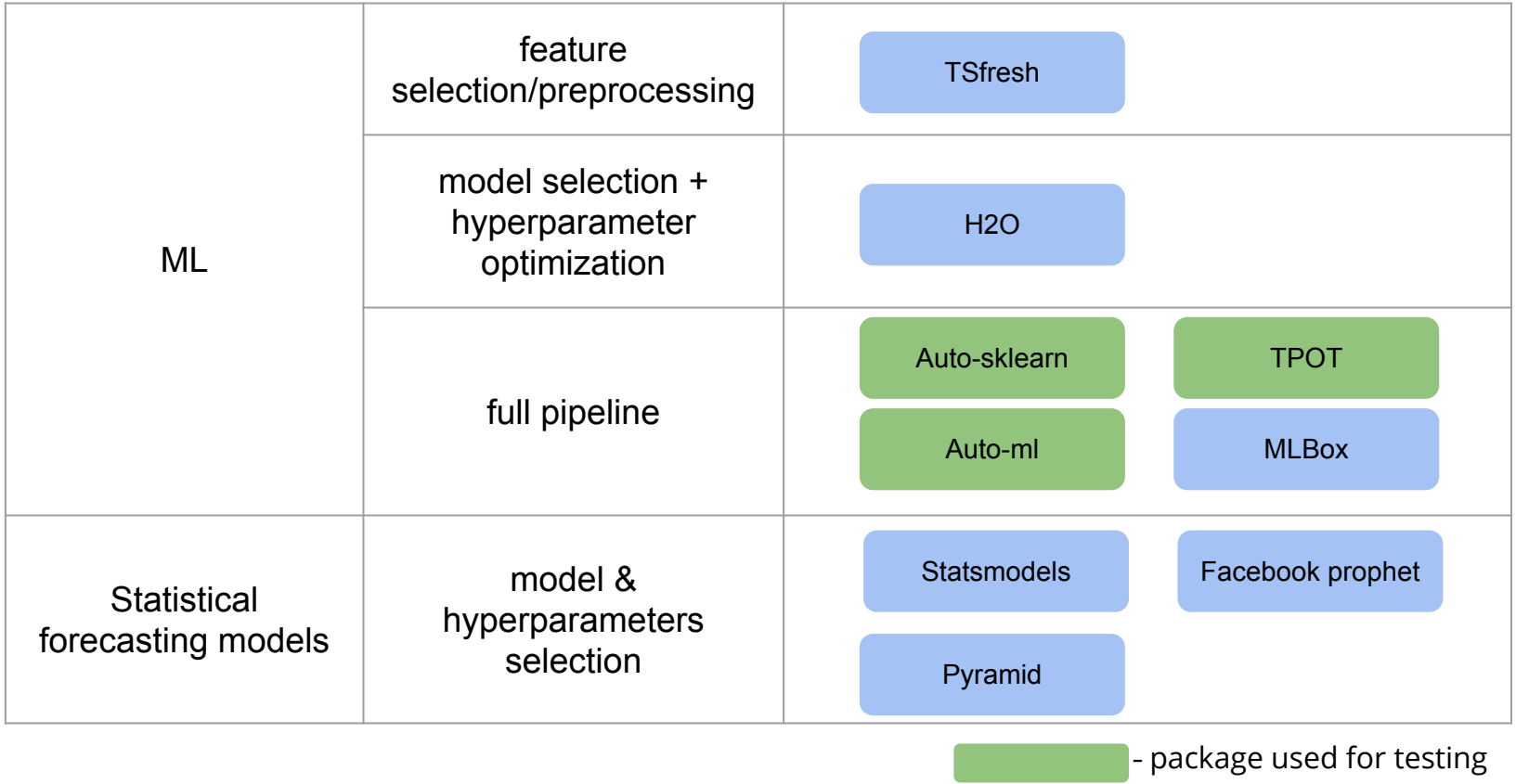
Testing auto-ml python packages

How to test auto-ml packages?

