Feature engineering

Jakub Ševcech

Obsah

Praca s casom a datumami

Zoskupovanie viacerych pozorovani

Numericke atributy

Normalizacia

Vyrabanie atributov kombinovanim

Suvisiace temy, ktore dnes nestihame

Transformacia kategorickych atributov na numericke

Vyber atributov - Feature selection

Extrakcia atributov

Nevyvazene datasety

Vycerpavajuci zoznam mnozstva veci, co sa da robit na pripravu dat a na feature engineering najdete tu: http://www.datasciencecentral.com/profiles/blogs/feature-engineering-data-scientist-s-secret-sauce-1 (http://www.datasciencecentral.com/profiles/blogs/feature-engineering-data-scientist-s-secret-sauce-1)

Nanestastie je to len zoznam

Celkom pekny zoznam krokov a aj metod je tu: http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/ (http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/)

Zopar najcastejsie pouzivanych veci z toho vyberiem a ukazem na co je to dobre

In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn

plt.rcParams['figure.figsize'] = 9, 6
from IPython.display import Image
```

Najskor sa skuste pozriet do dat a skusit tam najst nejake vlastnosti sami.

Napriklad rozbite atributy, kde sa v jednom nachadza viacero hodnot.

Priklad na datach o potopeni Titanicu https://www.kaggle.com/c/titanic (https://www.kaggle.com/c/titanic)

In [2]:
```
titanic = pd.read_csv('titanic/train.csv')
titanic.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [3]:
```
import re
titanic.Name.apply(lambda x: re.split('[,.]', x)).head(10)
```

Out[3]:
```
0                          [Braund,  Mr,  Owen Harris]
1    [Cumings,  Mrs,  John Bradley (Florence Briggs...
2                           [Heikkinen,  Miss,  Laina]
3     [Futrelle,  Mrs,  Jacques Heath (Lily May Peel)]
4                         [Allen,  Mr,  William Henry]
5                                 [Moran,  Mr,  James]
6                          [McCarthy,  Mr,  Timothy J]
7                  [Palsson,  Master,  Gosta Leonard]
8    [Johnson,  Mrs,  Oscar W (Elisabeth Vilhelmina...
9              [Nasser,  Mrs,  Nicholas (Adele Achem)]
Name: Name, dtype: object
```

In [4]:
```
titanic.Name.apply(lambda x: re.split('\s*[,.]\s*', x)).head(10)
```

Out[4]:
```
0                              [Braund, Mr, Owen Harris]
1    [Cumings, Mrs, John Bradley (Florence Briggs T...
2                               [Heikkinen, Miss, Laina]
3       [Futrelle, Mrs, Jacques Heath (Lily May Peel)]
4                             [Allen, Mr, William Henry]
5                                     [Moran, Mr, James]
6                              [McCarthy, Mr, Timothy J]
7                       [Palsson, Master, Gosta Leonard]
8    [Johnson, Mrs, Oscar W (Elisabeth Vilhelmina B...
9                  [Nasser, Mrs, Nicholas (Adele Achem)]
Name: Name, dtype: object
```

In [5]:
```
set(titanic.Name.apply(lambda x: re.split('\s*[,.]\s*', x)[1]))
```

Out[5]:
```
{'Capt',
 'Col',
 'Don',
 'Dr',
 'Jonkheer',
 'Lady',
 'Major',
 'Master',
 'Miss',
 'Mlle',
 'Mme',
 'Mr',
 'Mrs',
 'Ms',
 'Rev',
 'Sir',
 'the Countess'}
```

In [6]:
```
titanic['title'] = titanic.Name.apply(lambda x: re.split('\s*[,.]\s*', x)[1])
titanic.title.head(10)
```

Out[6]:
```
0        Mr
1       Mrs
2      Miss
3       Mrs
4        Mr
5        Mr
6        Mr
7    Master
8       Mrs
9       Mrs
Name: title, dtype: object
```

In [7]:
```
titanic.title.value_counts()
```

Out[7]:
```
Mr              517
Miss            182
Mrs             125
Master           40
Dr                7
Rev               6
Major             2
Mlle              2
Col               2
Ms                1
Mme               1
Lady              1
the Countess      1
Capt              1
Jonkheer          1
Sir               1
Don               1
Name: title, dtype: int64
```

Teraz je tu zopar titulov, ktorych pocty su fakt zanedbatelne a ten model by sa ich tazko ucil. Hrozilo by praveze, ze sa preuci

Chcelo by to aby sa pospajali dohromady.

