Numeric features

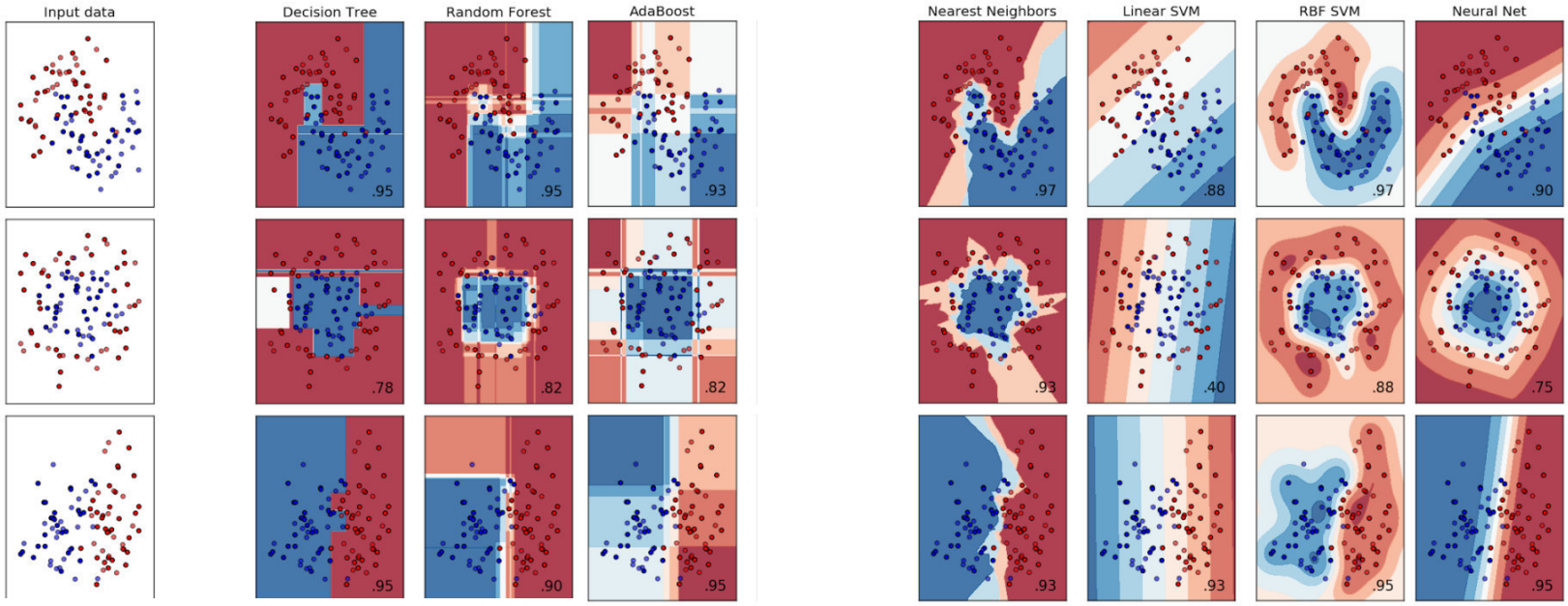
Numeric

- Preprocessing
  a) Tree-based models
  b) Non-tree-based models
- Feature generation

Preprocessing

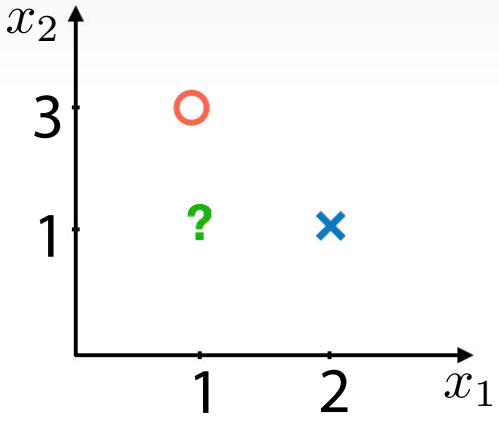
Tree-based models

Non-tree-based models

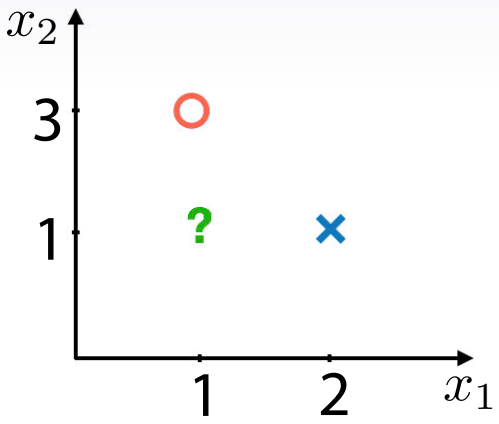


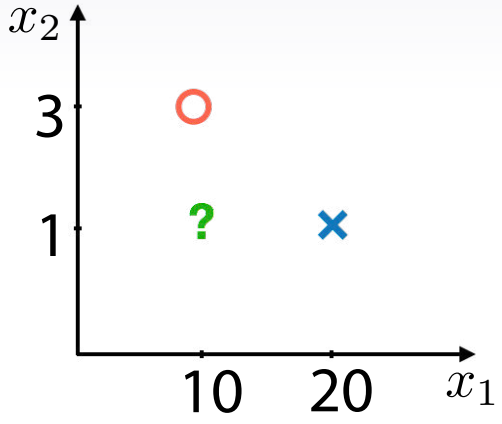
Classifier comparison¶, http://scikit - learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Preprocessing: scaling

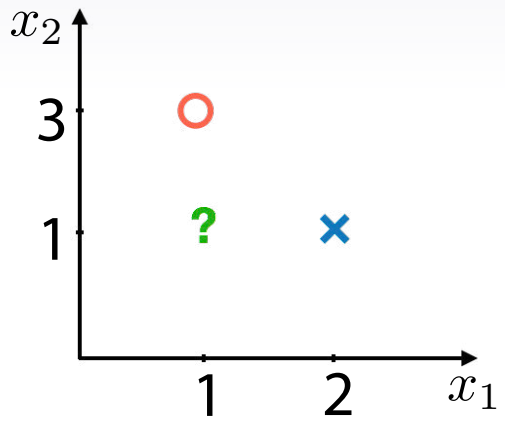


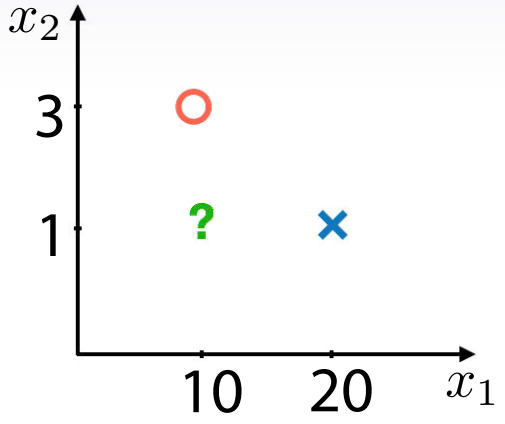
Preprocessing: scaling





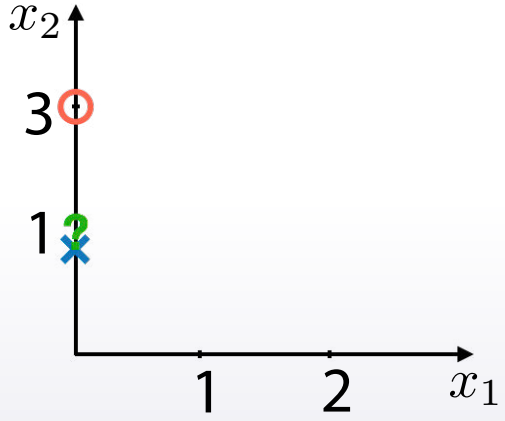
Preprocessing: scaling

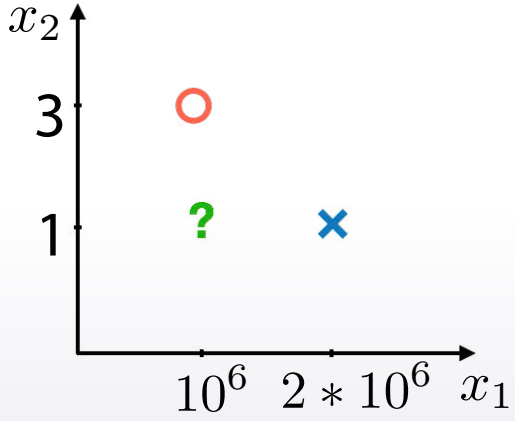




$$x_1 = x_1 * 0$$

$$x_1 = x_1 * 10^6$$





Preprocessing: scaling

1. To [0,1]

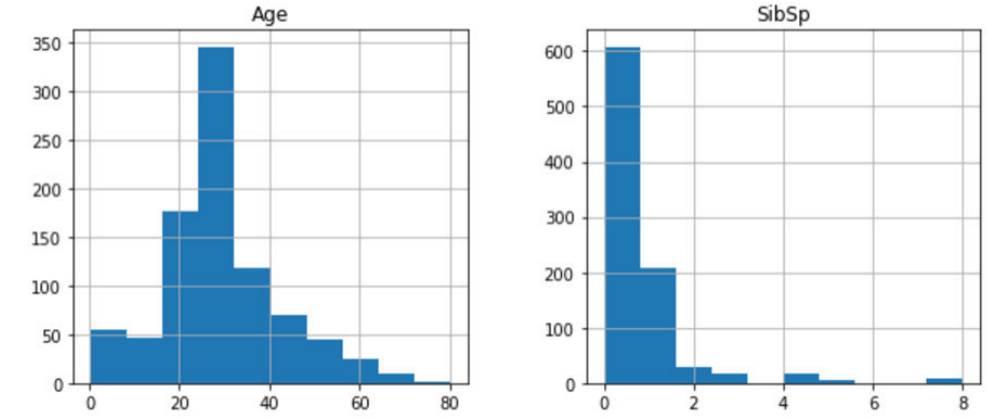
   `sklearn.preprocessing.MinMaxScaler`

$$X = (X - X.min())/(X.max() - X.min())$$

Preprocessing: scaling

$$X = (X - X.min())/(X.max() - X.min())$$

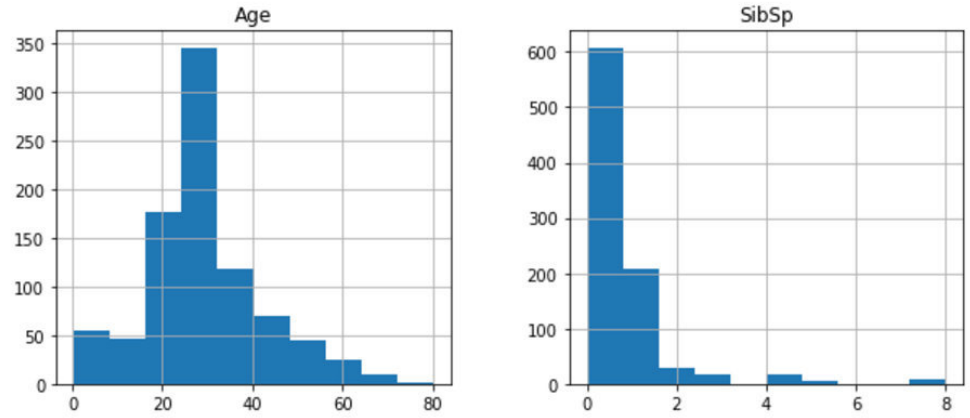
```
train[['Age', 'SibSp']].hist(figsize=(10,4));
```



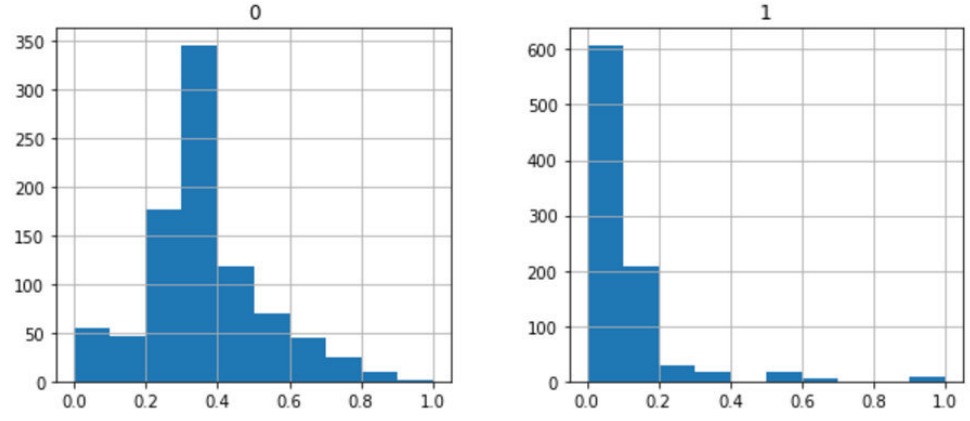
Preprocessing: scaling

$$X = (X - X.min())/(X.max() - X.min())$$

```
train[['Age', 'SibSp']].hist(figsize=(10,4));
```




```
scaler = MinMaxScaler()
xtrain = scaler.fit_transform(train[['Age', 'SibSp']])
pd.DataFrame(xtrain).hist(figsize=(10,4));
```



