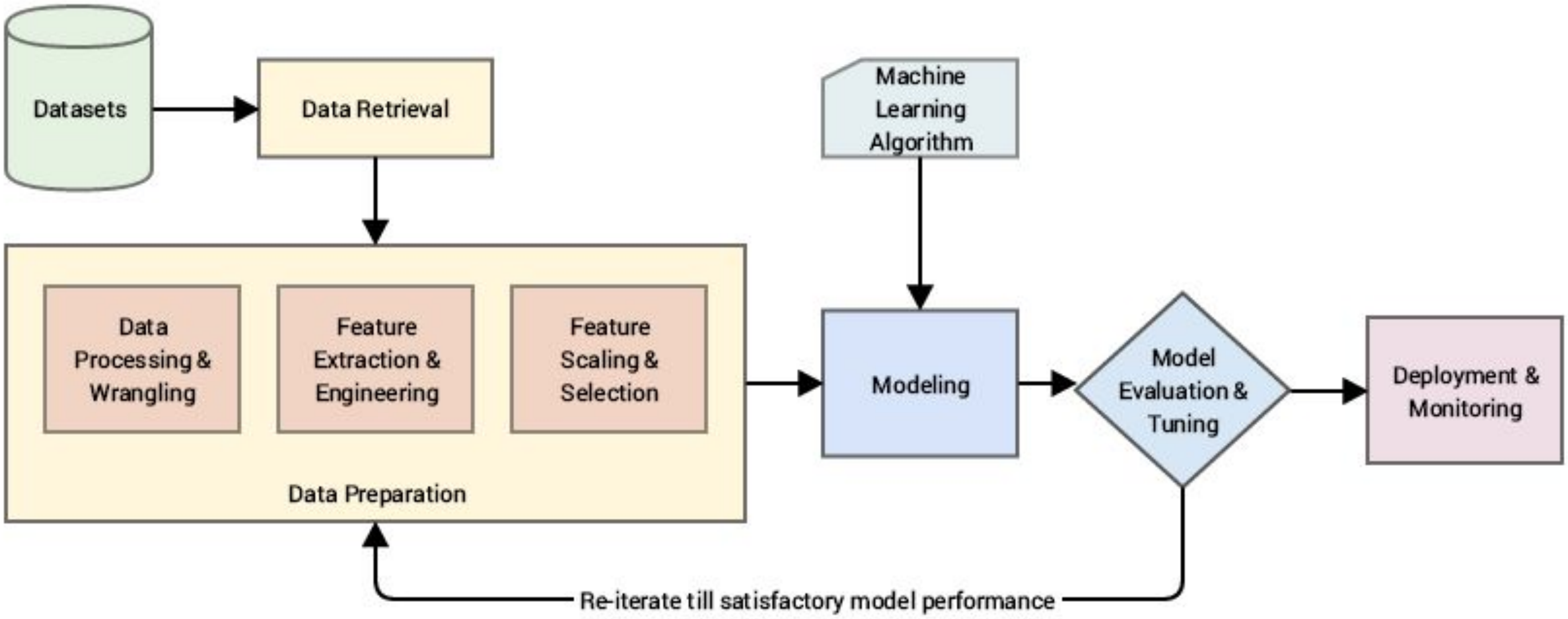