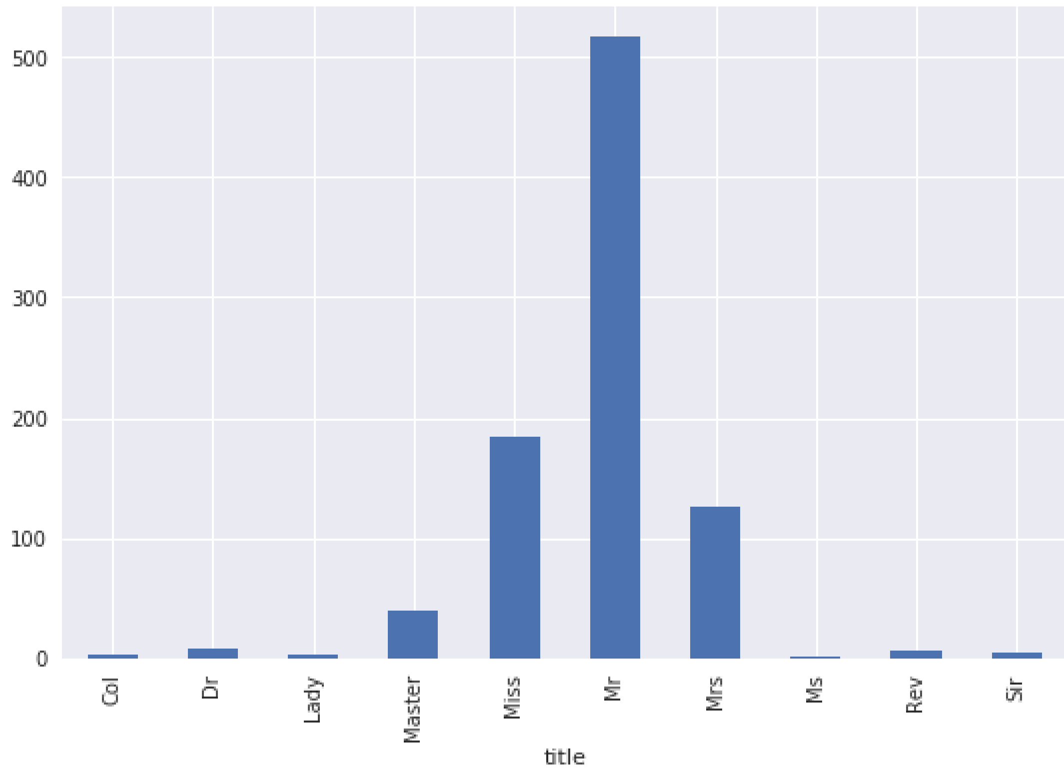
Napriklad Mlle = Mademoiselle = Miss

In [8]:
```
titanic.loc[titanic.title == 'Mlle', 'title'] = 'Miss'
titanic.loc[titanic.title == 'Mme', 'title'] = 'Mrs'
titanic.loc[titanic.title.isin(['Capt', 'Don', 'Major']), 'title'] = 'Sir'
titanic.loc[titanic.title.isin(['Dona', 'Lady', 'the Countess', 'Jonkheer']), 'title'] = 'Lady'
```