Preprocessing: scaling

1. To [0,1]

   `sklearn.preprocessing.MinMaxScaler`

$$X = (X - X.min())/(X.max() - X.min())$$

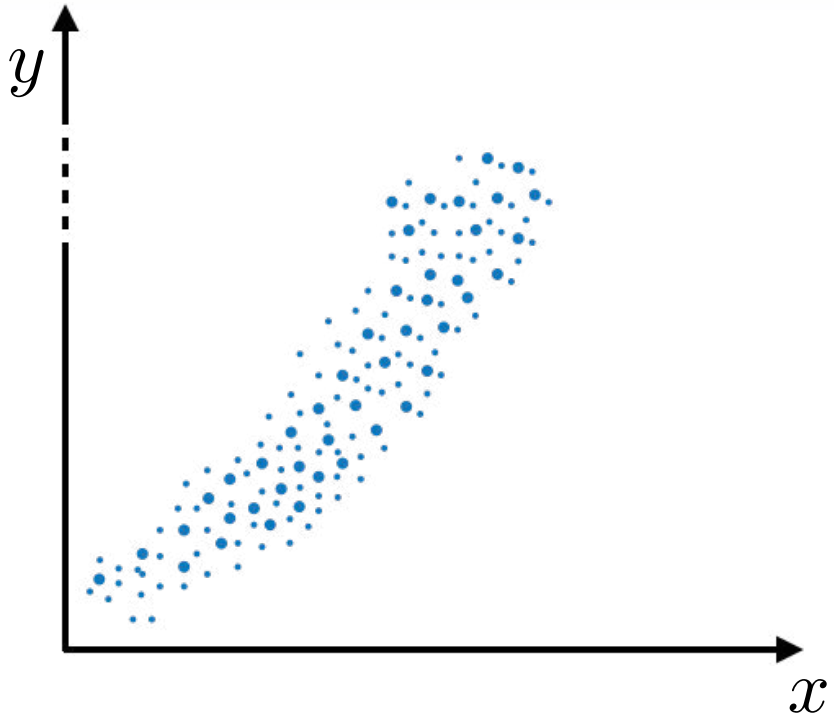
2. To mean=0, std=1

   `sklearn.preprocessing.StandardScaler`

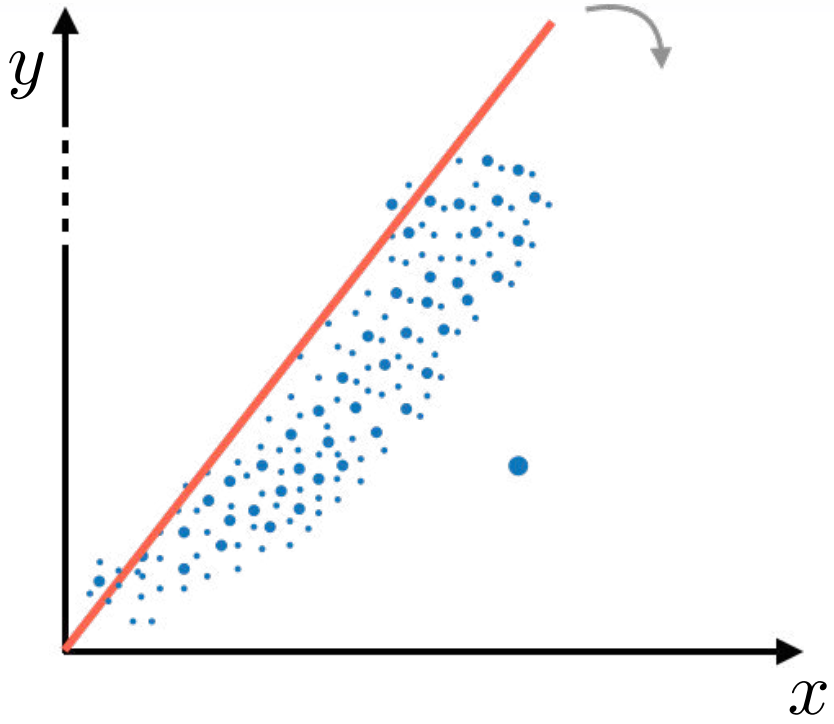
$$X = (X - X.mean())/X.std()$$

Preprocessing: outliers

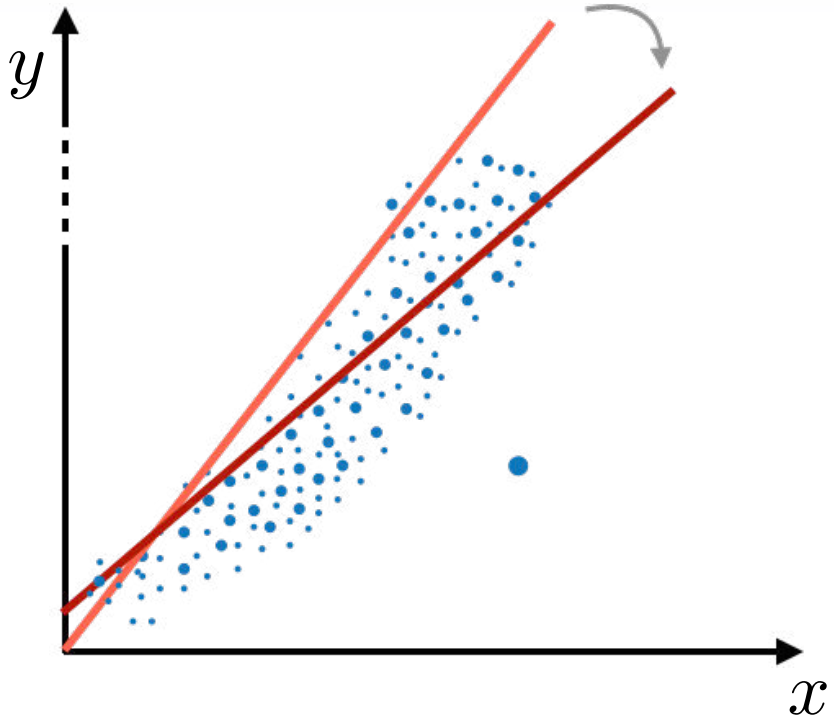
Preprocessing: outliers



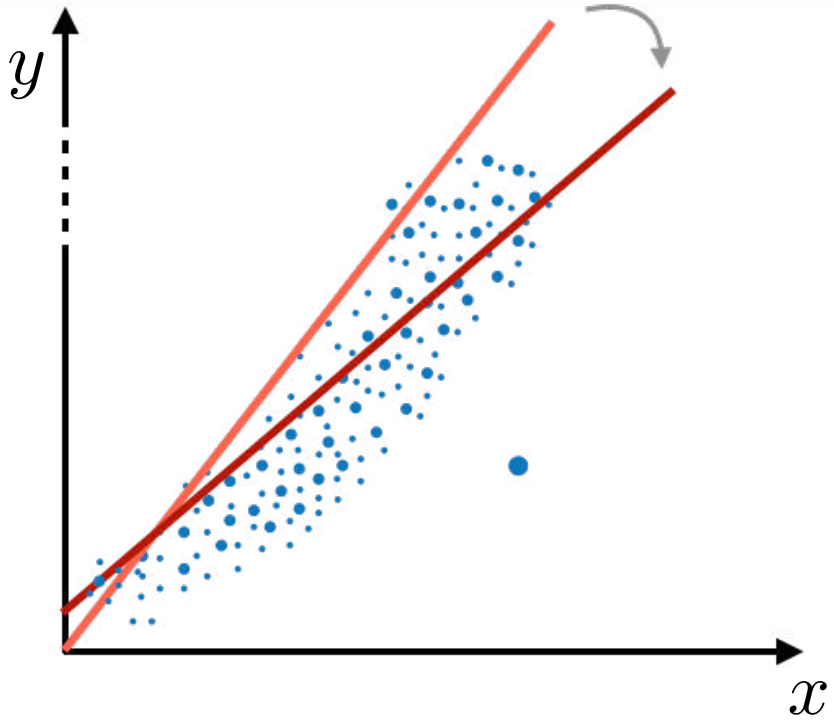
Preprocessing: outliers

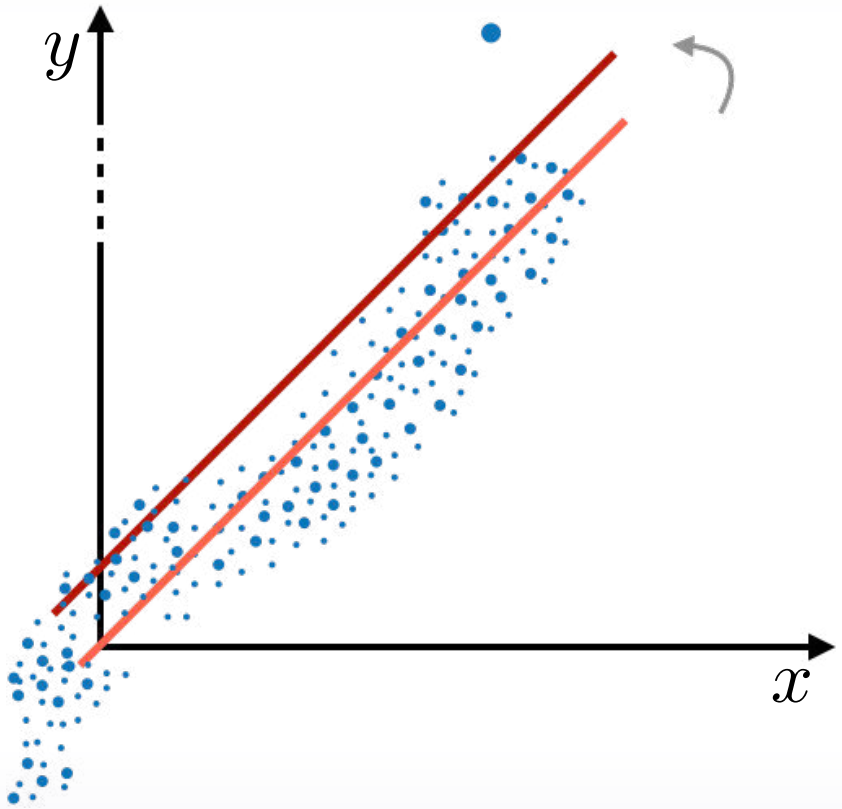


Preprocessing: outliers

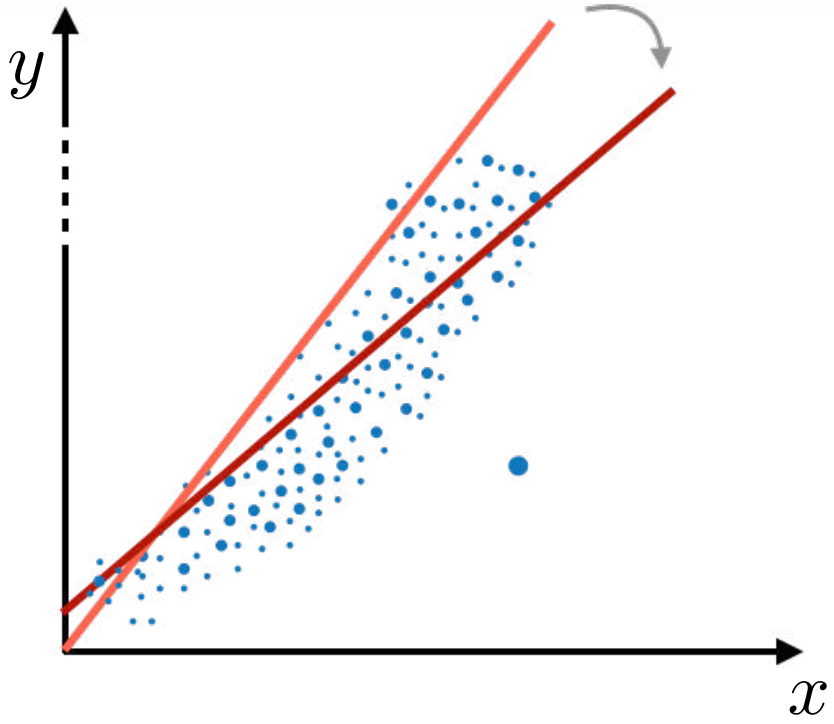


Preprocessing: outliers

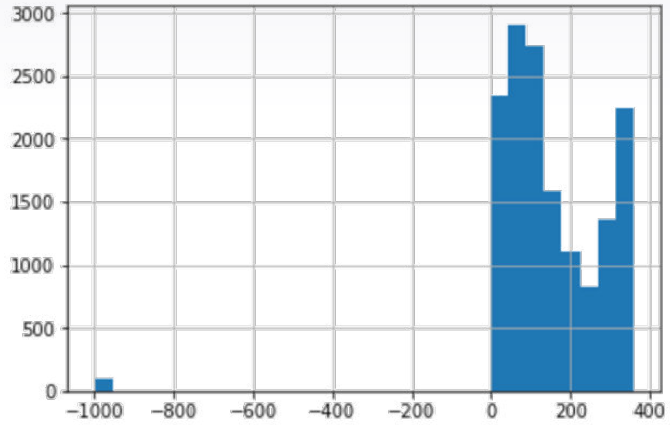




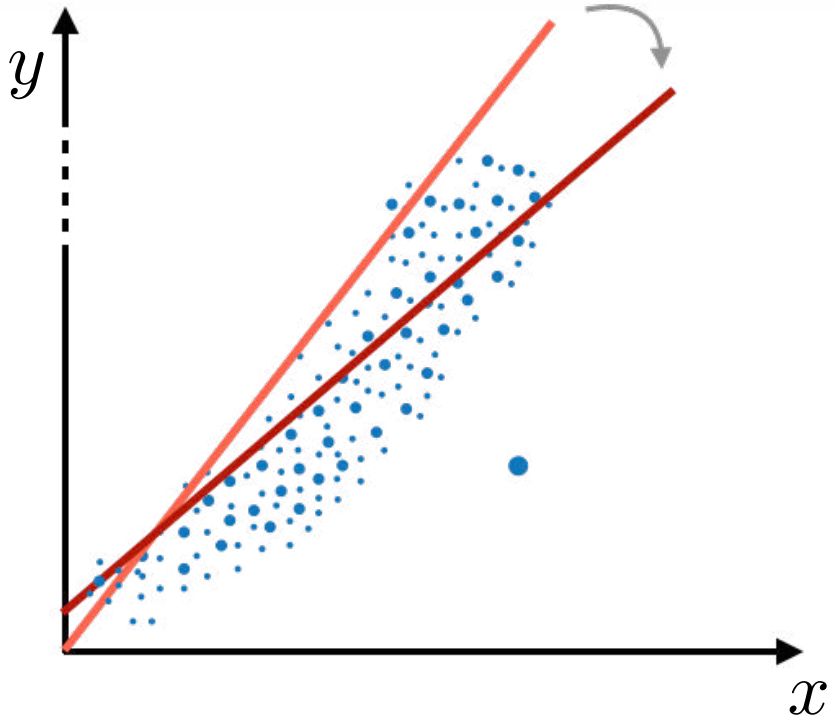
Preprocessing: outliers



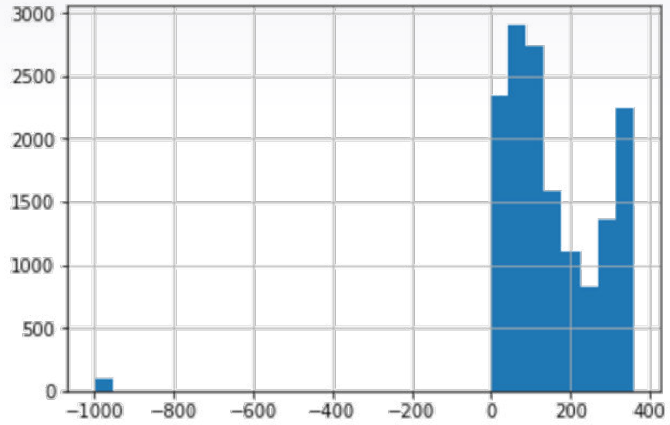

```
In [17]: pd.Series(x).hist(bins=30) ;
```



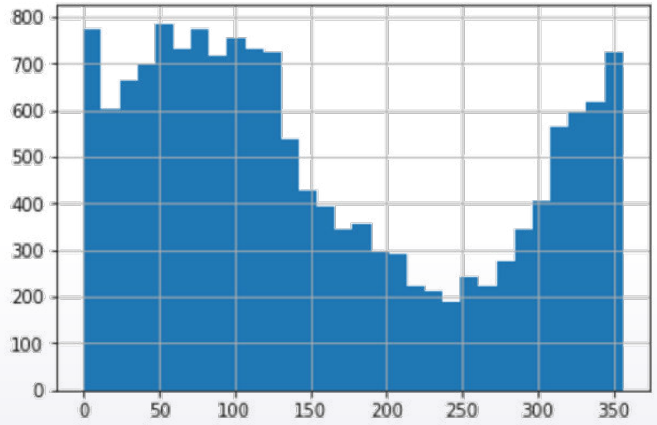
Preprocessing: outliers




```
In [17]: pd.Series(x).hist(bins=30) ;
```




```
In [18]: UPPERBOUND, LOWERBOUND = np.percentile(x, [1, 99])
y = np.clip(x, UPPERBOUND, LOWERBOUND)
pd.Series(y).hist(bins=30) ;
```



Preprocessing: rank

Preprocessing: rank

- `rank([-100, 0, 1e5]) == [0,1,2]`
- `rank([1000,1,10]) = [2,0,1]`

Preprocessing: rank

- `rank([-100, 0, 1e5]) == [0,1,2]`
- `rank([1000,1,10]) = [2,0,1]`

`scipy.stats.rankdata`

Preprocessing

1. Log transform:
   np.log(1 + x)

2. Raising to the power < 1:
   np.sqrt(x + 2/3)

Feature generation

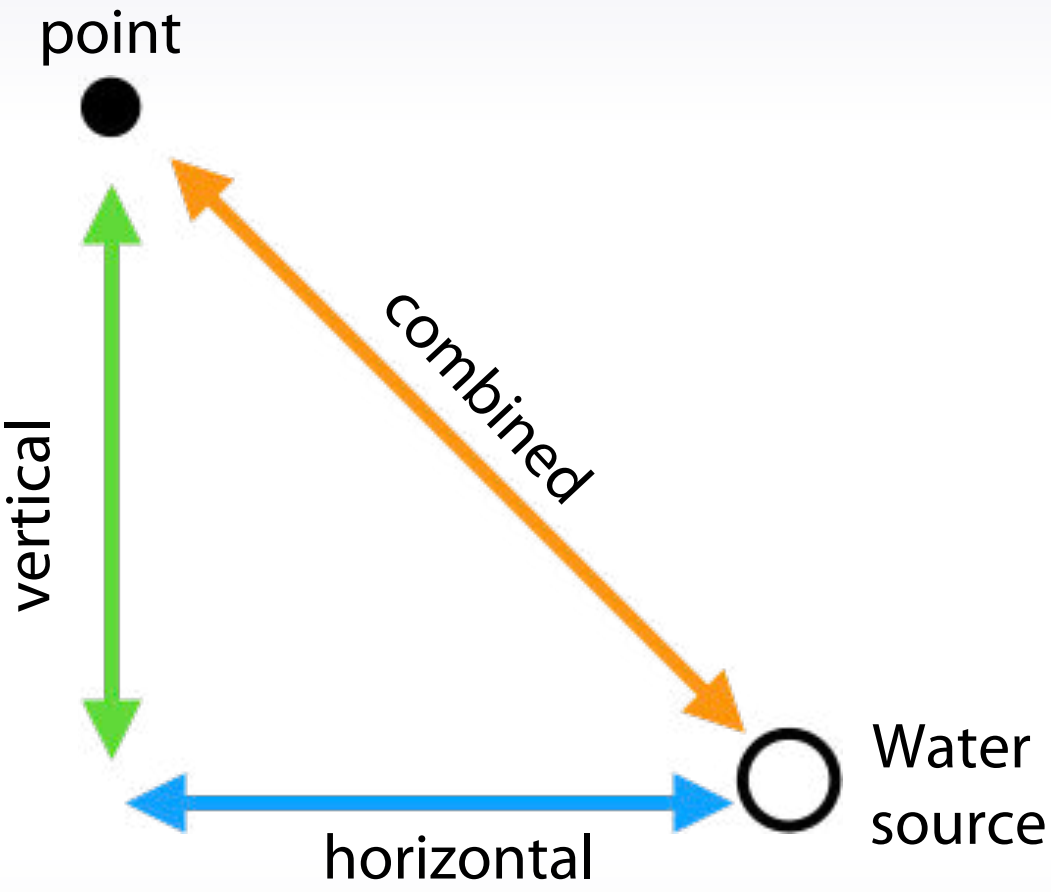
Ways to proceed:

- prior knowledge
- EDA

Feature generation

Squared area: 55 m^2
Price: 107000 $
Price for 1m2: 107000 $ / 55 m2

Feature generation



Combined = (horizontal ** 2 + vectical ** 2) ** 0.5

Feature generation

price	fractional_part