1. Run auto-ml on datasets with known benchmarks (kaggle, m4)
2. Do not use complex or specific data preprocessing (minimum of manual work)
3. Compare auto-ml packages with best solutions from benchmarks
4. Compare auto-ml packages between themselves

AutoML + stats python packages



Selected datasets for testing

№	Dataset name	Link	Problem description
1	Web Traffic Time Series Forecasting	https://www.kaggle.com/c/web-traffic-time-series-forecasting/data	Forecast future traffic to Wikipedia pages, многофакторная регрессия
2	**Rossmann Store Sales**	**https://www.kaggle.com/c/rossmann-store-sales**	**Forecast sales using store, promotion, and competitor data**
3	Global Energy Forecasting Competition 2012 - Wind Forecasting	https://www.kaggle.com/c/GEF2012-wind-forecasting	A wind power forecasting problem: predicting hourly power generation up to 48 hours ahead at 7 wind farms
4	Walmart Recruiting - Store Sales Forecasting	https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting#description	One challenge of modeling retail data is the need to make decisions based on limited history.
5	**Restaurant revenue prediction**	**https://www.kaggle.com/c/restaurant-revenue-prediction/submissions?sortBy=date&group=all&page=1**	**Predict sales for stores on the next periods of time**

Rossman Store Sales

https://www.kaggle.com/c/rossmann-store-sales

- Problem: Regression
- Train: 1017209 samples
- Test: 41088 samples
- Features:
  - 7 categorical
  - 4 numerical
- Score: Root Mean Square Percentage Error (less is better)

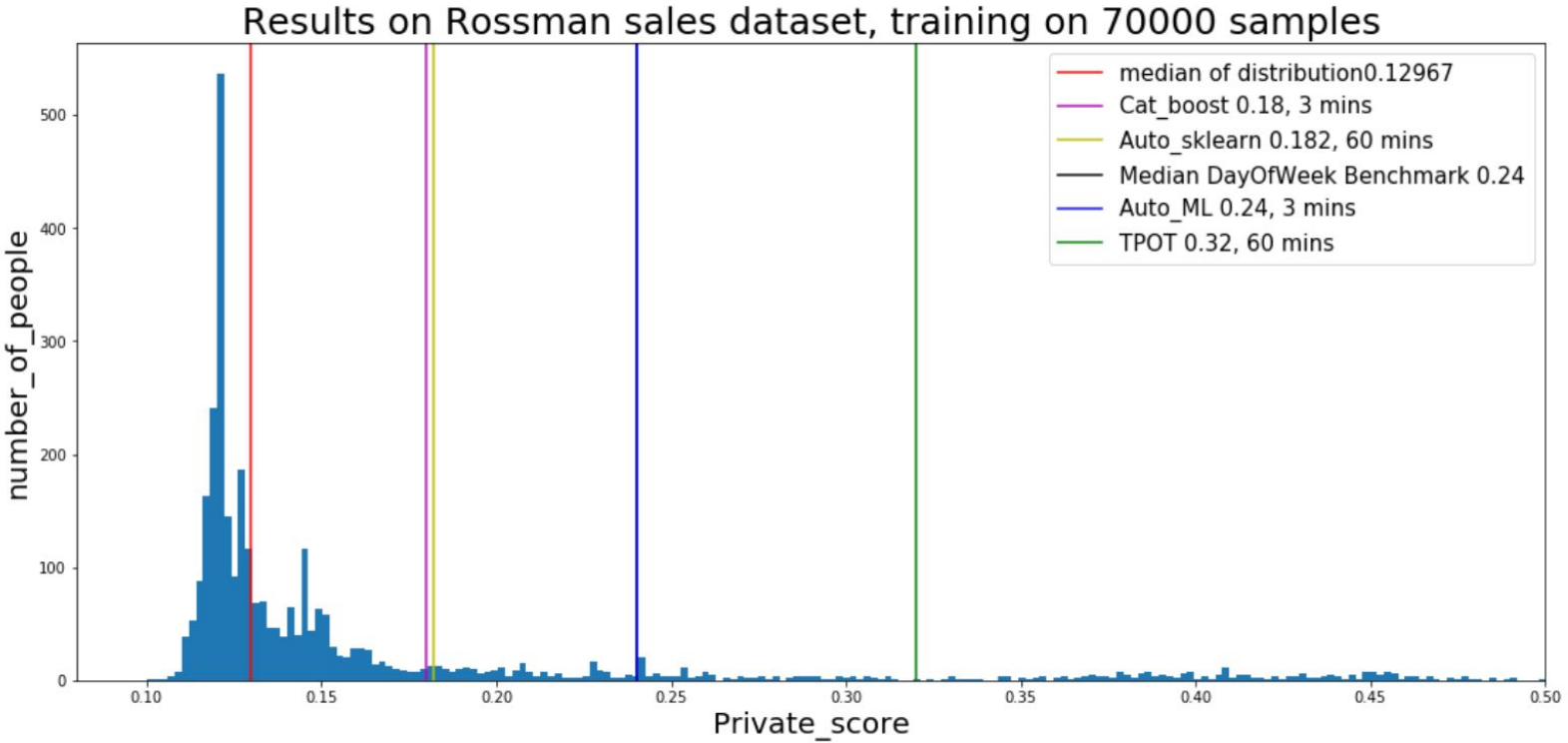
$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