In [9]:
```
titanic.groupby('title').size().plot(kind='bar')
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff34478b828>
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/font_manager.py:1297: UserWarning: findfont: Font family ['sa
ns-serif'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



Velmi pekna ukazka predspracovania na titanic datasete je tu: http://trevorstephens.com/kaggle-titanic-tutorial/r-part-4-feature-engineering/ (http://trevorstephens.com/kaggle-titanic-tutorial/r-part-4-feature-engineering/)

Niekedy ma zmysel upravovat nielen nezavisle premenne (atributy), ale aj tie zavisle (predikovana hodnota)

Napriklad pri predikovani casu pristania lietadla nieje uplne dobry napad predikovat ten cas, ale skor dobu letu a potom ju len pripocitat k casu vzlietnutie

In [10]:
```
url = 'http://www.datasciencecentral.com/profiles/blogs/predicting-flights-delay-using-supervised-learning'
from IPython.display import IFrame
IFrame(url, width=700, height=350)
```

Out[10]:

Praca s casom a datumami

In [11]:
```
# https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+#
# klasifikacia, ci je miestnost obsadena alebo nie na zaklade udajov zo senzorov
occupancy = pd.read_csv('occupancy/datatraining.txt', sep=',')
date = pd.to_datetime(occupancy.date, format='%Y-%m-%d %H:%M:%S')
occupancy.date = date
occupancy.head()
```

Out[11]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1

Cas ako taky velmi vyuzit neviete. To rozlysenie je prilis drobne a ziadny model sa z toho nic nenauci.

Potrebovali by sme ten datum nejak predspracovat.

Identifikator dni by nam uz mohol pomoct. Granularita je podstatne hrubsia a model by to uz mohol vediet vyuzit.

In [12]:
```
occupancy['weekday'] = occupancy.date.dt.weekday
occupancy.head()
```

Out[12]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	weekday
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1	2
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1	2
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1	2
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1	2
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1	2

Co ked sa dozvieme, ze cez vikend sa v tej miestnosti nekuri a rozhadzuje nam to teplotu v nameranych datach?

In [13]:
```
occupancy['weekend'] = False
occupancy.loc[occupancy.weekday.isin([5, 6]), 'weekend'] = True
occupancy.tail()
```

Out[13]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	weekday	weekend
8139	2015-02-10 09:29:00	21.05	36.0975	433.0	787.250000	0.005579	1	1	False
8140	2015-02-10 09:29:59	21.05	35.9950	433.0	789.500000	0.005563	1	1	False
8141	2015-02-10 09:30:59	21.10	36.0950	433.0	798.500000	0.005596	1	1	False
8142	2015-02-10 09:32:00	21.10	36.2600	433.0	820.333333	0.005621	1	1	False
8143	2015-02-10 09:33:00	21.10	36.2000	447.0	821.000000	0.005612	1	1	False

In [14]:
```
occupancy.weekend.value_counts()
```

Out[14]:
```
False    5263
True     2880
Name: weekend, dtype: int64
```

- Takto mozeme pracovat s dnami, tyzdnami, hodinami ....
- Vieme vyberat konkretne hodiny a najst napriklad aktivnu cast dna

- Mozeme najst rozne fazy dna kde sa ludia spravaju rozne

> uzitocne napriklad pri analyze spravania sa ludi na webe - tesne poobede asi ludia citaju ine clanky ako v produktivnejsej casti dna

- vlastne, rozne dlhe intervaly

Uloha na doma

- Co ked sa v noci tiez nekuri?
- Co ked v nejake hodiny svieti cez okno slnko a tiez to rozhadzuje teplotu?

In [15]:
```
# rozne atributy, ktore sa daju vytiahnut z casovej peciatky
url = 'http://pandas.pydata.org/pandas-docs/stable/api.html#datetimelike-properties'
IFrame(url, width=1000, height=350)
```

Out[15]:

Zoskupovanie viacerych pozorovani

Zoskupovanie viacerych transakcii per objekt

Typicke pri sledovani aktivity pouzivatelov kde zoskupujeme per pouzivatel

Napriklad kliknutia na stranky alebo operacie v banke

Ukazeme si zopar prikladov na datach o bankovych operaciach nedzi roznymi uctami.

Je to typicky priklad transakcnych dat

[http://lisp.vse.cz/pkdd99/berka.htm (http://lisp.vse.cz/pkdd99/berka.htm)](http://lisp.vse.cz/pkdd99/berka.htm)

In [16]:
```
trans = pd.read_csv('berka/trans.asc', sep=';', nrows=1000)
trans.head()
```

Out[16]:

	trans_id	account_id	date	type	operation	amount	balance	k_symbol	bank	account
0	695247	2378	930101	PRIJEM	VKLAD	700.0	700.0	NaN	NaN	NaN
1	171812	576	930101	PRIJEM	VKLAD	900.0	900.0	NaN	NaN	NaN
2	207264	704	930101	PRIJEM	VKLAD	1000.0	1000.0	NaN	NaN	NaN
3	1117247	3818	930101	PRIJEM	VKLAD	600.0	600.0	NaN	NaN	NaN
4	579373	1972	930102	PRIJEM	VKLAD	400.0	400.0	NaN	NaN	NaN

In [17]:
```
date = pd.to_datetime(trans.date, format='%y%m%d')
trans.date = date
trans.head()
```

Out[17]:

	trans_id	account_id	date	type	operation	amount	balance	k_symbol	bank	account
0	695247	2378	1993-01-01	PRIJEM	VKLAD	700.0	700.0	NaN	NaN	NaN
1	171812	576	1993-01-01	PRIJEM	VKLAD	900.0	900.0	NaN	NaN	NaN
2	207264	704	1993-01-01	PRIJEM	VKLAD	1000.0	1000.0	NaN	NaN	NaN
3	1117247	3818	1993-01-01	PRIJEM	VKLAD	600.0	600.0	NaN	NaN	NaN
4	579373	1972	1993-01-02	PRIJEM	VKLAD	400.0	400.0	NaN	NaN	NaN