0.99	0.99
2.49	0.49
1.0	0.0
9.99	0.99

Conclusion

1. Numeric feature preprocessing is different for tree and non-tree models:
    a. Tree-based models doesn't depend on scaling
    b. Non-tree-based models hugely depend on scaling

Conclusion

1. Numeric feature preprocessing is different for tree and non-tree models:
   a. Tree-based models doesn't depend on scaling
   b. Non-tree-based models hugely depend on scaling
2. Most often used preprocessings are:
   a. MinMaxScaler - to [0,1]
   b. StandardScaler - to mean==0, std==1
   c. Rank - sets spaces between sorted values to be equal
   d. np.log(1+x) and np.sqrt(1+x)

Conclusion

1. Scaling and Rank for numeric features:
    a. Tree-based models doesn't depend on them
    b. Non-tree-based models hugely depend on them
2. Most often used preprocessings are:
    a. MinMaxScaler - to [0,1]
    b. StandardScaler - to mean==0, std==1
    c. Rank - sets spaces between sorted values to be equal
    d. np.log(1+x) and np.sqrt(1+x)
3. Feature generation is powered by:
    a. Prior knowledge
    b. Exploratory data analysis

Categorical and ordinal features

Categorical

Titanic dataset

	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	1	3	Heikkinen, Miss. Laina
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	0	3	Allen, Mr. William Henry
5	6	0	3	Moran, Mr. James
6	7	0	1	McCarthy, Mr. Timothy J
7	8	0	3	Palsson, Master. Gosta Leonard

	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S
3	female	35.000000	1	0	113803	53.1000	C123	S
4	male	35.000000	0	0	373450	8.0500	NaN	S
5	male	29.699118	0	0	330877	8.4583	NaN	Q
6	male	54.000000	0	0	17463	51.8625	E46	S
7	male	2.000000	3	1	349909	21.0750	NaN	S