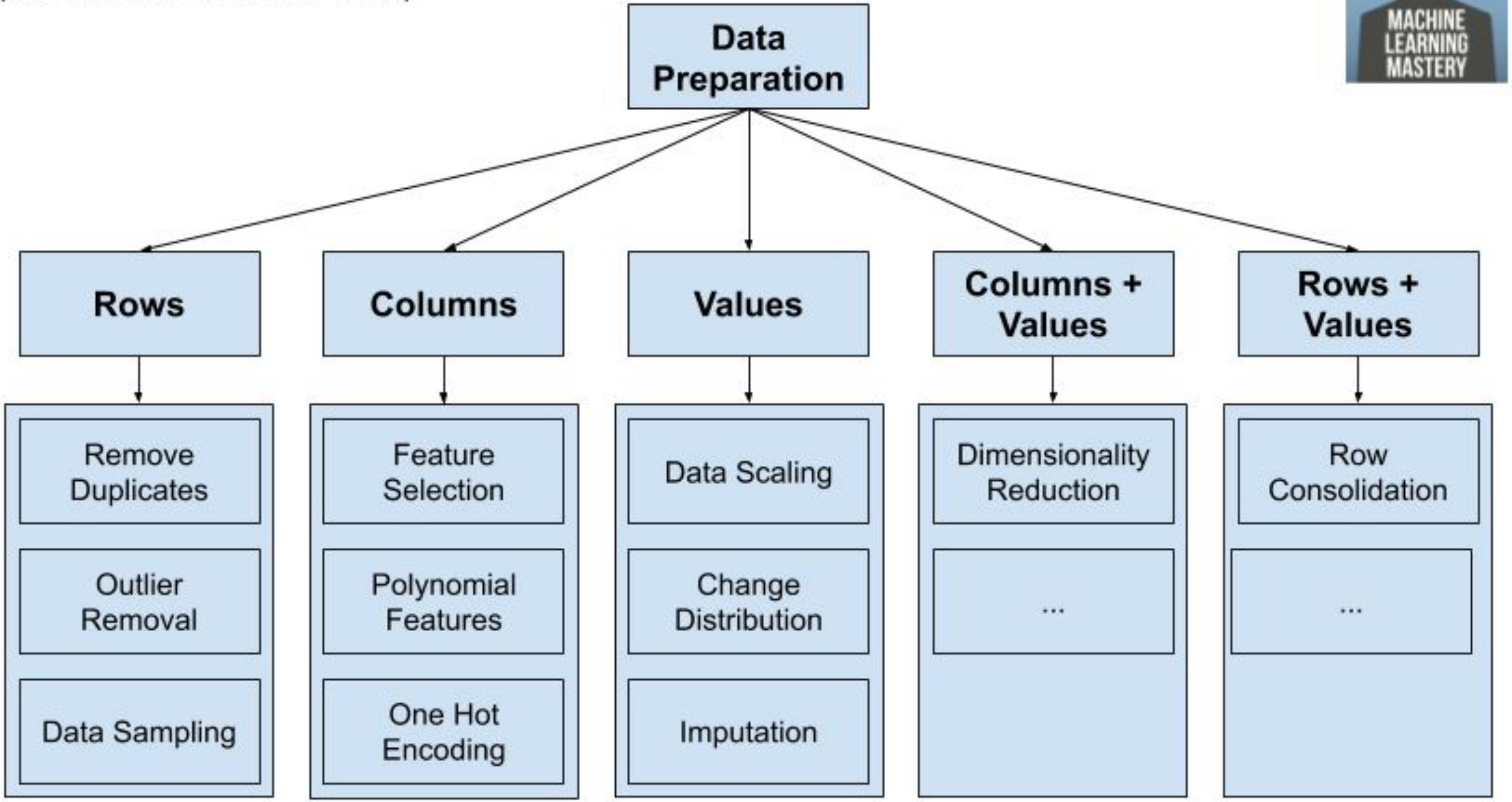
Feature engineering