Rozne typy operacii maju uplne ine vlastnosti. Mohli by sme to pouzit na to, aby sme vytvorili nove atributy, ktore budu tieto skupiny oddelovat.

In [18]:
```
means = trans.groupby('operation').amount.mean()
means
```

Out[18]:
```
operation
PREVOD NA UCET     6361.600000
PREVOD Z UCTU     11329.065359
VKLAD              8025.238953
VYBER             10925.340000
Name: amount, dtype: float64
```

In [19]:
```
trans['mean_amount_per_operation'] = trans.apply(lambda x: 0 if pd.isnull(x['operation']) else means[x['operatio
trans.tail()
```

Out[19]:

	trans_id	account_id	date	type	operation	amount	balance	k_symbol	bank	account	mean_amount_per_operation
995	1977575	6701	1993-03-26	PRIJEM	VKLAD	29082.0	29282.0	NaN	NaN	NaN	8025.238953
996	1602509	5442	1993-03-26	PRIJEM	VKLAD	34681.0	42632.0	NaN	NaN	NaN	8025.238953
997	2127306	7203	1993-03-26	PRIJEM	VKLAD	900.0	900.0	NaN	NaN	NaN	8025.238953
998	308090	1050	1993-03-26	VYDAJ	VYBER	800.0	25517.0	NaN	NaN	NaN	10925.340000
999	401329	1366	1993-03-26	PRIJEM	VKLAD	5200.0	13726.0	NaN	NaN	NaN	8025.238953

Podobny vysledok viem dosiahnut aj operaciou merge (join v SQL)

je to trochu rychlejsie aj ked lambda mi dava podstatne sirsie moznosti operacii

In [20]:
```
means_df = pd.DataFrame(means)
means_df.columns = ['mean_amount_per_operation']
means_df['operation'] = means_df.index
means_df.index = range(len(means_df))
means_df
```

Out[20]:

	mean_amount_per_operation	operation
0	6361.600000	PREVOD NA UCET
1	11329.065359	PREVOD Z UCTU
2	8025.238953	VKLAD
3	10925.340000	VYBER

In [21]:
```
pd.merge(trans, means_df, how='left', on='operation')
```

Out[21]:

	trans_id	account_id	date	type	operation	amount	balance	k_symbol	bank	account	mean_amount_per_operation_x	me
0	695247	2378	1993-01-01	PRIJEM	VKLAD	700.0	700.0	NaN	NaN	NaN	8025.238953	
1	171812	576	1993-01-01	PRIJEM	VKLAD	900.0	900.0	NaN	NaN	NaN	8025.238953	
2	207264	704	1993-01-01	PRIJEM	VKLAD	1000.0	1000.0	NaN	NaN	NaN	8025.238953	
3	1117247	3818	1993-01-01	PRIJEM	VKLAD	600.0	600.0	NaN	NaN	NaN	8025.238953	
4	579373	1972	1993-01-02	PRIJEM	VKLAD	400.0	400.0	NaN	NaN	NaN	8025.238953	
5	771035	2632	1993-01-02	PRIJEM	VKLAD	1100.0	1100.0	NaN	NaN	NaN	8025.238953	
6	452728	1539	1993-01-03	PRIJEM	VKLAD	600.0	600.0	NaN	NaN	NaN	8025.238953	

Podobnym sposobom mozete vyrobit velmi vela roznych atributov

- nepridat len priemer, ale aj pomer, >, < alebo rozdiel medzi priemerom a aktualnou hodnotou
- priemer per account => odrazi zvyky konkretneho zakaznika
- viete pouzit hociaky kategoricky atribut an vytvorenie segmentov
- agregovanie inych atributov
- agregovanie v casovom okne
- ine agregacne funkcie: count, std, kvartily ....
- viete dokonca vytvorit nove kategoriccke atributy podla ktorych ma zmysel agregovat pomocou zhlukovania

Numericke atributy

co zaujimave sa da robit s nimi?

In [22]:
```
# data o komunikacii pocitacov v sieti z www.neteye-blog.com/netcla-the-ecml-pkdd-network-classification-challeng
# uloha je klasifikovat aka aplikacia data generovala
data_file = "../vos/challenge/NetCla/data/train.csv"
netcla = pd.read_csv(data_file, nrows=1000, sep='\t')
netcla.head()
```

Out[22]:

	cli_pl_header	cli_pl_body	cli_cont_len	srv_pl_header	srv_pl_body	srv_cont_len	aggregated_sessions	bytes	net_samples	tcp_frag
0	667	0	0	336	143151	143151	1	155494	1	0
1	667	0	0	336	143151	143151	1	310988	1	0
2	667	0	0	336	143151	143151	1	310988	1	0
3	592	0	0	238	1269	1269	1	2401	1	0
4	592	0	0	238	1269	1269	1	4802	1	0

5 rows × 49 columns

In [23]:
```
netcla.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 49 columns):
cli_pl_header          1000 non-null int64
cli_pl_body            1000 non-null int64
cli_cont_len           1000 non-null int64
srv_pl_header          1000 non-null int64
srv_pl_body            1000 non-null int64
srv_cont_len           1000 non-null int64
aggregated_sessions    1000 non-null int64
bytes                  1000 non-null int64
net_samples            1000 non-null int64
tcp_frag               1000 non-null int64
tcp_pkts               1000 non-null int64
tcp_retr               1000 non-null int64
tcp_ooo                1000 non-null int64
cli_tcp_pkts           1000 non-null int64
cli_tcp_ooo            1000 non-null int64
cli_tcp_retr           1000 non-null int64
cli_tcp_frag           1000 non-null int64
cli_tcp_empty          1000 non-null int64
cli_win_change         1000 non-null int64
cli_win_zero           1000 non-null int64
cli_tcp_full           1000 non-null int64
cli_tcp_tot_bytes      1000 non-null int64
cli_pl_tot             1000 non-null int64
cli_pl_change          1000 non-null int64
srv_tcp_pkts           1000 non-null int64
srv_tcp_ooo            1000 non-null int64
srv_tcp_retr           1000 non-null int64
srv_tcp_frag           1000 non-null int64
srv_tcp_empty          1000 non-null int64
srv_win_change         1000 non-null int64
srv_win_zero           1000 non-null int64
srv_tcp_full           1000 non-null int64
srv_tcp_tot_bytes      1000 non-null int64
srv_pl_tot             1000 non-null int64
srv_pl_change          1000 non-null int64
srv_tcp_win            1000 non-null int64
srv_tx_time            1000 non-null float64
cli_tcp_win            1000 non-null int64
```