	Store	DayOfWeek	Open	Promo	SchoolHoliday	Year	Month	HolidayBin	StoreType	Assortment	CompetitionDistance
0	1	4	1.0	1	0	2015	9	0	c	a	1270.0
1	3	4	1.0	1	0	2015	9	0	a	a	14130.0
2	7	4	1.0	1	0	2015	9	0	a	c	24000.0
3	8	4	1.0	1	0	2015	9	0	a	a	7520.0
4	9	4	1.0	1	0	2015	9	0	a	c	2030.0

Most of the fields are self-explanatory. The following are descriptions for those that aren't.

- **Store** - a unique Id for each store
- **Sales** - the turnover for any given day (this is what you are predicting) **[target]**
- **Open** - an indicator for whether the store was open: 0 = closed, 1 = open
- **Promo** - indicates whether a store is running a promo on that day
- **SchoolHoliday** - indicates if the (Store, Date) was affected by the closure of public schools
- **HolidayBin** - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. 1-holiday, 0 = None
- **StoreType** - differentiates between 4 different store models: a, b, c, d
- **Assortment** - describes an assortment level: a = basic, b = extra, c = extended
- **CompetitionDistance** - distance in meters to the nearest competitor store

AutoML and CatBoost results confidently overperform median score



Restaurant revenue prediction

https://www.kaggle.com/c/restaurant-revenue-prediction/submissions?sortBy=date&group=all&page=1