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

Data Preparation Framework
(*for structured/tabular data*)



Заполнение пропусков

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	Boston Celtics	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
11	Isaiah Thomas	Boston Celtics	4.0	PG	27.0	5-9	185.0	Washington	6912869.0

Вариант 1

```
nba["College"].fillna("No College", inplace =  True)
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	No College	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	No College	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0

Вариант 2

```
nba["College"].fillna( method ='ffill', inplace =  True)
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	Georgia State	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	Georgia State	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0

Замена значений колонок

Out[3]:

	ID	Address	City	State	Country	Name	Employees	commissioned
0	1	3666 21st St	San Francisco	CA 94114	USA	Madeira	8	yes
1	2	735 Dolores St	San Francisco	CA 94119	USA	Bready Shop	15	no
2	3	332 Hill St	San Francisco	California 94114	USA	Super River	25	no
3	4	3995 23rd St	San Francisco	CA 94114	USA	Ben's Shop	10	yes
4	5	1056 Sanchez St	San Francisco	California	USA	Sanchez	12	yes
5	6	551 Alvarado St	San Francisco	CA 94114	USA	Richvalley	20	no

Замена на True/False (1/0)

```
df['commissioned'] = df['commissioned'].map({'yes':True ,'no':False})
```

	ID	Address	City	State	Country	Name	Employees	commissioned
0	1	3666 21st St	San Francisco	CA 94114	USA	Madeira	8	True
1	2	735 Dolores St	San Francisco	CA 94119	USA	Bready Shop	15	False
2	3	332 Hill St	San Francisco	California 94114	USA	Super River	25	False
3	4	3995 23rd St	San Francisco	CA 94114	USA	Ben's Shop	10	True
4	5	1056 Sanchez St	San Francisco	California	USA	Sanchez	12	True
5	6	551 Alvarado St	San Francisco	CA 94114	USA	Richvalley	20	False

Преобразование в категориальный признак

```
In [6]: df = pd.DataFrame({'value': np.random.randint(0, 100, 20)})

In [7]: labels = ["{0} - {1}".format(i, i + 9) for i in range(0, 100, 10)]

In [8]: df['group'] = pd.cut(df.value, range(0, 105, 10), right=False, labels=labels)

In [9]: df.head(10)
Out[9]:
   value    group
0     65  60 - 69
1     49  40 - 49
2     56  50 - 59
3     43  40 - 49
4     43  40 - 49
5     91  90 - 99
6     32  30 - 39
7     87  80 - 89
8     36  30 - 39
9      8    0 - 9
```

LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
df['code'] = LE.fit_transform(df['cc'])

print(df)

   cc  temp  code
0  US  37.0     2
1  CA  12.0     1
2  US  35.0     2
3  AU  20.0     0
```

lambda

```
df[['cc']] = df[['cc']].apply(lambda
col:pd.Categorical(col).codes).replace(-1,np.nan)
```

One hot encoding

```
y = pd.get_dummies(df.Pet, prefix='Pet')
```

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

BinaryEncoder

```
cat_df_flights_ce = cat_df_flights.copy()

import category_encoders as ce

encoder = ce.BinaryEncoder(cols=['carrier'])
df_binary = encoder.fit_transform(cat_df_flights_ce)

df_binary.head()
```

	carrier_0	carrier_1	carrier_2	carrier_3	tailnum	origin	dest
0	0	0	0	0	N508AS	PDX	ANC
1	0	0	0	1	N195UW	SEA	CLT
2	0	0	1	0	N37422	PDX	IAH
3	0	0	0	1	N547UW	PDX	CLT
4	0	0	0	0	N762AS	SEA	ANC

BackwardDifferenceEncoder

```
encoder = ce.BackwardDifferenceEncoder(cols=['carrier'])
df_bd = encoder.fit_transform(cat_df_flights_ce)


df_bd.head()
```

	col_carrier_0	col_carrier_1	col_carrier_2	col_carrier_3	col_carrier_4	col_carrier_5	col_carrier_6	col_carrier_7	col_carrier_8	col_carrier_9	col_carrier_10	col_tailnum	col_origin	col_dest
0	1.0	-0.909091	-0.818182	-0.727273	-0.636364	-0.545455	-0.454545	-0.363636	-0.272727	-0.181818	-0.090909	N508AS	PDX	ANC
1	1.0	0.090909	-0.818182	-0.727273	-0.636364	-0.545455	-0.454545	-0.363636	-0.272727	-0.181818	-0.090909	N195UW	SEA	CLT
2	1.0	0.090909	0.181818	-0.727273	-0.636364	-0.545455	-0.454545	-0.363636	-0.272727	-0.181818	-0.090909	N37422	PDX	IAH
3	1.0	0.090909	-0.818182	-0.727273	-0.636364	-0.545455	-0.454545	-0.363636	-0.272727	-0.181818	-0.090909	N547UW	PDX	CLT
4	1.0	-0.909091	-0.818182	-0.727273	-0.636364	-0.545455	-0.454545	-0.363636	-0.272727	-0.181818	-0.090909	N762AS	SEA	ANC

Комбинирование признаков

Combine using a simple function that chooses the smaller column.