```
client_latency         1000 non-null float64
application_latency    1000 non-null float64
cli_tx_time            1000 non-null float64
load_time              1000 non-null float64
server_latency         1000 non-null float64
proxy                  1000 non-null int64
sp_healthscore         1000 non-null int64
sp_req_duration        1000 non-null int64
sp_is_lat              1000 non-null int64
sp_error               1000 non-null int64
throughput             1000 non-null float64
dtypes: float64(7), int64(42)
memory usage: 382.9 KB
```

In [24]:

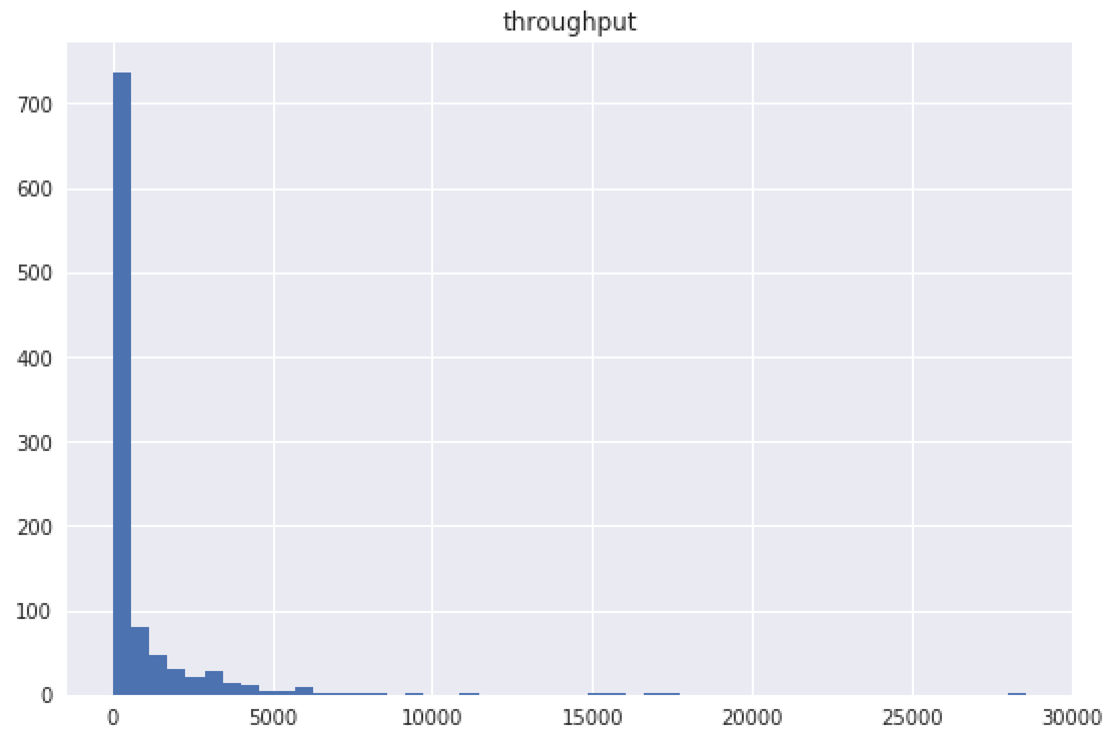
```
# seaborn.pairplot(netcla) # toto nije dobry napad. Tych atributov je strasne vela a kazdy s kazdym je dost velk
plt.rcParams['figure.figsize'] = 18, 12
netcla.hist()
plt.rcParams['figure.figsize'] = 9, 6
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/font_manager.py:1297: UserWarning: findfont: Font family ['sa
ns-serif'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```

In [25]:
```
pom = netcla.throughput.hist(bins=50)
pom.set_title('throughput')
```