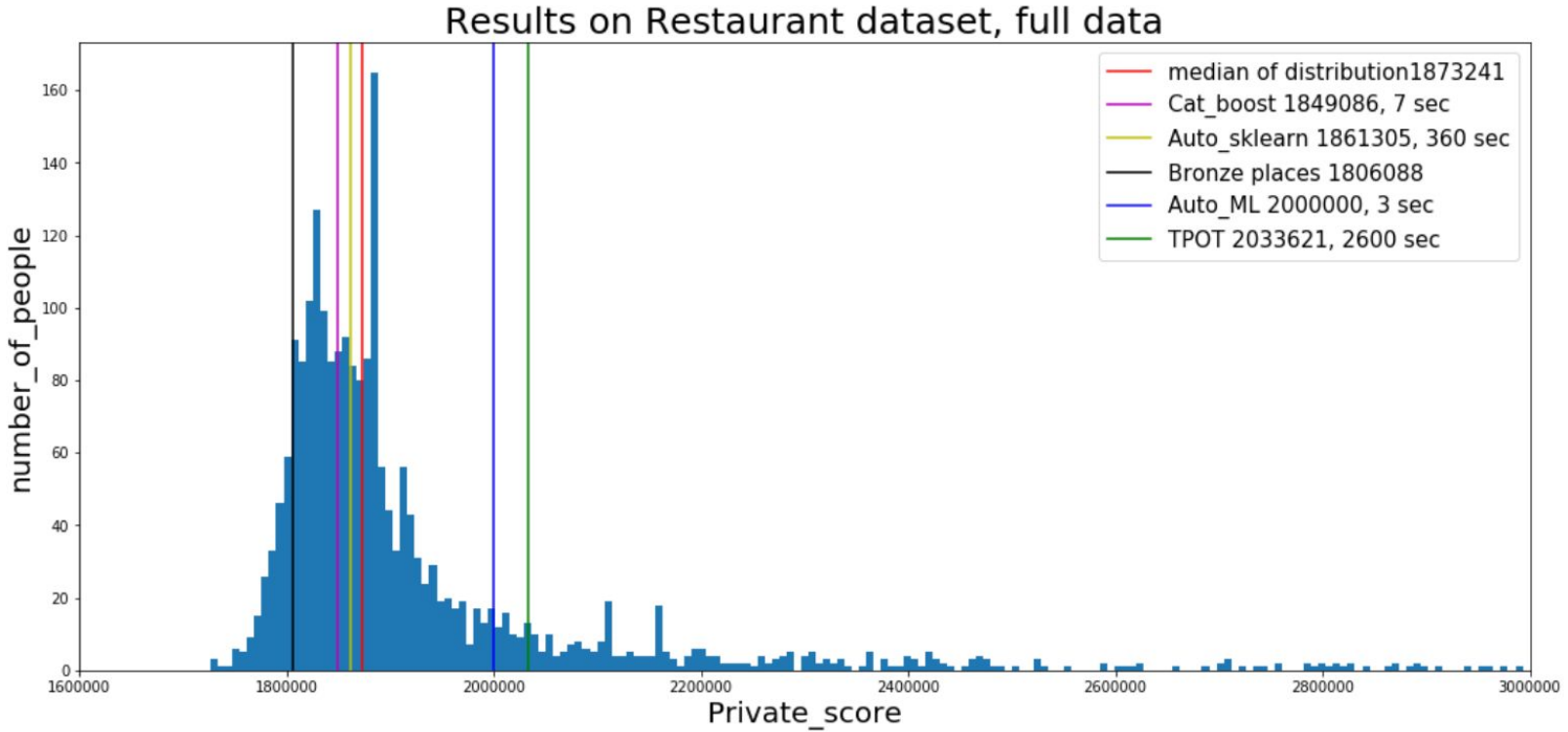
- Problem: Regression
- Train: 137 samples
- Test: 100000 samples
- Features:
  - 38 categorical
  - 4 numerical
- Score: Root Mean Square Percentage Error
  (less is better)

	Id	City	City Group	Type	P1	P2	P3	P4	P5	P6	...	P31	P32	P33	P34	P35	P36	P37	Day	Month	Year
0	0	İstanbul	Big Cities	IL	4	5.0	4.0	4.0	2	2	...	3	4	5	5	4	3	4	7	17	1999
1	1	Ankara	Big Cities	FC	4	5.0	4.0	4.0	1	2	...	0	0	0	0	0	0	0	2	14	2008
2	2	Diyarbakır	Other	IL	2	4.0	2.0	5.0	2	3	...	0	0	0	0	0	0	0	3	9	2013
3	3	Tokat	Other	IL	6	4.5	6.0	6.0	4	4	...	12	10	6	18	12	12	6	2	2	2012
4	4	Gaziantep	Other	IL	3	4.0	3.0	4.0	2	2	...	1	3	2	3	4	3	3	5	9	2009

Most of the fields are self-explanatory. The following are descriptions for those that aren't.

- **Id** : Restaurant id.
- **Open Date** : opening date for a restaurant
- **City :** City that the restaurant is in. Note that there are unicode in the names.
- **City Group**: Type of the city. Big cities, or Other.
- **Type**: Type of the restaurant. FC: Food Court, IL: Inline, DT: Drive Thru, MB: Mobile
- **P1, P2 - P37**: There are three categories of these obfuscated data. **Demographic data** are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution, development scales. **Real estate data** mainly relate to the m2 of the location, front facade of the location, car park availability. **Commercial data** mainly include the existence of points of interest including schools, banks, other QSR operators.
- **Revenue**: The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the **target** of predictive analysis. Please note that the values are transformed so they don't mean real dollar values.