```
>>> df1 = pd.DataFrame({'A': [0, 0], 'B': [4, 4]})
>>> df2 = pd.DataFrame({'A': [1, 1], 'B': [3, 3]})
>>> take_smaller = lambda s1, s2: s1 if s1.sum() < s2.sum() else s2
>>> df1.combine(df2, take_smaller)
   A  B
0  0  3
1  0  3
```

Example using a true element-wise combine function.

```
>>> df1 = pd.DataFrame({'A': [5, 0], 'B': [2, 4]})
>>> df2 = pd.DataFrame({'A': [1, 1], 'B': [3, 3]})
>>> df1.combine(df2, np.minimum)
   A  B
0  1  2
1  0  3
```

Using *fill_value* fills Nones prior to passing the column to the merge function.

```
>>> df1 = pd.DataFrame({'A': [0, 0], 'B': [None, 4]})
>>> df2 = pd.DataFrame({'A': [1, 1], 'B': [3, 3]})
>>> df1.combine(df2, take_smaller, fill_value=-5)
   A    B
0  0 -5.0
1  0  4.0
```

Создание нового признака с помощью функции

```python
# using apply function to create a new column
df['Discounted_Price'] = df.apply(lambda row: row.Cost -
                                  (row.Cost * 0.1), axis = 1)