Out[25]: <matplotlib.text.Text at 0x7ff3426aa320>

```
/usr/local/lib/python3.5/dist-packages/matplotlib/font_manager.py:1297: UserWarning: findfont: Font family ['sa
ns-serif'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



Rozne algoritmy mozu mat s takymto rozdelenim problem.

Logisticka regresia, neuronova siet alebo hocico, co pouziva vahy na atributoch

Aku mate dat vahu atributu ak ma rozsah od 0 do 10^6 a vacsina hodnot je sustredena na jednu stranu?

Velke hodnty vam budu velmi ovplyvnovat cely vypocet a nebudete vediet rozlysit tie male.

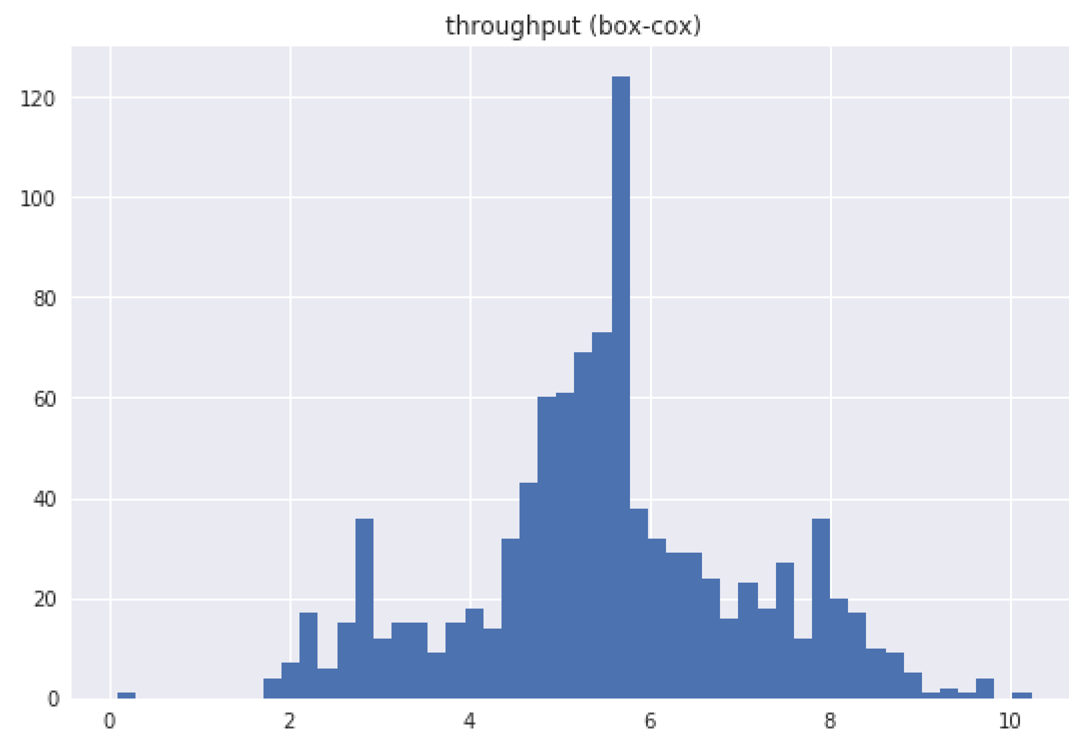
In [26]:
```
from scipy.stats import boxcox
```

namiesto box-cox by sa dal pouzit logaritmus povodnej hodnoty, ale to je hack. box-cox sa dokaze postarat o to, aby sa vysledne rozelenie podobalo normalnemu

In [27]:
```
# boxcox vrati transformovane data a parametre transformacie. Tie viem zafixovat a v tom pripade mi to vrati len
transformed, att = boxcox(netcla.throughput+1)# nevieme transformovat 0 a zaporne hodnoty. preto + 1
pom = pd.Series(transformed).hist(bins=50)
pom.set_title('throughput (box-cox)')
```

Out[27]:
```
<matplotlib.text.Text at 0x7ff34113a2e8>
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/font_manager.py:1297: UserWarning: findfont: Font family ['sa
ns-serif'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



Toto sa uz trochu podoba na normalne rozdelenie ale nie je normalizovane

Zakladny typ normalizacie (skalovania) je vydelit na opravenie skaly

a odcitat nieco na opravu posunutia

In [28]:
```
def normalization(data, shift, scale):
    return (np.array(data) - float(shift))/scale
```

- z-normalization: shift = mean, scale = std
- 0-1 normalization: shift = min, scale = max - min
- kvartily na odstranenie outlierov

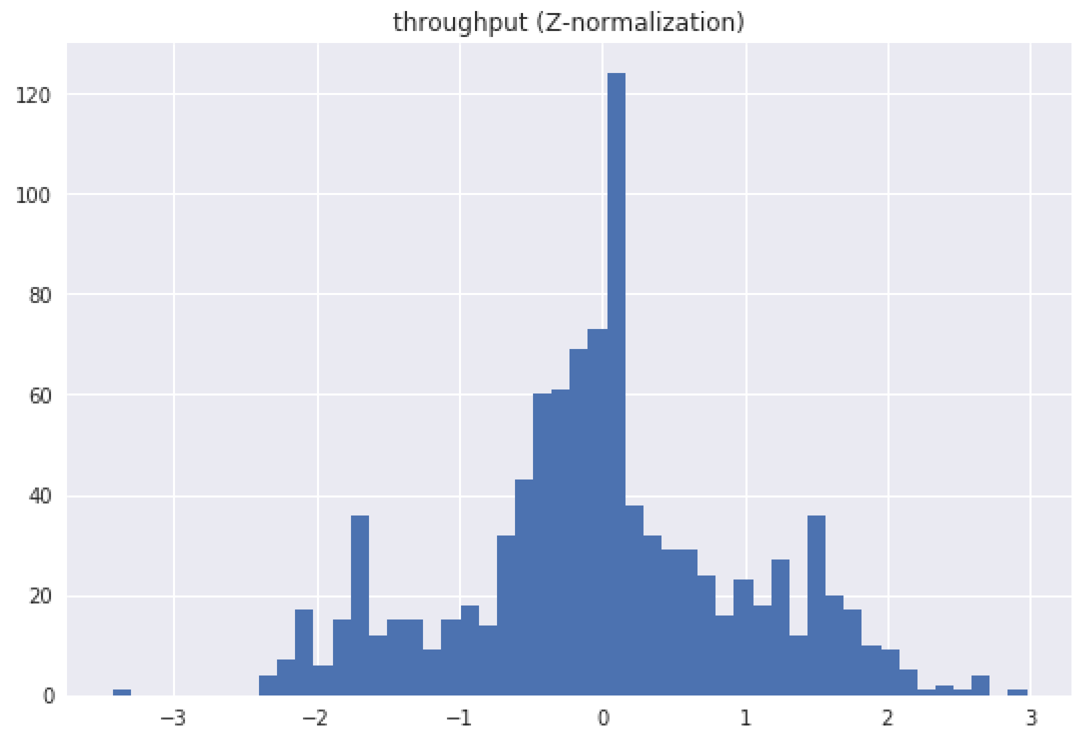
Nemusite to programovat. Sklearn uz ma nieco pripravene