Ordinal features

Ticket class: 1,2,3

Driver's license: A, B, C, D

Education: kindergarden, school, undergraduate, bachelor, master, doctoral

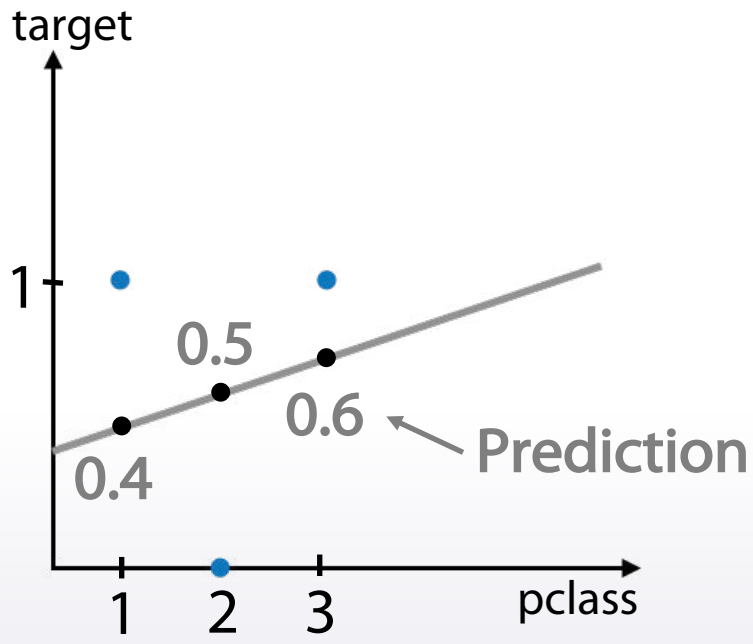
Label encoding

Label encoding

pclass	1	2	3
target	1	0	1

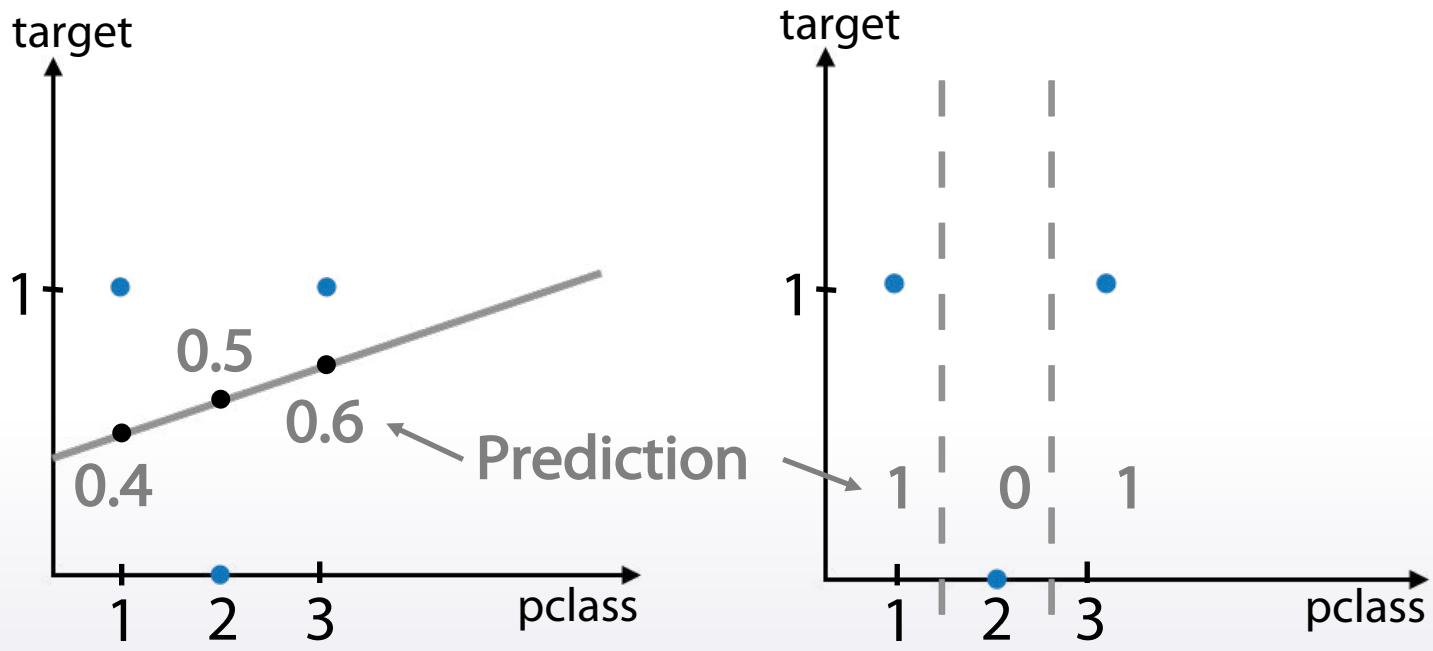
Label encoding

pclass	1	2	3
target	1	0	1



Label encoding

pclass	1	2	3
target	1	0	1



Label encoding

	K
	embarked
	S
	C
	S
	S
	S
	Q
	S
	S
	S
	C
	S
	S

1. Alphabetical (sorted)
   [S,C,Q] -> [2, 1, 3]

   `sklearn.preprocessing.LabelEncoder`

2. Order of appearance
   [S,C,Q] -> [1, 2, 3]

   `Pandas.factorize`

Frequency encoding

	K
	embarked
	S
	C
	S
	S
	S
	Q
	S
	S
	S
	C
	S
	S

[S,C,Q] -> [0.5, 0.3, 0.2]

```
encoding = titanic.groupby('Embarked').size()
encoding = encoding/len(titanic)
titanic['enc'] = titanic.Embarked.map(encoding)
```

Frequency encoding

	K
	embarked
	S
	C
	S
	S
	S
	Q
	S
	S
	S
	C
	S
	S

[S,C,Q] -> [0.5, 0.3, 0.2]

```
encoding = titanic.groupby('Embarked').size()
encoding = encoding/len(titanic)
titanic['enc'] = titanic.Embarked.map(encoding)
```

```
from scipy.stats import rankdata
```

Categorical features

One-hot encoding

pclass
1
2
1
3

pclass==1	pclass==2	pclass==3
1		
	1	
1		
		1

`pandas.get_dummies`, `sklearn.preprocessing.OneHotEncoder`

Categorical features

pclass	sex	pclass_sex
3	male	3male
1	female	1female
3	female	3female
1	female	1female

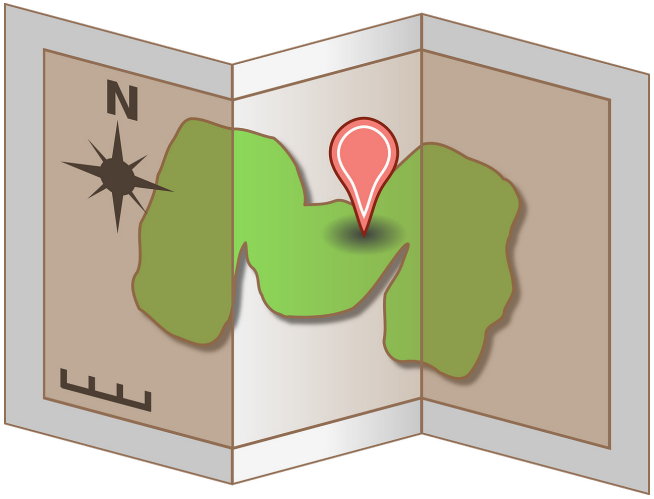
Pclass_sex==					
1male	1female	2male	2female	3male	3female
				1	
	1				
					1
	1				

Categorical features

1. Values in ordinal features are sorted in some meaningful order
2. Label encoding maps categories to numbers
3. Frequency encoding maps categories to their frequencies
4. Label and Frequency encodings are often used for tree-based models
5. One-hot encoding is often used for non-tree-based models
6. Interactions of categorical features can help linear models and KNN

Datetime and coordinates

Datetime and coordinates



Date and time

1. Periodicity
2. Time since
3. Difference between dates

Date and time

1. Periodicity

Day number in week, month, season, year
second, minute, hour.

2. Time since

3. Difference between dates

Date and time

1. Periodicity

Day number in week, month, season, year
second, minute, hour.

2. Time since
    a. Row-independent moment
       For example: since 00:00:00 UTC, 1 January 1970;
    b. Row-dependent important moment
       Number of days left until next holidays/ time passed after last holiday.

3. Difference between dates

Date and time

1. Periodicity

Day number in week, month, season, year
second, minute, hour.

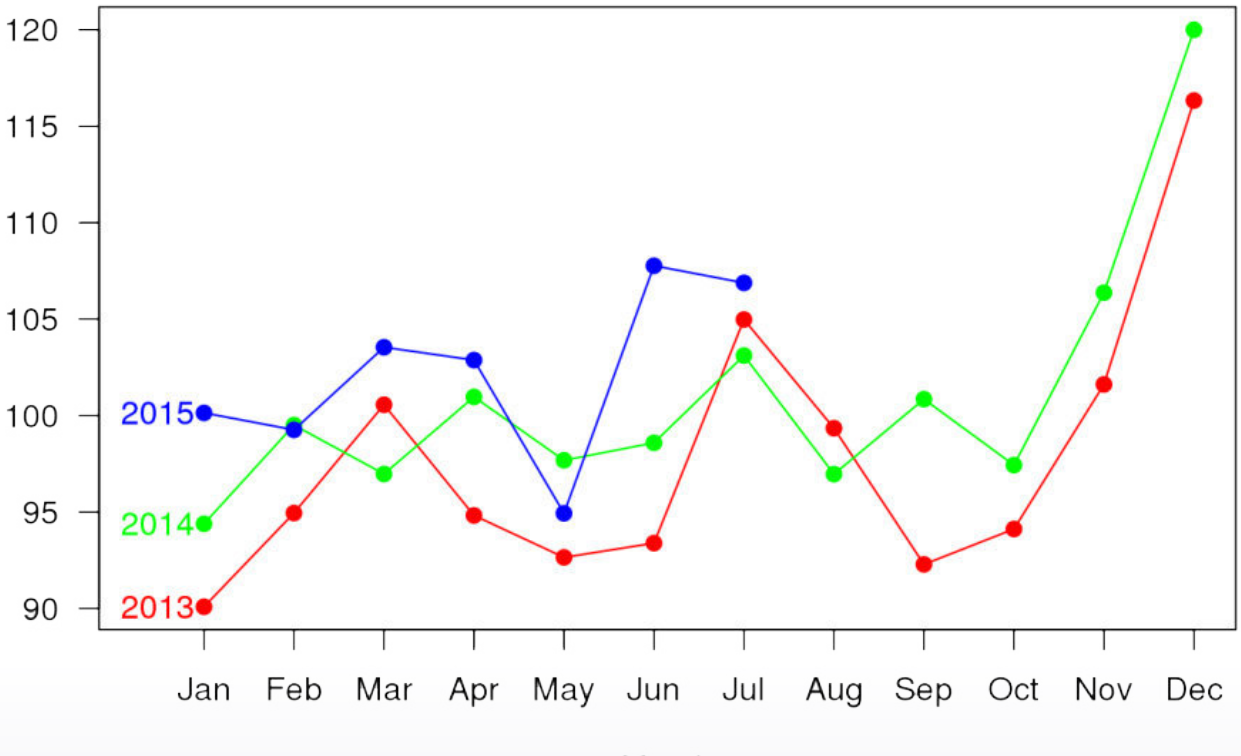
2. Time since
    a. Row-independent moment
       For example: since 00:00:00 UTC, 1 January 1970;
    b. Row-dependent important moment
       Number of days left until next holidays/ time passed after last holiday.

3. Difference between dates

datetime_feature_1 - datetime_feature_2

Date and time

ROSSMANN



Rossmann Store Sales, https://www.kaggle.com/thie1e/exploratory-analysis-rossmann

Periodicity. «Time since»

Date	weekday	daynumber_since_year_2014	is_holiday	days_till_holidays	*sales*
01.01.2014	5	0	True	0	*1213*
02.01.2014	6	1	False	3	*938*
03.01.2014	0	2	False	2	*2448*
04.01.2014	1	3	False	1	*1744*
05.01.2014	2	4	True	0	*1732*
06.01.2014	3	5	False	9	*1022*

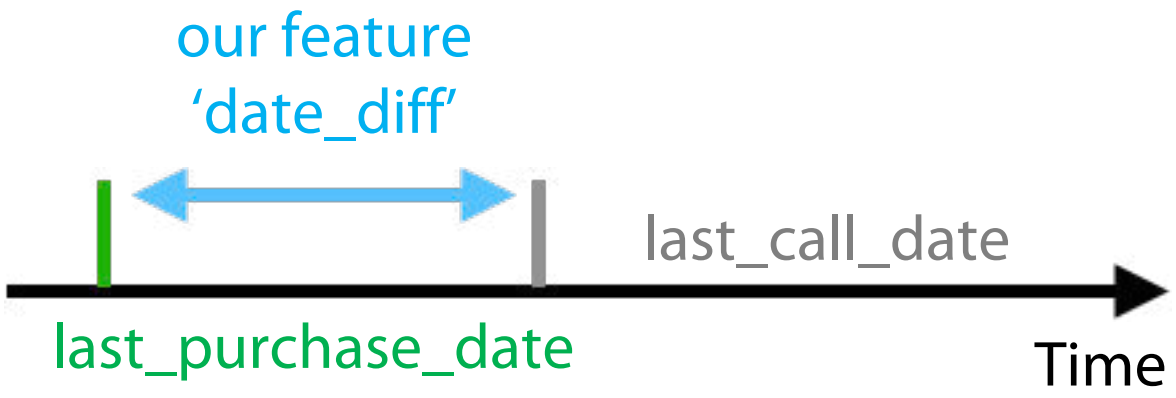
Difference between dates

Churn prediction

user_id	registration_date	*last_purchase_date*	*last_call_date*	date_diff	churn
14	10.02.2016	21.04.2016	26.04.2016	5	0
15	10.02.2016	03.06.2016	01.06.2016	-2	1
16	11.02.2016	11.01.2017	11.01.2017	1	1
20	12.02.2016	06.11.2016	08.02.2017	94	0

Difference between dates

For a particular user:



Competitions with coordinates

Western Australia Rental Prices

Deloitte.