AutoML and CatBoost results confidently overperform median of distribution score



Pros/cons of packages:

Auto-sklearn	TPOT	Auto-ml
• derives best configurations (ensembles + weights) • allows time limit • allows exclude feature preprocessors and estimators • implicit parallelization, but exists • allows refit to new train_data	• explicit specification of models and parameters • wide range of models • easy to use (sklearn api) • allows number of iters	• full pipeline • strong models (catboost, deep) • ensembling weak estimators • Stop in any time
• requires preprocessing (doesn't work with NANs and strings) • use a lot memory (one hot encoding for categorical) • no meta-learning configurations for regression (but allows to add)	• requires preprocessing (doesn't work with strings) • use a lot memory (one hot encoding for categorical) • no explicit parallelization	• poor performance the problem that it evaluates papelines during small amount of time)

M4 forecasting competition overview

1. 100,000 different datasets
   a. Yearly time series ~ 50 samples
   b. Monthly time series ~ 200 samples
   c. Daily time series ~ 4000 samples
2. Additional information about the nature of time-series
3. The aim is to forecast time series for future time periods

#datasets in competition

Frequency	Demographic	Finance	Industry	Macro	Micro	Other	Total
Yearly	1,088	6,519	3,716	3,903	6,538	1,236	23,000
Quarterly	1,858	5,305	4,637	5,315	6,020	865	24,000
Monthly	5,728	10,987	10,017	10,016	10,975	277	48,000
Weekly	24	164	6	41	112	12	359
Daily	10	1,559	422	127	1,476	633	4,227
Hourly	0	0	0	0	0	414	414
Total	8,708	24,534	18,798	19,402	25,121	3,437	100,000

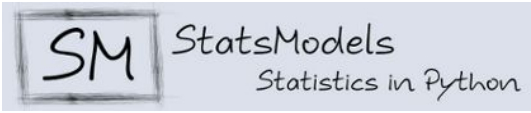
Example of daily time series

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V9911	V9912	V9913
0	D1	1017.10	1019.30	1017.00	1019.20	1018.70	1015.60	1018.50	1018.30	1018.4	...	NaN	NaN	NaN
1	D2	2793.70	2793.80	2803.70	2805.80	2802.30	2795.00	2806.40	2782.20	2780.3	...	NaN	NaN	NaN
2	D3	1091.30	1088.50	1085.70	1082.90	1080.10	1077.30	1074.50	1071.70	1068.9	...	NaN	NaN	NaN
3	D4	1092.00	1078.00	1064.00	1050.00	1036.00	1022.00	1008.00	1092.00	1078.0	...	NaN	NaN	NaN
4	D5	2938.63	2956.44	2964.41	2972.41	3014.97	3014.23	3024.08	3031.97	3062.7	...	NaN	NaN	NaN

	M4id	category	Frequency	Horizon	SP
0	Y1	Macro	1	6	Yearly
1	Y2	Macro	1	6	Yearly
2	Y3	Macro	1	6	Yearly
3	Y4	Macro	1	6	Yearly
4	Y5	Macro	1	6	Yearly
NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	

Pyramid (auto-arima) 2017

https://github.com/tgsmith61591/pyramid



Pyramid is auto-arima library, which automatically select (S)ARIMA model and tune their parameters.

Main concepts:

- Pyramid operates by wrapping statsmodels.tsa.ARIMA and statsmodels.tsa.statespace.SARIMAX
- uses stepwise algorithm laid out in a paper by Hyndman and Khandakar (2008) to select model hyper-parameters
- allows updating model
- as ARIMA and SARIMAX works with exogenous data
- the periodicity of the seasons should be determined
- you set the boundaries for hyperparameters

Autobox (Automatic Forecasting System) 2010 - **commercial** forecasting solution

http://autobox.com/cms/

Library is an enhancement of the original work of Box-Jenkins

Rewads

1. Picked as the “Best Dedicated Forecasting” Software in the “Principles of Forecasting” text book
2. Placed 12-th in the “NN5” 2010 Forecasting Competition on “Daily data”, but 1-st among Automated software (www.neural-forecasting-competition.com).
3. Placed 2-nd in the “NN3” 2007 Forecasting Competition on “Monthly data”, abt the 1-st on more difficult datasets (www.neural-forecasting-competition.com).