- toto iste robi sklearn.preprocessing.StandardScaler pre vsetky atributy. Pozor ale na power law
- ak pouzijete RobustScaler, tak si viete poradit aj s outliermi (pouziva rozsah 1st quartile (25th quantile) and the 3rd quartile (75th quantile))

In [29]:
```
z_transformed = normalization(transformed, np.mean(transformed), np.std(transformed))
pom = pd.Series(z_transformed).hist(bins=50)
pom.set_title('throughput (Z-normalization)')
```

Out[29]: <matplotlib.text.Text at 0x7ff3410b47b8>

```
/usr/local/lib/python3.5/dist-packages/matplotlib/font_manager.py:1297: UserWarning: findfont: Font family ['sa
ns-serif'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



Skalovanie / normalizacia nie je dolezita len pri trenovani modelu.

PCA (redukcia dimenzionality) napriklad to potrebuje tiez

PCA sa snazi vysvetlit varianciu v datach co najmensim poctom atributov. Vie ovahovat atributy datasetu podla toho kolko je v nich variancie a s takou vahou ich pouziva. Ak mate nenormalizovany atribut, ktory ma varianciu ovela vacsiu ako ostatne, tak bude vo vysledkej reprezentacii odrazeny velmi silno na ukor ostatnych. Nie vzdy to chcete.

Vyrabanie atributov kombinovanim

Niekedy vam samotny atribut velmi nepomoze. Ak ho ale nejak skombinujete s inym, tak uz moze.

Napriklad mate dataset aut, kde je spotreba a velkost nadrze. Tieto atributy su fajn, ale ak ich medzi sebou prenasobite, tak dostanete dojazd co moze tiez velmi dolezite pri vybere auta.

Na toto obycajne potrebujete domenovu znalost. Tento proces sa ale da do nejakej miery automatizovat.

Polynomialne kombinovanie

In [30]:
```
X = np.arange(6).reshape(3, 2)
X
```

Out[30]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [31]:
```
from sklearn import preprocessing
poly = preprocessing.PolynomialFeatures(3)
poly.fit_transform(X) # vytvorenie polynomialnych kombinacii
```

Out[31]:
```
array([[   1.,    0.,    1.,    0.,    0.,    1.,    0.,    0.,    0.,
           1.],
       [   1.,    2.,    3.,    4.,    6.,    9.,    8.,   12.,   18.,
          27.],
       [   1.,    4.,    5.,   16.,   20.,   25.,   64.,   80.,  100.,
         125.]])
```

a^0, a, b, a^2, a*b, b^2, a^3, a^2*b, a*b^2, b^3*

Znovu si zoberiem data z Titanicu

In [32]:
```
titanic = pd.read_csv('titanic/train.csv')
titanic.head()
```

Out[32]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

nevieme pouzivat prazdne hodnoty pri vypocte polynomialnych vlastnosti

rozdelime si data na zavisle a nezavisle premenne

In [33]:
```
titanic_X = titanic.dropna().reindex(columns=[x for x in titanic.columns.values if x != 'Survived']).reset_index(
titanic_y = titanic.dropna().reindex(columns=['Survived']).reset_index(drop=True)
```

In [35]:
```
titanic_X
```

Out[35]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
1	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
2	7	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
3	11	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
4	12	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
5	22	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
6	24	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
7	28	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
8	53	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	D33	C
9	55	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792	B30	C
10	63	1	Harris, Mr. Henry Birkhardt	male	45.0	1	0	36973	83.4750	C83	S
11	67	2	Nye, Mrs. (Elizabeth Ramell)	female	29.0	0	0	C.A. 29395	10.5000	F33	S
12	76	3	Moen, Mr. Sigurd Hansen	male	25.0	0	0	348123	7.6500	F G73	S
13	89	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
14	93	1	Chaffee, Mr. Herbert Fuller	male	46.0	1	0	W.E.P. 5734	61.1750	E31	S
15	97	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
16	98	1	Greenfield, Mr. William Bertram	male	23.0	0	1	PC 17759	63.3583	D10 D12	C
17	103	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	D26	S
18	111	1	Porter, Mr. Walter Chamberlain	male	47.0	0	0	110465	52.0000	C110	S
19	119	1	Baxter, Mr. Quigg Edmond	male	24.0	0	1	PC 17558	247.5208	B58 B60	C

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
20	124	2	Webber, Miss. Susan	female	32.5	0	0	27267	13.0000	E101	S
21	125	1	White, Mr. Percival Wayland	male	54.0	0	1	35281	77.2875	D26	S
22	137	1	Newsom, Miss. Helen Monypeny	female	19.0	0	2	11752	26.2833	D47	S
23	138	1	Futrelle, Mr. Jacques Heath	male	37.0	1	0	113803	53.1000	C123	S
24	140	1	Giglio, Mr. Victor	male	24.0	0	0	PC 17593	79.2000	B86	C
25	149	2	Navratil, Mr. Michel ("Louis M Hoffman")	male	36.5	0	2	230080	26.0000	F2	S
26	152	1	Pears, Mrs. Thomas (Edith Wearne)	female	22.0	1	0	113776	66.6000	C2	S
27	171	1	Van der hoef, Mr. Wyckoff	male	61.0	0	0	111240	33.5000	B19	S
28	175	1	Smith, Mr. James Clinch	male	56.0	0	0	17764	30.6958	A7	C
29	178	1	Isham, Miss. Ann Elizabeth	female	50.0	0	0	PC 17595	28.7125	C49	C
...	...	...	...	...	...	...	...	...	...	...	...
153	738	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C
154	742	1	Cavendish, Mr. Tyrell William	male	36.0	1	0	19877	78.8500	C46	S
155	743	1	Ryerson, Miss. Susan Parker "Suzette"	female	21.0	2	2	PC 17608	262.3750	B57 B59 B63 B66	C
156	746	1	Crosby, Capt. Edward Gifford	male	70.0	1	1	WE/P 5735	71.0000	B22	S
157	749	1	Marvin, Mr. Daniel Warner	male	19.0	1	0	113773	53.1000	D30	S
158	752	3	Moor, Master. Meier	male	6.0	0	1	392096	12.4750	E121	S
159	760	1	Rothes, the Countess. of (Lucy Noel Martha Dye...	female	33.0	0	0	110152	86.5000	B77	S
160	764	1	Carter, Mrs. William Ernest (Lucile Polk)	female	36.0	1	2	113760	120.0000	B96 B98	S
161	766	1	Hogeboom, Mrs. John C (Anna Andrews)	female	51.0	1	0	13502	77.9583	D11	S
162	773	2	Mack, Mrs. (Mary)	female	57.0	0	0	S.O./P.P. 3	10.5000	E77	S
163	780	1	Robert, Mrs. Edward Scott (Elisabeth Walton Mc...	female	43.0	0	1	24160	211.3375	B3	S
164	782	1	Dick, Mrs. Albert Adrian (Vera Gillespie)	female	17.0	1	0	17474	57.0000	B20	S

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
165	783	1	Long, Mr. Milton Clyde	male	29.0	0	0	113501	30.0000	D6	S
166	790	1	Guggenheim, Mr. Benjamin	male	46.0	0	0	PC 17593	79.2000	B82 B84	C
167	797	1	Leader, Dr. Alice (Farnham)	female	49.0	0	0	17465	25.9292	D17	S
168	803	1	Carter, Master. William Thornton II	male	11.0	1	2	113760	120.0000	B96 B98	S
169	807	1	Andrews, Mr. Thomas Jr	male	39.0	0	0	112050	0.0000	A36	S
170	810	1	Chambers, Mrs. Norman Campbell (Bertha Griggs)	female	33.0	1	0	113806	53.1000	E8	S
171	821	1	Hays, Mrs. Charles Melville (Clara Jennings Gr...	female	52.0	1	1	12749	93.5000	B69	S
172	824	3	Moor, Mrs. (Beila)	female	27.0	0	1	392096	12.4750	E121	S
173	836	1	Compton, Miss. Sara Rebecca	female	39.0	1	1	PC 17756	83.1583	E49	C
174	854	1	Lines, Miss. Mary Conover	female	16.0	0	1	PC 17592	39.4000	D28	S
175	858	1	Daly, Mr. Peter Denis	male	51.0	0	0	113055	26.5500	E17	S
176	863	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0	0	17466	25.9292	D17	S
177	868	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	PC 17590	50.4958	A24	S
178	872	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
179	873	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
180	880	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
181	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
182	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

183 rows × 11 columns

In [36]:
```
poly = preprocessing.PolynomialFeatures(2)
# pozor na prilis velke cislo. Vznikne vela atributov a hrozi ze budete mat malo dat na natrenovanie
polynomial_titanic = poly.fit_transform(titanic_X)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-36-b0714673e126> in <module>()
      1 poly = preprocessing.PolynomialFeatures(2)
      2 # pozor na prilis velke cislo. Vznikne vela atributov a hrozi ze budete mat malo dat na natrenovanie
----> 3 polynomial_titanic = poly.fit_transform(titanic_X)