Sberbank Russian Housing Market

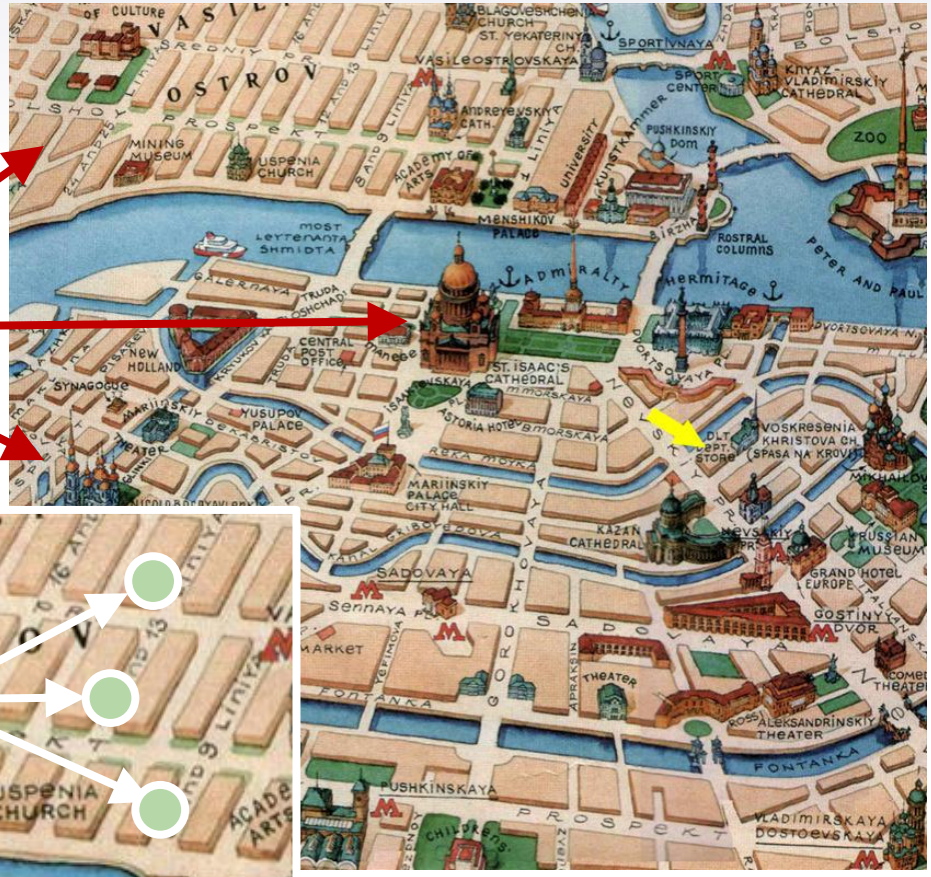
Coordinates

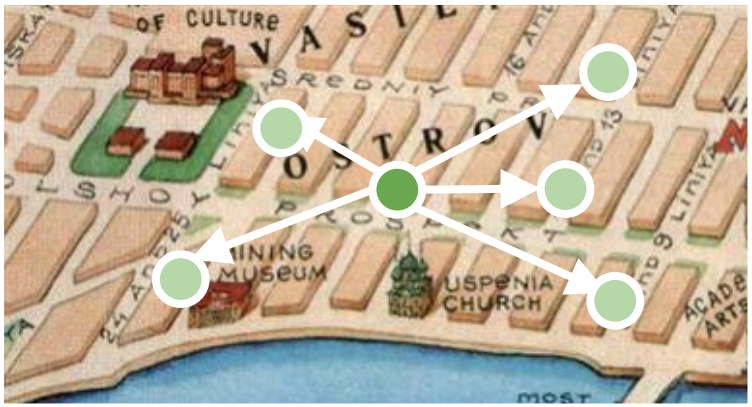
Additional data

St. Petersburg

Coordinates

Additional data



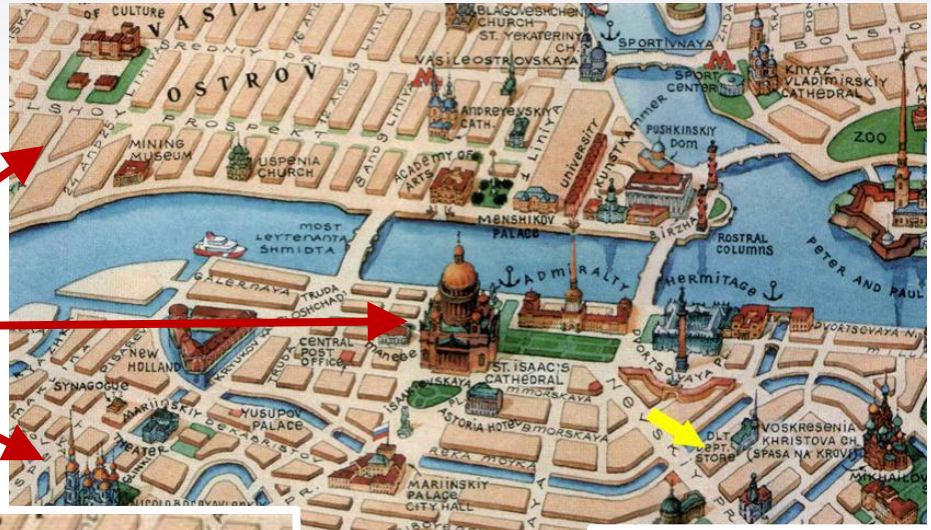


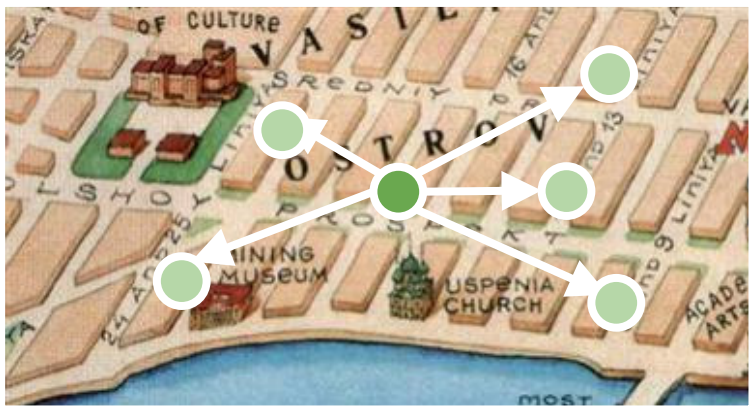
Other train samples and centers of clusters

St. Petersburg

Coordinates

Additional data



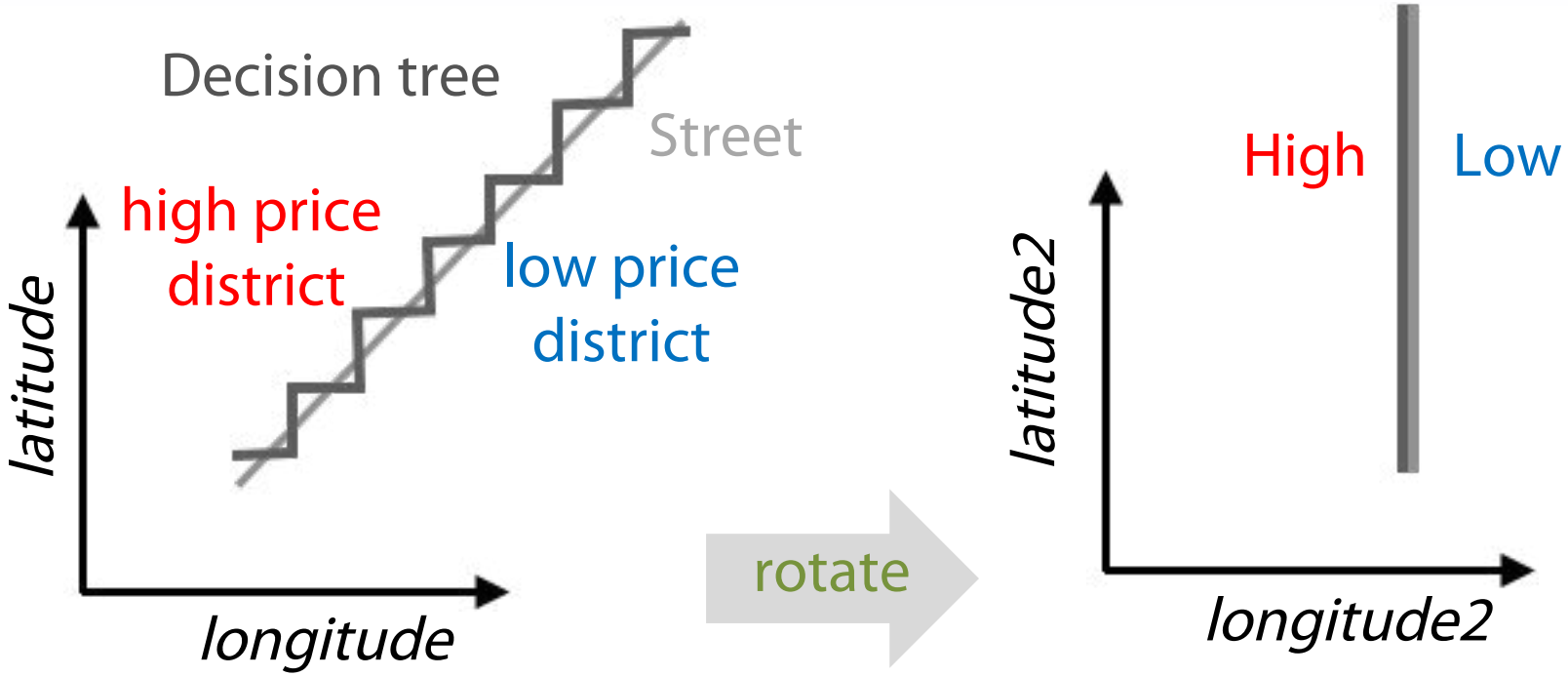


Other train samples and centers of clusters

Aggregated stats

St. Petersburg

Coordinates

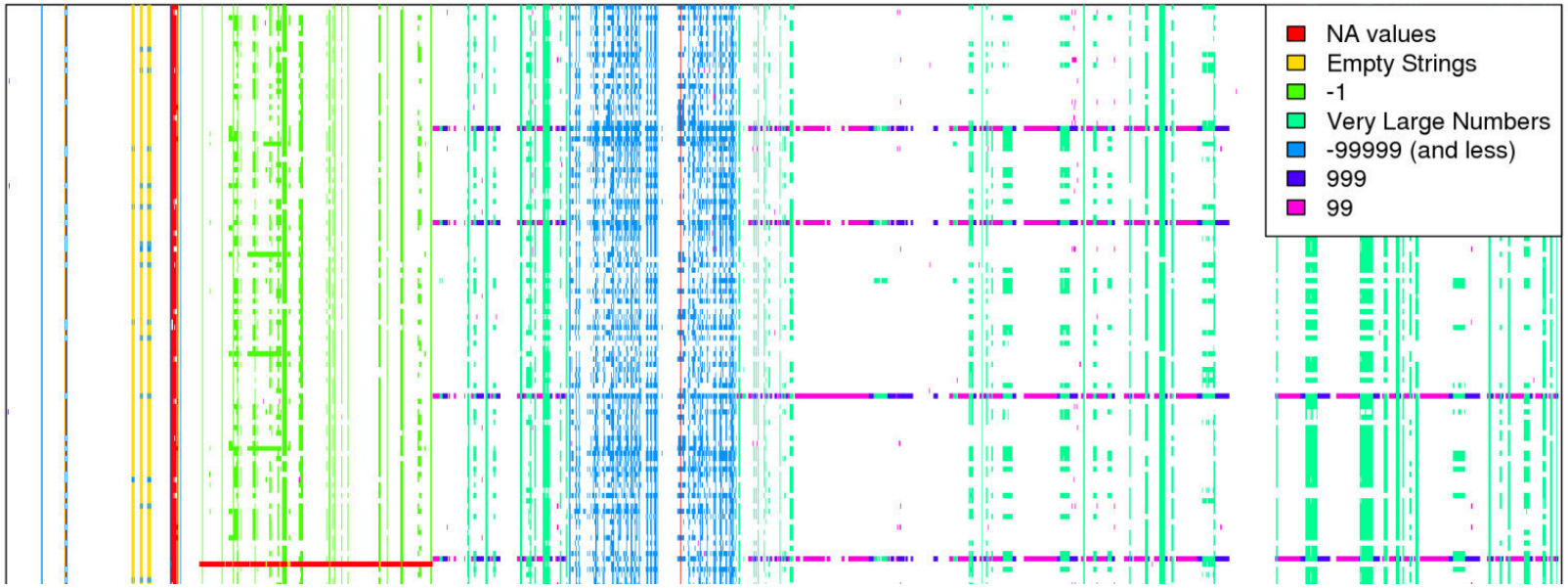


Conclusion

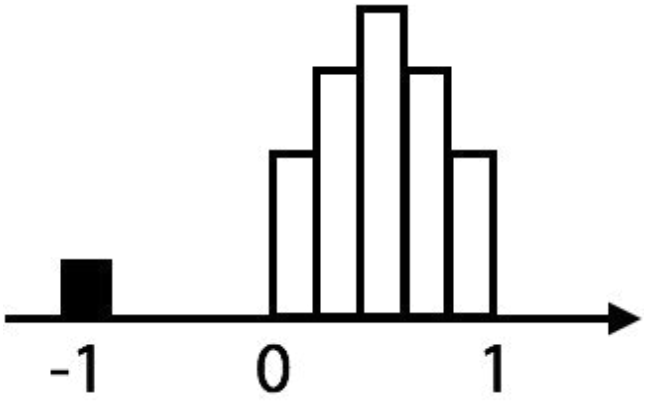
1. Datetime
    a. Periodicity
    b. Time since row-independent/dependent event
    c. Difference between dates
2. Coordinates
    a. Interesting places from train/test data or additional data
    b. Centers of clusters
    c. Aggregated statistics