About Autobox

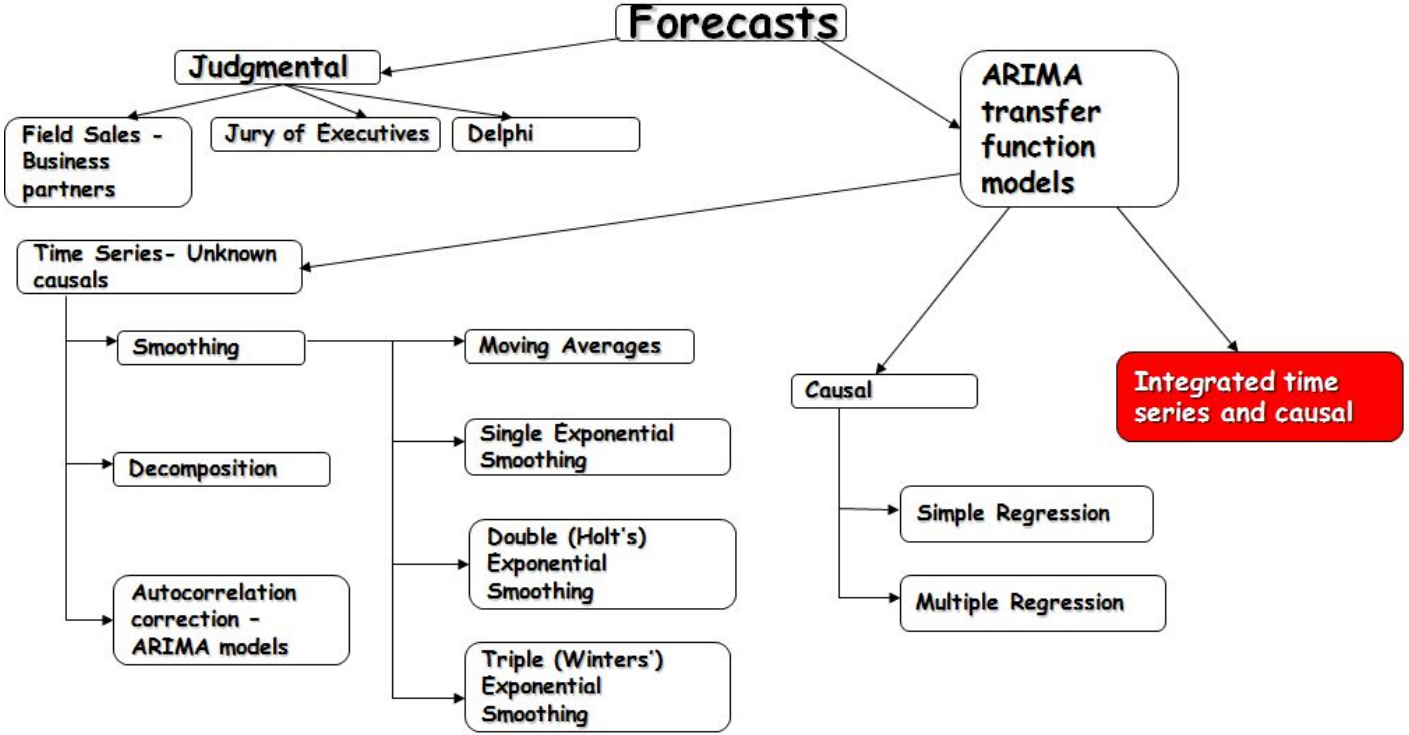
Autobox features and capabilities

1. Forecasting - Univariate, Multivariate
2. Data Cleaning - Generates a new dataset cleansed of outliers
3. Causality – Proof that a variable caused the outcome
4. Table of Probabilities- Determines probability of making month/quarter/year-end target

Autobox can **AUTOMATICALLY** detect

1. Outliers
2. Level Shifts
3. Trend Changes
4. Seasonality Changes
5. Parameter Changes
6. Variance Changes

Autobox - Forecasting methods family tree



Featuretools - automated feature generation python library

https://docs.featuretools.com/

Featuretools is a framework to perform automated feature engineering - it excels at transforming temporal and relational datasets into feature matrices for machine learning.