/usr/local/lib/python3.5/dist-packages/sklearn/base.py in fit_transform(self, X, y, **fit_params)
    516         if y is None:
    517             # fit method of arity 1 (unsupervised transformation)
--> 518             return self.fit(X, **fit_params).transform(X)
    519         else:
    520             # fit method of arity 2 (supervised transformation)

/usr/local/lib/python3.5/dist-packages/sklearn/preprocessing/data.py in fit(self, X, y)
   1307         self : instance
   1308         """
-> 1309         n_samples, n_features = check_array(X).shape
   1310         combinations = self._combinations(n_features, self.degree,
   1311                                           self.interaction_only,

/usr/local/lib/python3.5/dist-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, dtype,
 order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_on_dtype, es
timator)
    415         # make sure we actually converted to numeric:
    416         if dtype_numeric and array.dtype.kind == "O":
--> 417             array = array.astype(np.float64)
    418         if not allow_nd and array.ndim >= 3:
    419             raise ValueError("Found array with dim %d. %s expected <= 2."

ValueError: could not convert string to float: 'C'
```

Nemozeme pouzivat kategoricke atributy. Len numericke

In [37]:
```
polynomial_titanic = poly.fit_transform(titanic_X._get_numeric_data())
```

In [38]:
```
polynomial = pd.DataFrame(polynomial_titanic)
polynomial.head()
```

Out[38]:

	0	1	2	3	4	5	6	7	8	9	...	18	19	20	21	22	23	24	25	26	27
0	1.0	2.0	1.0	38.0	1.0	0.0	71.2833	4.0	2.0	76.0	...	1444.0	38.0	0.0	2708.7654	1.0	0.0	71.2833	0.0	0.0	5081.308859
1	1.0	4.0	1.0	35.0	1.0	0.0	53.1000	16.0	4.0	140.0	...	1225.0	35.0	0.0	1858.5000	1.0	0.0	53.1000	0.0	0.0	2819.610000
2	1.0	7.0	1.0	54.0	0.0	0.0	51.8625	49.0	7.0	378.0	...	2916.0	0.0	0.0	2800.5750	0.0	0.0	0.0000	0.0	0.0	2689.718906
3	1.0	11.0	3.0	4.0	1.0	1.0	16.7000	121.0	33.0	44.0	...	16.0	4.0	4.0	66.8000	1.0	1.0	16.7000	1.0	16.7	278.890000
4	1.0	12.0	1.0	58.0	0.0	0.0	26.5500	144.0	12.0	696.0	...	3364.0	0.0	0.0	1539.9000	0.0	0.0	0.0000	0.0	0.0	704.902500

5 rows × 28 columns

In [39]:
```
titanic_X._get_numeric_data().shape
```

Out[39]:
```
(183, 6)
```

ak by ste chceli zachovat rozumne nazvy stlpcov: [http://stackoverflow.com/questions/36728287/sklearn-preprocessing-polynomialfeatures-how-to-keep-column-names-headers-of (http://stackoverflow.com/questions/36728287/sklearn-preprocessing-polynomialfeatures-how-to-keep-column-names-headers-of)](http://stackoverflow.com/questions/36728287/sklearn-preprocessing-polynomialfeatures-how-to-keep-column-names-headers-of)

Dobre, skusme natrenovat model a pozriet sa ako nam tieto polynomialne vlastnosti zmenia uspesnost

Najskor povodne data

In [40]:
```
# berieme len numericke data aby sme mali rovnake podmienky
original = titanic.dropna()._get_numeric_data()
original.head()
```

Out[40]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
1	2	1	1	38.0	1	0	71.2833
3	4	1	1	35.0	1	0	53.1000
6	7	0	1	54.0	0	0	51.8625
10	11	1	3	4.0	1	1	16.7000
11	12	1	1	58.0	0	0	26.5500

In [41]:
```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth=5)


cross_validation_results = cross_val_score(clf,
                                           original[original.columns[original.columns != 'Survived']],
                                           original['Survived'], cv=6)
(cross_validation_results.mean(), cross_validation_results.std())
```

Out[41]:
```
(0.48870967741935484, 0.16332615999783529)
```

In [42]:
```
cross_validation_results
```

Out[42]:
```
array([ 0.35483871,  0.4516129 ,  0.22580645,  0.56666667,  0.7       ,
        0.63333333])
```

Tramtadada .... vysledok pre nase kombinovane data

In [43]:
```
# pricapime naspat info o triedach
polynomial['Survived'] = titanic_y
polynomial.head()
```

Out[43]:

	0	1	2	3	4	5	6	7	8	9	...	19	20	21	22	23	24	25	26	27	Survived
0	1.0	2.0	1.0	38.0	1.0	0.0	71.2833	4.0	2.0	76.0	...	38.0	0.0	2708.7654	1.0	0.0	71.2833	0.0	0.0	5081.308859	1
1	1.0	4.0	1.0	35.0	1.0	0.0	53.1000	16.0	4.0	140.0	...	35.0	0.0	1858.5000	1.0	0.0	53.1000	0.0	0.0	2819.610000	1
2	1.0	7.0	1.0	54.0	0.0	0.0	51.8625	49.0	7.0	378.0	...	0.0	0.0	2800.5750	0.0	0.0	0.0000	0.0	0.0	2689.718906	0
3	1.0	11.0	3.0	4.0	1.0	1.0	16.7000	121.0	33.0	44.0	...	4.0	4.0	66.8000	1.0	1.0	16.7000	1.0	16.7	278.890000	1
4	1.0	12.0	1.0	58.0	0.0	0.0	26.5500	144.0	12.0	696.0	...	0.0	0.0	1539.9000	0.0	0.0	0.0000	0.0	0.0	704.902500	1

5 rows × 29 columns

In [44]:
```
from sklearn.model_selection import cross_val_score
# clf = LogisticRegression()
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth=5)

cross_validation_results = cross_val_score(clf,
                                           polynomial[polynomial.columns[polynomial.columns != 'Survived']],
                                           polynomial['Survived'], cv=6)
(cross_validation_results.mean(), cross_validation_results.std())
```

Out[44]:
```
(0.5956989247311828, 0.096158936277069368)
```

In [45]:
```
cross_validation_results
```

Out[45]:
```
array([ 0.41935484,  0.64516129,  0.70967742,  0.66666667,  0.53333333,
        0.6       ])
```

Nanestastie tieto Polynomialne vlastnosti vyrabaju len mocniny a nasobky

Daju sa ale spravit vlastne transformatory

In [46]:
```
import numpy as np
from sklearn.preprocessing import FunctionTransformer
transformer = FunctionTransformer(np.log1p)
X = np.array([[0, 1], [2, 3]])
transformer.transform(X)
```

Out[46]:
```
array([[ 0.        ,  0.69314718],
       [ 1.09861229,  1.38629436]])
```

Varovanie na koniec

- Ked budete robit hociake operacie nad datami, tak si dajte pozor aby ste robili rovnake operacie nad trenovacou aj testovacou sadou (transformacie atributov, vytvaranie novych, filtrovanie ...)
- Dajte si pozor aby vam do trenovania nepretiekli udaje z buducnosti

> ked idem napriklad normalizovat nieco priemerom, tak hodnotu priemeru pocitam len nad trenovacimi datami a nie nad vsetkymi Pri tomto vedia velmi pomoct tzv. Pipeliny> http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html (http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html)

- ked budete normalizovat udaje, tak na normalizovanie testovacej vzorky pouzite koeficienty z trenovacej vzorky

Na co by bolo dobre sa poziret niekedy nabuduce

Co su suvisiace temy, ktore sme teraz nestihli, ale ktore sa vam mozu velmi hodit v buducnosti

Transformacia kategorickych atributov na numericke

Vyber atributov

Extrakcia atributov

Nevyvazene datasety

Pipeline