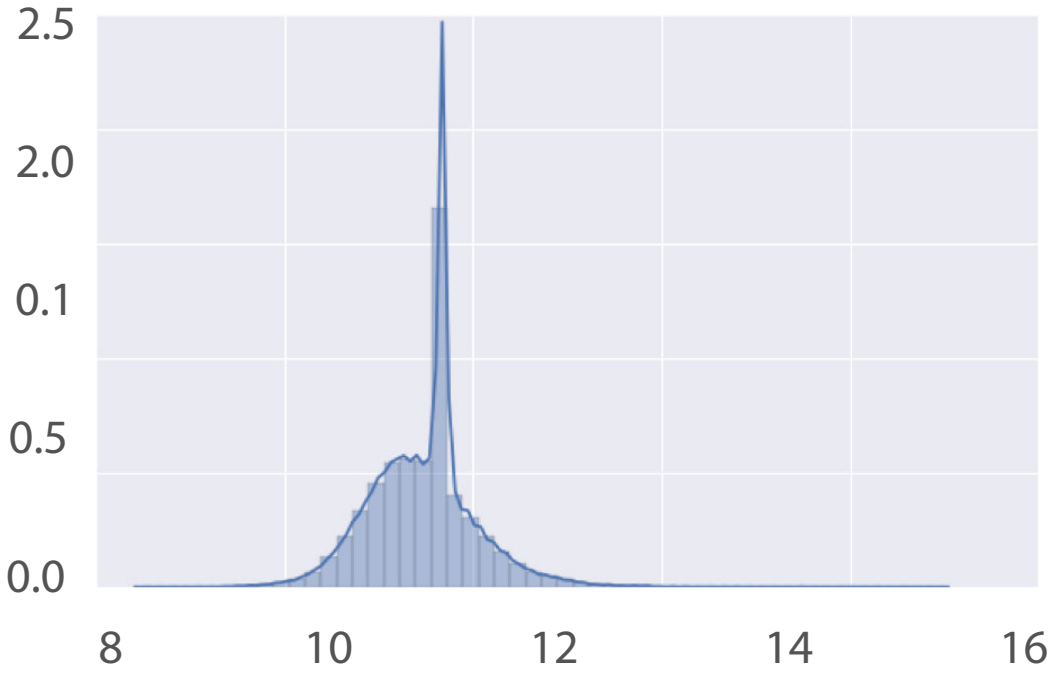
Missing values

Missing data, numeric



Hidden NaNs





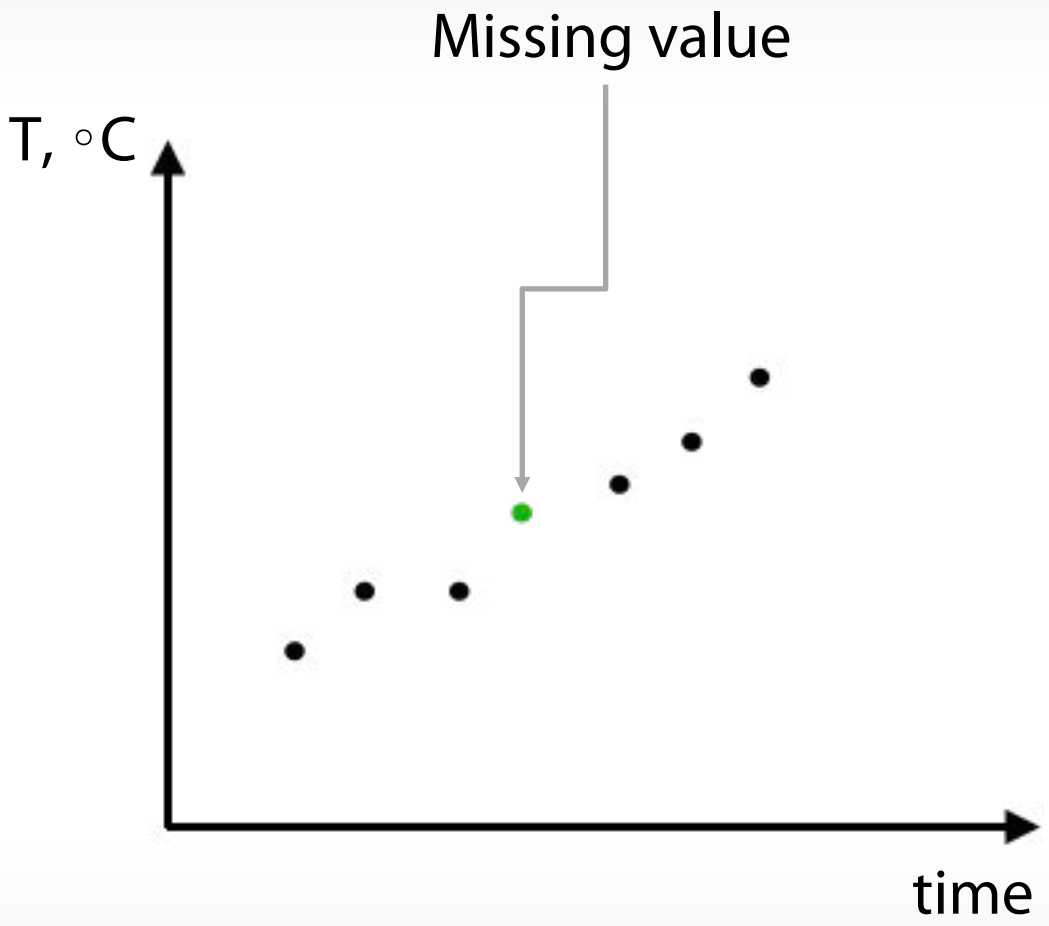
Fillna approaches

1. -999, -1, etc
2. mean, median
3. Reconstruct value

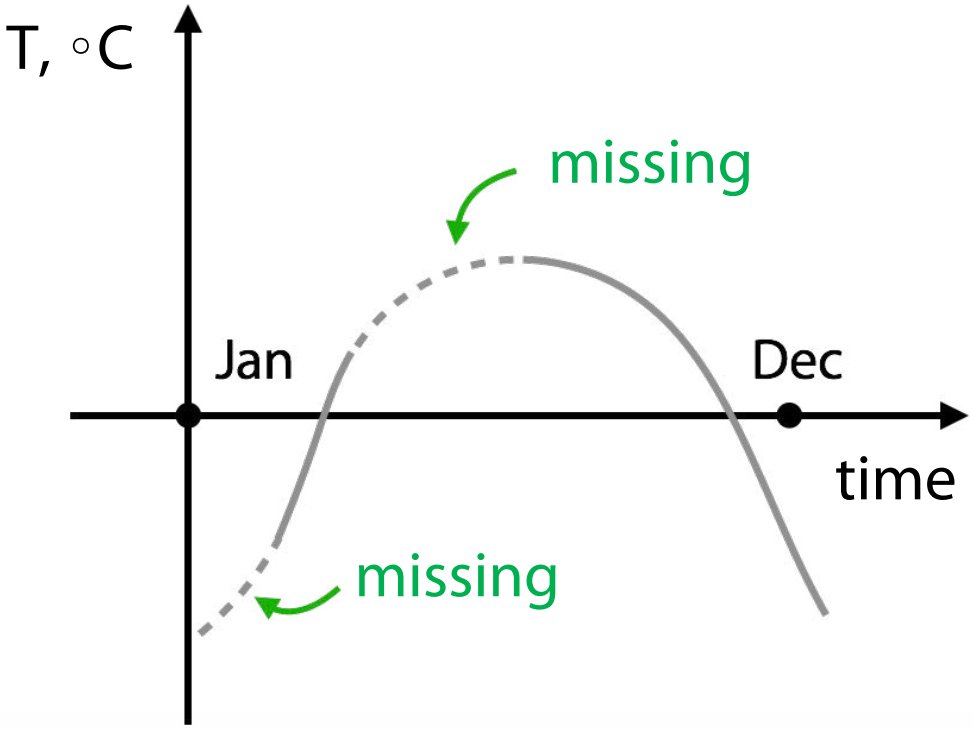
“Isnull” feature

feature	isnull
0.1	False
0.95	False
NaN	True
-3	False
NaN	True

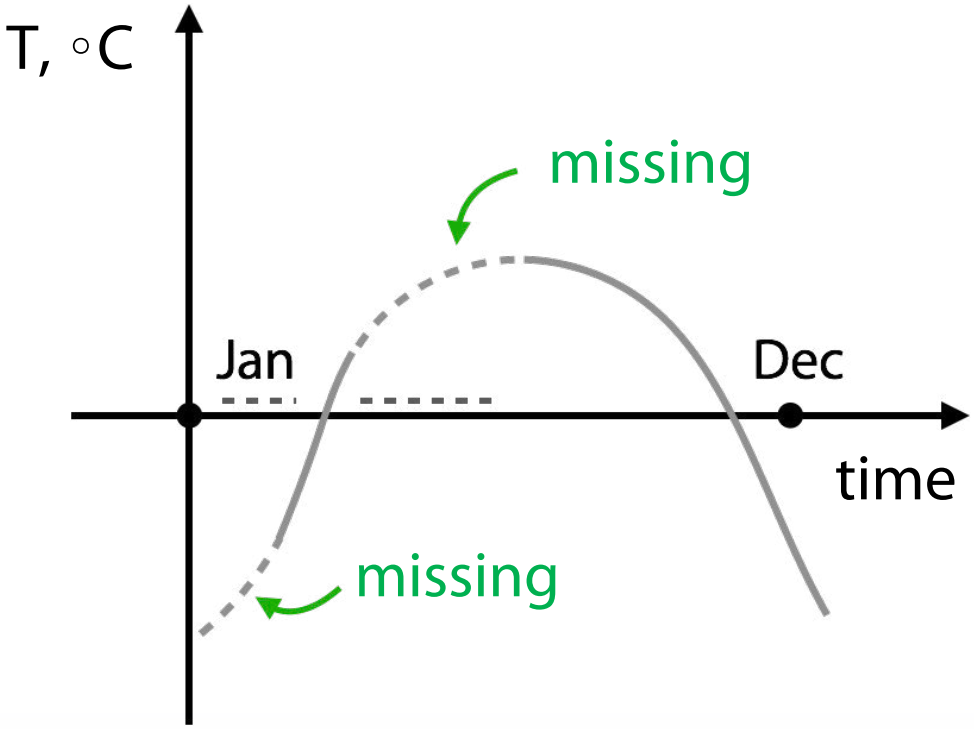
Missing values reconstruction



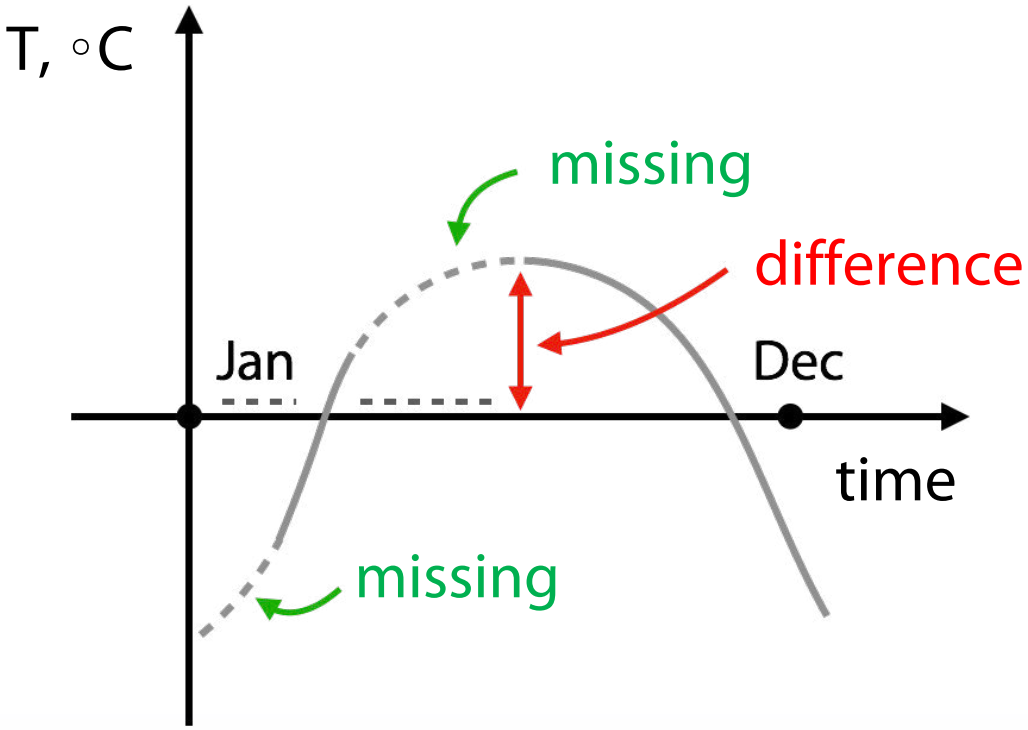
Feature generation with missing values



Feature generation with missing values



Feature generation with missing values



Feature generation with missing values

categorical_ feature	numeric _feature
A	1
A	4
A	2
A	-1
B	9
B	NaN

Feature generation with missing values

categorical_ feature	numeric _feature	numeric_ feature_filled
A	1	1
A	4	4
A	2	2
A	-1	-1
B	9	9
B	NaN	-999

Feature generation with missing values

categorical_ feature	numeric _feature	numeric_ feature_filled	categorical _encoded
A	1	1	1.5
A	4	4	1.5
A	2	2	1.5
A	-1	-1	1.5
B	9	9	-495
B	NaN	-999	-495

Treating values which do not present in train data

Train:

categorical _feature	target
A	0
A	1
A	1
A	1
B	0
B	0
D	1

Test:

categorical _feature	target
A	?
A	?
B	?
C	?