Main concepts:

- works with dataframes and their relationships
- uses feature primitives of two types:
  - **aggregation**: function that groups together child datapoints for each parent and then calculates a statistic such as mean, min, max, or standard deviation. An example is calculating the maximum loan amount for each client. An aggregation works across multiple tables using relationships between tables.
  - **transformation**: an operation applied to one or more columns in a single table. An example would be extracting the day from dates, or finding the difference between two columns in one table
- automated **Deep Feature Synthesis** with given max_depth
- *allows custom primitives*

Featuretools, transformations list

https://docs.featuretools.com/

Examples of primitives:

	name	type	description
0	mode	aggregation	Finds the most common element in a categorical feature.
1	any	aggregation	Test if any value is 'True'.
2	last	aggregation	Returns the last value.
3	all	aggregation	Test if all values are 'True'.
4	median	aggregation	Finds the median value of any feature with well-ordered values.
5	min	aggregation	Finds the minimum non-null value of a numeric feature.
6	avg_time_between	aggregation	Computes the average time between consecutive events.
7	max	aggregation	Finds the maximum non-null value of a numeric feature.
8	n_most_common	aggregation	Finds the N most common elements in a categorical feature.
9	percent_true	aggregation	Finds the percent of 'True' values in a boolean feature.

	name	type	description
19	add	transform	Creates a transform feature that adds two features.
20	seconds	transform	Transform a Timedelta feature into the number of seconds.
21	time_since_previous	transform	Compute the time since the previous instance.
22	cum_min	transform	Calculates the min of previous values of an instance for each value in a time-dependent entity.
23	divide	transform	Creates a transform feature that divides two features.
24	hour	transform	Transform a Datetime feature into the hour.
25	and	transform	For two boolean values, determine if both values are 'True'.
26	or	transform	For two boolean values, determine if one value is 'True'.
27	years	transform	Transform a Timedelta feature into the number of years.
28	isin	transform	For each value of the base feature, checks whether it is in a provided list.

Tsfresh - automated feature generation with focus on time-series data

http://tsfresh.readthedocs.io/en/latest/

tsfresh automatically calculates a large number of **time series characteristics**, the so called features. Further the package contains methods to evaluate the explaining power and importance of such characteristics for regression or classification tasks.

Example of features that are calculated by tsfresh

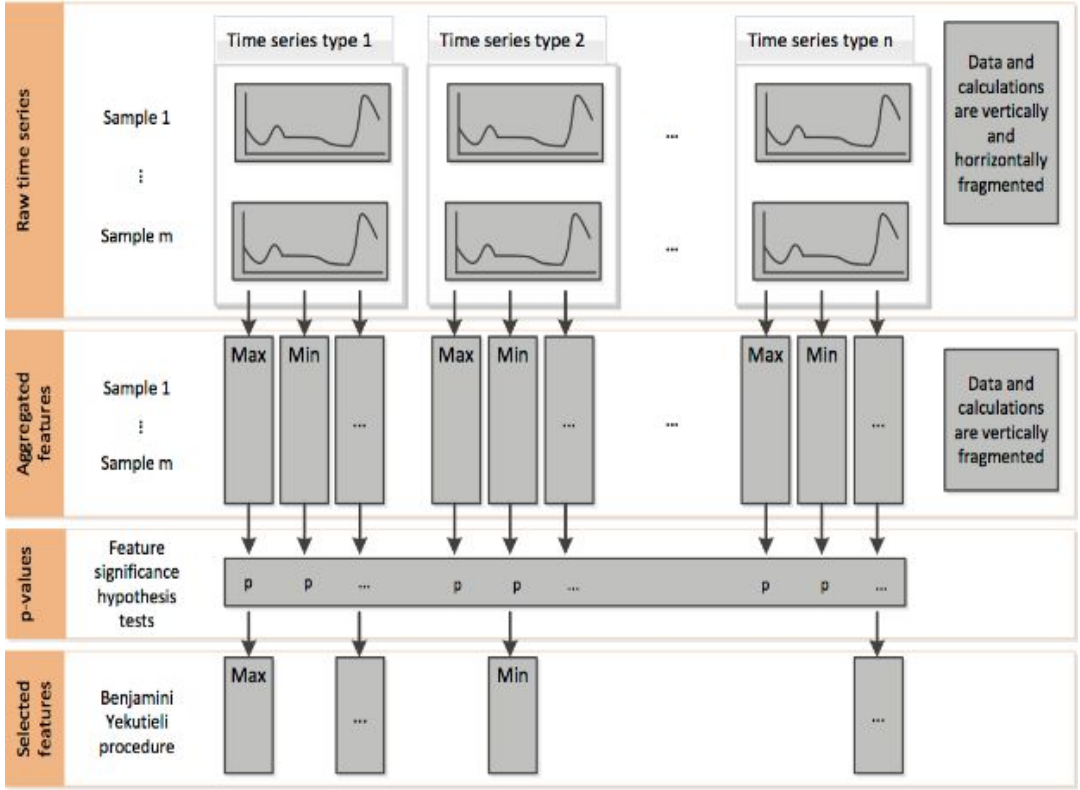
`kurtosis` (x)	
`large_standard_deviation` (x, r)	
`last_location_of_maximum` (x)	
`last_location_of_minimum` (x)	
`length` (x)	Returns the length of x
`linear_trend` (x, param)	Calculate a linear least-squares regression for th