# Print the DataFrame after addition
# of new column
print(df)
```

Output:

	Date	Event	Cost	Discounted_Price
0	10/2/2011	Music	10000	9000.0
1	11/2/2011	Poetry	5000	4500.0
2	12/2/2011	Theatre	15000	13500.0
3	13/2/2011	Comedy	2000	1800.0

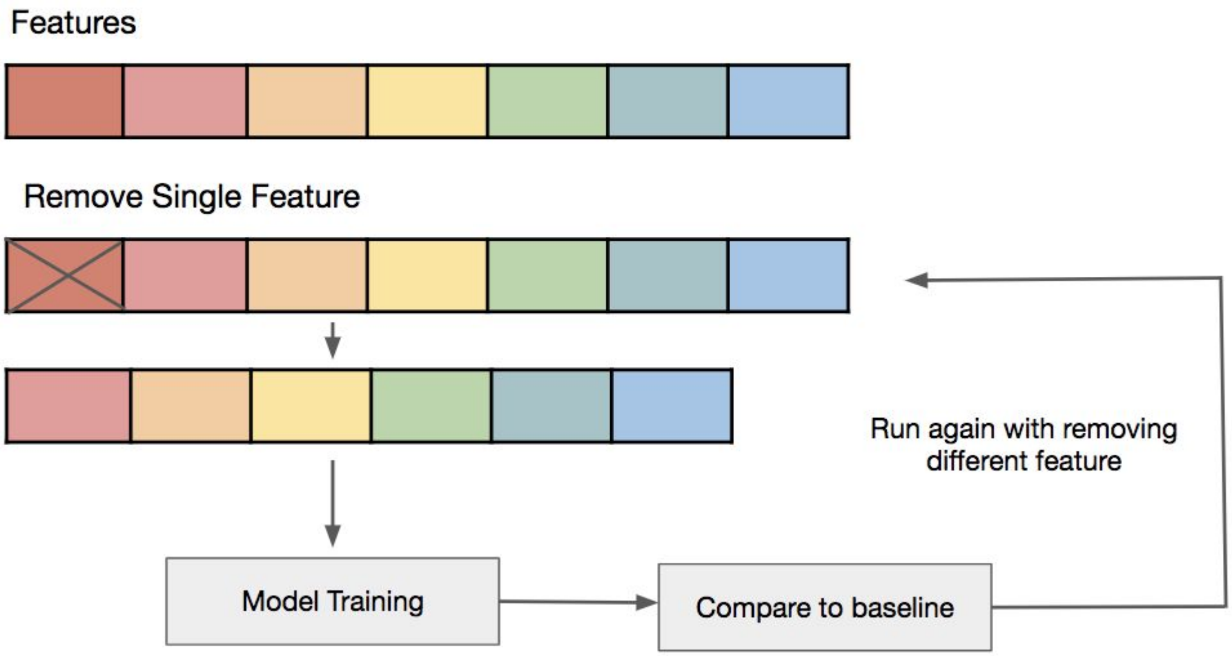
Автоматизация FeatureTools

name	type	description
num_true	aggregation	Finds the number of 'True' values in a boolean.
percent_true	aggregation	Finds the percent of 'True' values in a boolean feature.
time_since_last	aggregation	Time since last related instance.
num_unique	aggregation	Returns the number of unique categorical variables.
avg_time_between	aggregation	Computes the average time between consecutive events.
all	aggregation	Test if all values are 'True'.
min	aggregation	Finds the minimum non-null value of a numeric feature.
mean	aggregation	Computes the average value of a numeric feature.
seconds	transform	Transform a Timedelta feature into the number of seconds.
second	transform	Transform a Datetime feature into the second.
and	transform	For two boolean values, determine if both values are 'True'.
month	transform	Transform a Datetime feature into the month.
cum_sum	transform	Calculates the sum of previous values of an instance for each value in a time-dependent entity.
percentile	transform	For each value of the base feature, determines the percentile in relation
time_since_previous	transform	Compute the time since the previous instance.
cum_min	transform	Calculates the min of previous values of an instance for each value in a time-dependent entity.

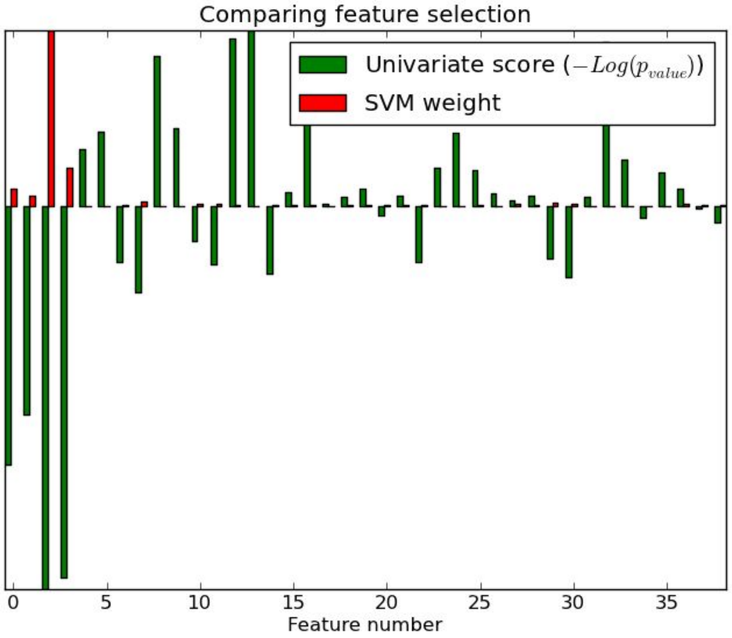
```
# Create new features using specified primitives

features, feature_names = ft.dfs(
entityset = es,
target_entity = 'clients',
agg_primitives = ['mean', 'max', 'percent_true',
'last'],
trans_primitives = ['years', 'month', 'subtract',
'divide'])
```

Отбор признаков

С обратной связью



По значению коэффициентов (для линейных моделей)




Q
A