Treating values which do not present in train data

Train:

categorical _feature	categorical _encoded	target
A	6	0
A	6	1
A	6	1
A	6	1
B	3	0
B	3	0
D	1	1

Test:

categorical _feature	categorical _encoded	target
A	6	?
A	6	?
B	3	?
C	1	?

Treating values which do not present in train data

1. The choice of method to fill NaN depends on the situation
2. Usual way to deal with missing values is to replace them with -999, mean or median
3. Missing values already can be replaced with something by organizers
4. Binary feature “isnull” can be beneficial
5. In general, avoid filling nans before feature generation
6. Xgboost can handle NaN

Feature extraction from texts and images

Solely text/images competitions

Feature extraction from texts and images

Common features + text

Titanic dataset

A1 | survived

	A	B	C	D	E	F
1	survived	pclass	name	sex	age	sibsp
2	0	3	Braund, Mr. Owen Harris	male	22	1
3	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1
4	1	3	Heikkinen, Miss. Laina	female	26	0
5	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1
6	0	3	Allen, Mr. William Henry	male	35	0
7	0	3	Moran, Mr. James	male		0
8	0	1	McCarthy, Mr. Timothy J	male	54	0
9	0	3	Palsson, Master. Gosta Leonard	male	2	3
10	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0
11	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1
12	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1
13	1	1	Bonnell, Miss. Elizabeth	female	58	0

Feature extraction from texts and images

Common features + images/text

TRADESHIFT®

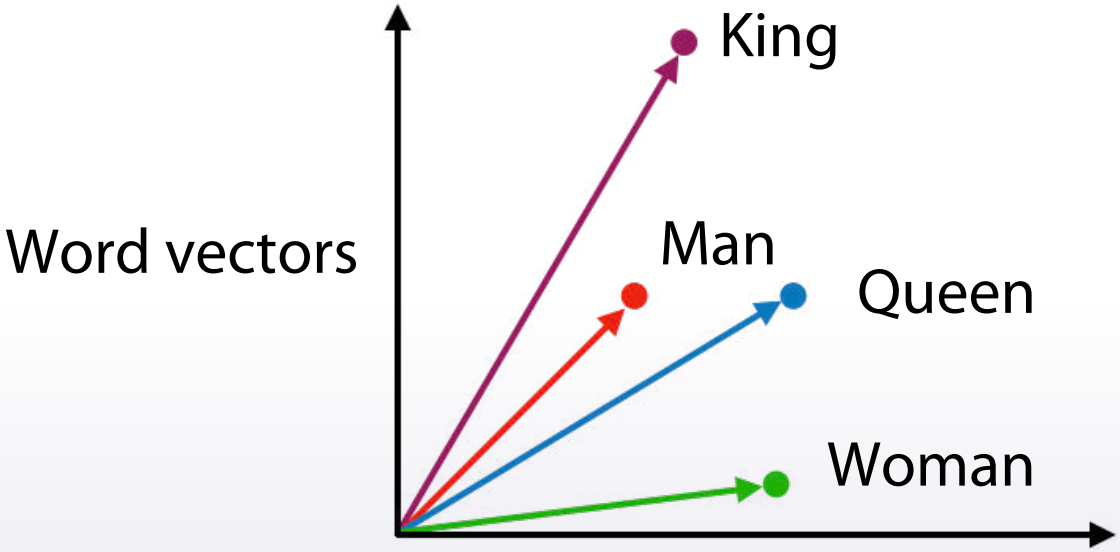
Text -> vector

1. Bag of words:

The dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

2. Embeddings (~word2vec):



Bag of words

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

CountVectorizer

hi	every one	I'm	so	excited	about	this	course
1	1			1			
		1	1	1	1	1	1
		1	2	3			

`sklearn.feature_extraction.text.CountVectorizer`

Bag of words: TFiDF

Bag of words: TFiDF

Term frequency

```
tf = 1 / x.sum(axis=1) [:,None]
x = x * tf
```

Bag of words: TFiDF

Term frequency

```
tf = 1 / x.sum(axis=1) [:,None]
x = x * tf
```

Inverse Document Frequency

```
idf = np.log(x.shape[0] / (x > 0).sum(0))
x = x * idf
```

Bag of words: TFiDF

Term frequency

tf = 1 / x.sum(axis=1) [:,None]

x = x * tf

Inverse Document Frequency

idf = np.log(x.shape[0] / (x > 0).sum(0))

x = x * idf

`sklearn.feature_extraction.text.TfidfVectorizer`

Bag of words: TF

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

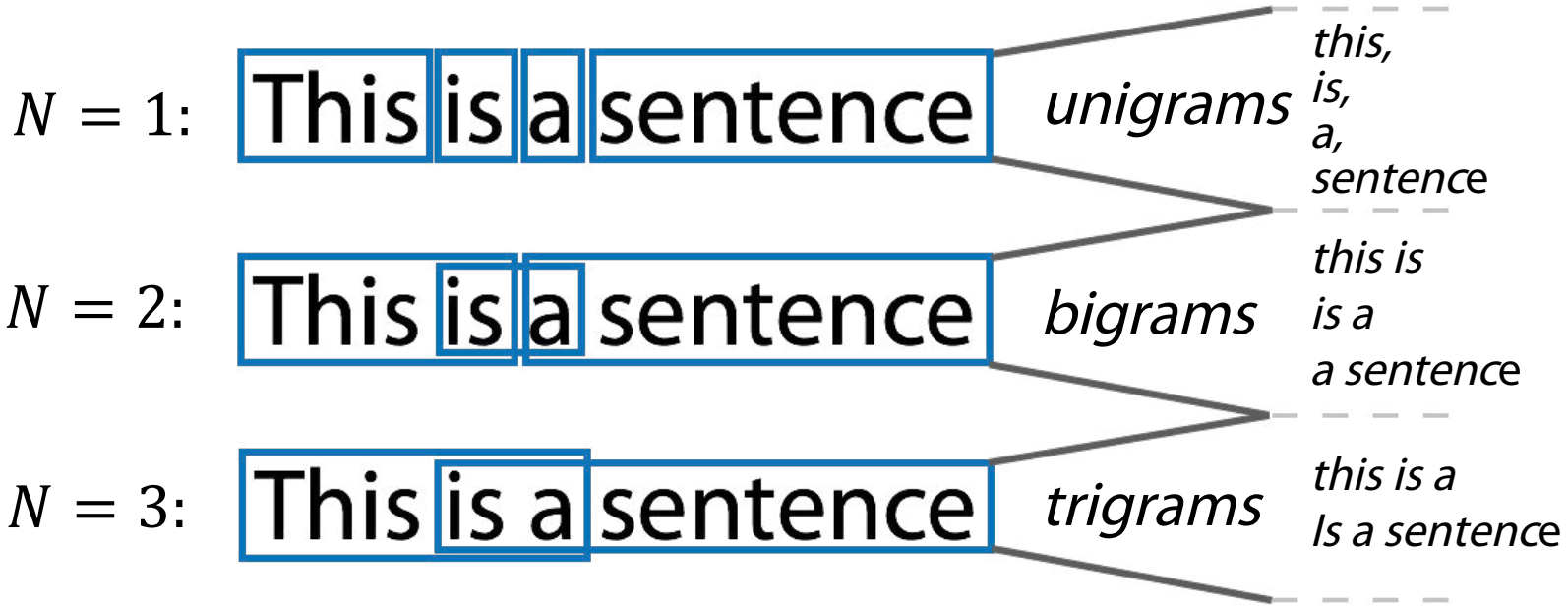
hi	every one	I'm	so	excited	about	this	course
0.33	0.33			0.33			
		0.16	0.16	0.16	0.16	0.16	0.16
		0.16	0.33	0.5			

Bag of words: TF+iDF

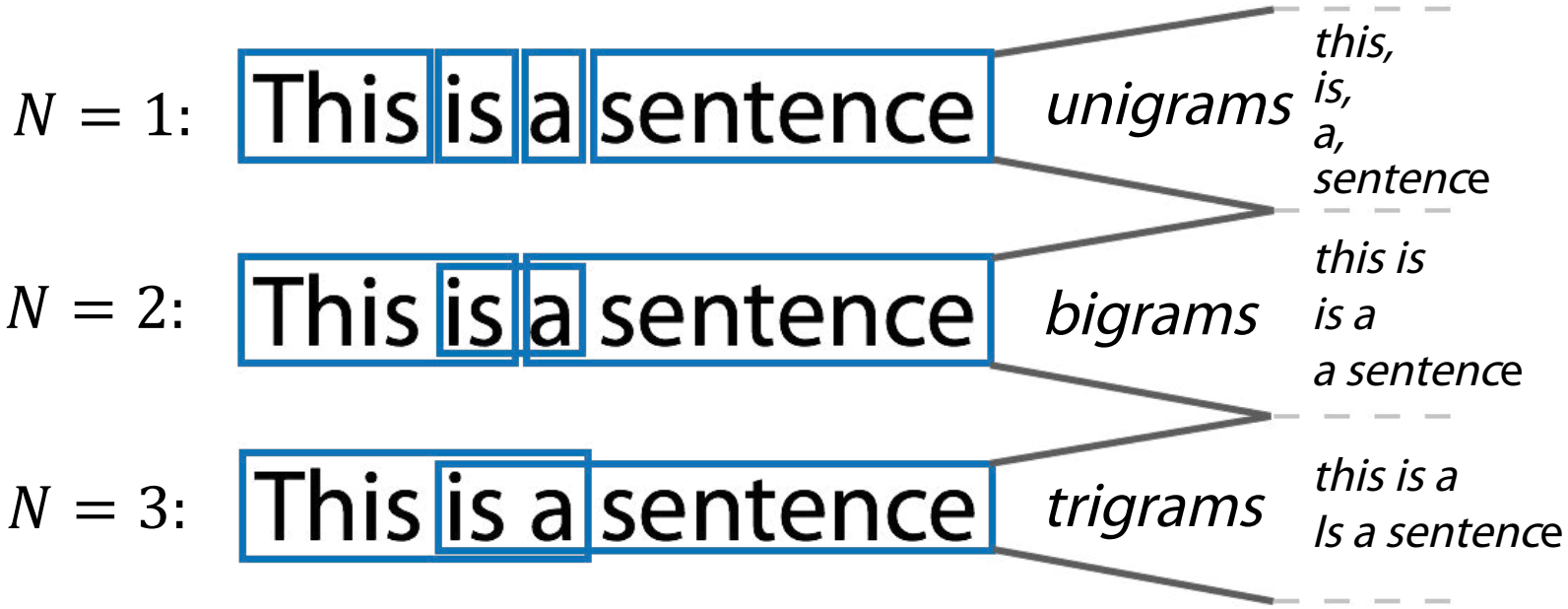
(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!

hi	every one	I'm	so	excited	about	this	course
0.36	0.36			0			
		0.06	0.06	0	0.18	0.18	0.18
		0.06	0.13	0			

N-grams


$N = 1$:
This is a sentence
unigrams
this,
is,
a,
sentence
$N = 2$:
This is a sentence
bigrams
this is
is a
a sentence
$N = 3$:
This is a sentence
trigrams
this is a
Is a sentence

N-grams



`sklearn.feature_extraction.text.CountVectorizer:`
`Ngram_range, analyzer`

Texts preprocessing

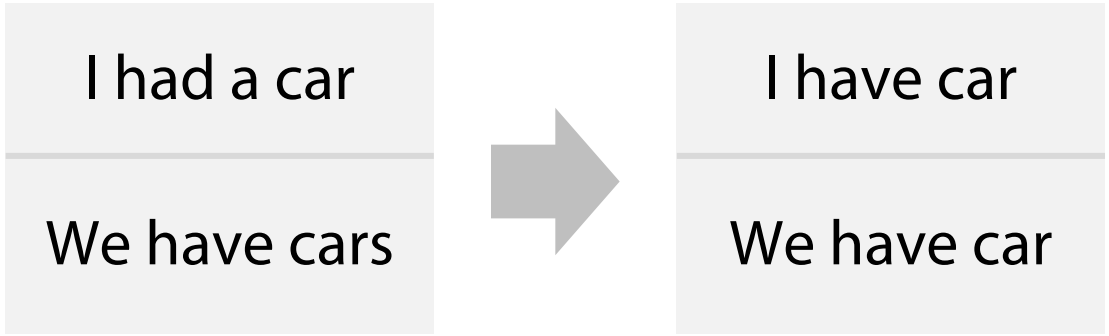
1. Lowercase
2. Lemmatization
3. Stemming
4. Stopwords

Texts preprocessing: lowercase

Very, very sunny.
Sunny... Sunny!