Easy to implement custom features

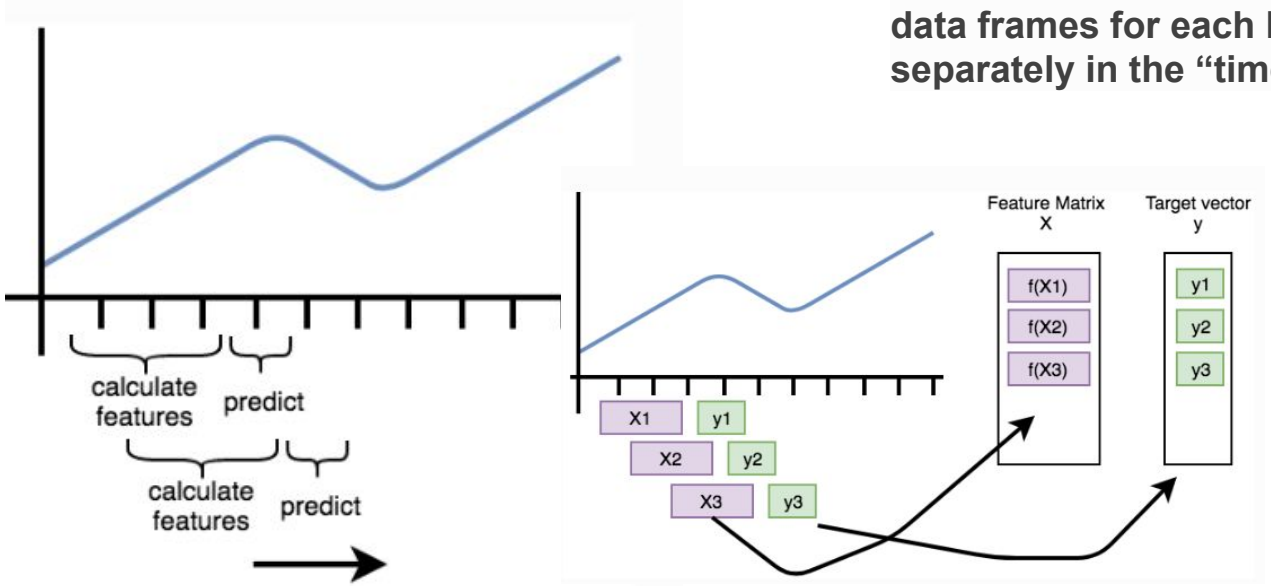
```
@set_property("fctype", "simple")
def your_feature_calculator(x):
    """
    The description of your feature

    :param x: the time series to calculate the feature of
    :type x: pandas.Series
    :return: the value of this feature
    :return type: bool, int or float
    """
    # Calculation of feature as float, int or bool
    f = f(x)
    return f
```

Tsfresh filtering automatically extracts relevant features

1. **Feature extraction**
2. **Feature significance testing**
3. **Multiple test procedure**



Time series forecasting with tsfresh



This method creates sub windows of the time series. It rolls the (sorted) data frames for each kind and each id separately in the “time” domain

Next week plans

1. Check Statsmodels, Pyramid(auto-arima)
2. Focus on automatic feature generation and feature selection
3. TSFresh + AutoML approach
4. Study solutions from M4 competition (pay attention on hybrid solutions)