Very	very	Sunny	sunny
1	1	0	1
0	0	2	0

Texts preprocessing: lemmatization and stemming



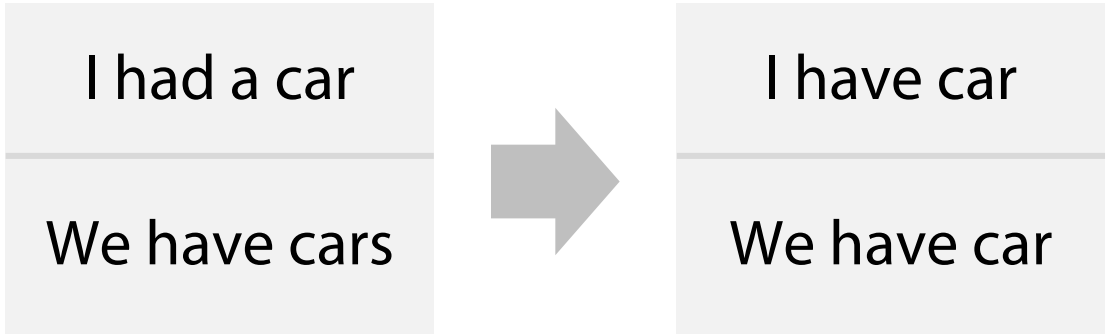
Texts preprocessing: lemmatization and stemming

I had a car	I have car
We have cars	We have car

Stemming:
democracy, democratic, and democratization -> democr

Lemmatization:
democracy, democratic, and democratization -> democracy

Texts preprocessing: lemmatization and stemming



Stemming:
democracy, democratic, and democratization -> democr

Saw -> s

Lemmatization:
democracy, democratic, and democratization -> democracy

Saw -> see or saw (depending on context)

Texts preprocessing: stopwords

Examples:

1. Articles or prepositions
2. Very common words

Texts preprocessing: stopwords

Examples:

1. Articles or prepositions
2. Very common words

NLTK, Natural Language Toolkit library for python

`sklearn.feature_extraction.text.CountVectorizer: max_df`

Conclusion

Pipeline of applying BOW

1. Preprocessing:

   Lowercase, stemming, lemmatization, stopwords

2. Bag of words:

   Ngrams can help to use local context

3. Postprocessing: TFiDF

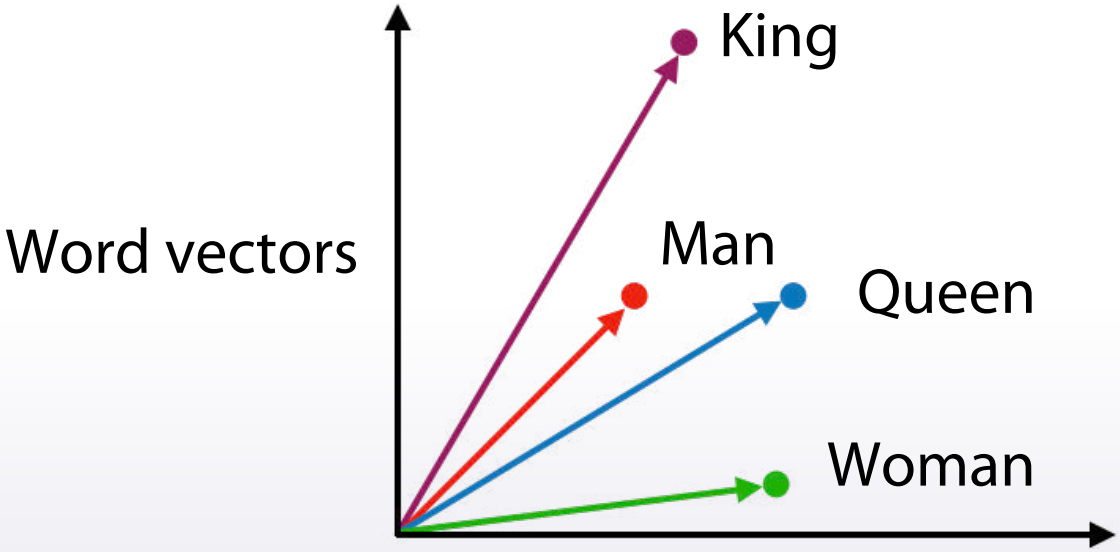
Text -> vector

1. Bag of words:

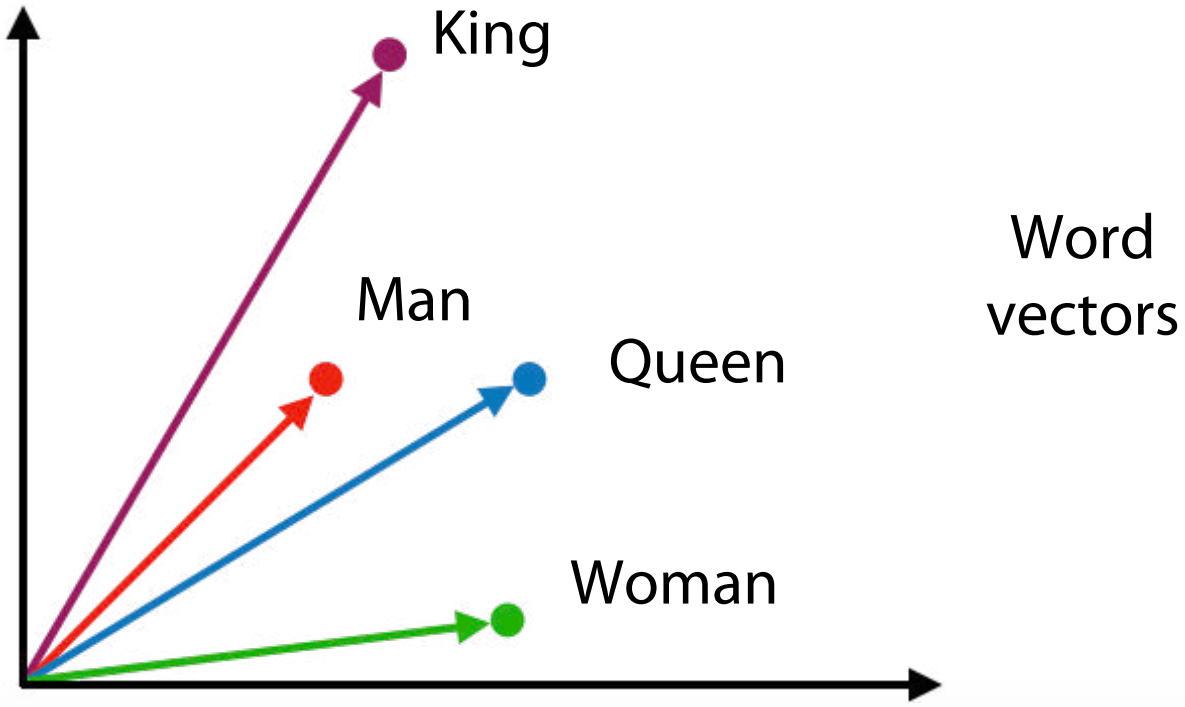
The dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

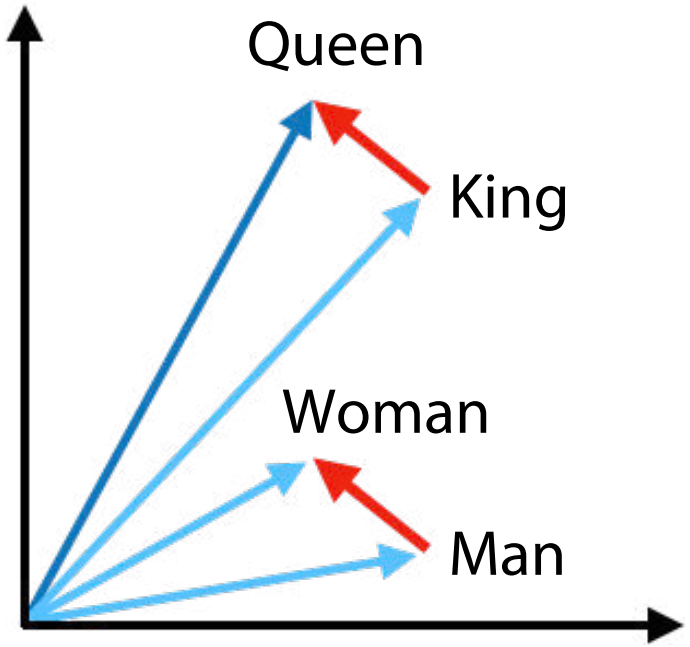
2. Embeddings (~word2vec):



Word2vec

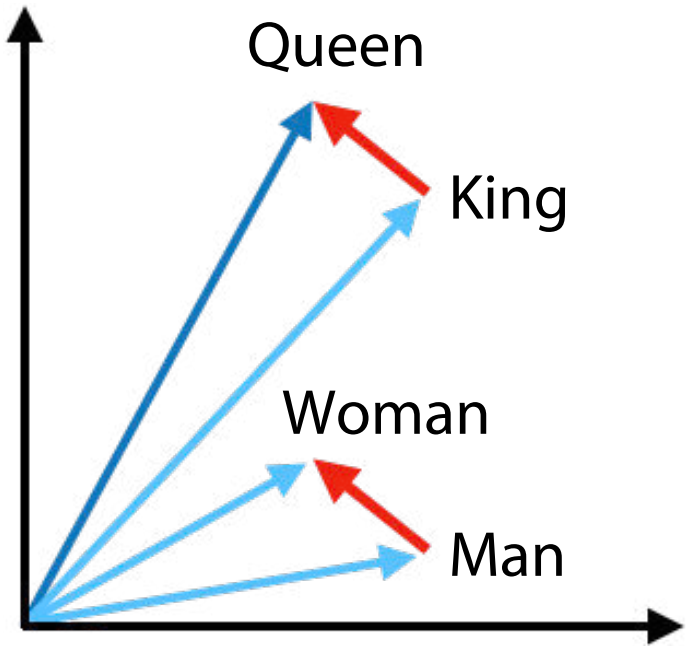


Word2vec

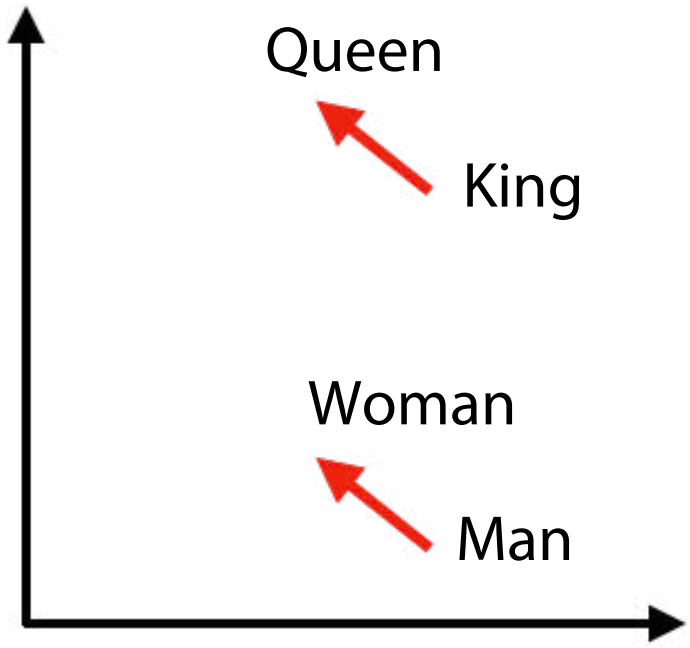


So king + man – woman = queen!

Word2vec



king + woman - man = queen



The red direction encodes gender

Word2vec

Words: Word2vec, Glove, FastText, etc

Sentences: Doc2vec, etc

There are pretrained models

BOW and w2v comparison

1. Bag of words
    a. Very large vectors
    b. Meaning of each value in vector is known
2. Word2vec
    a. Relatively small vectors
    b. Values in vector can be interpreted only in some cases
    c. The words with similar meaning often have similar embeddings

Image -> vector

1. Descriptors

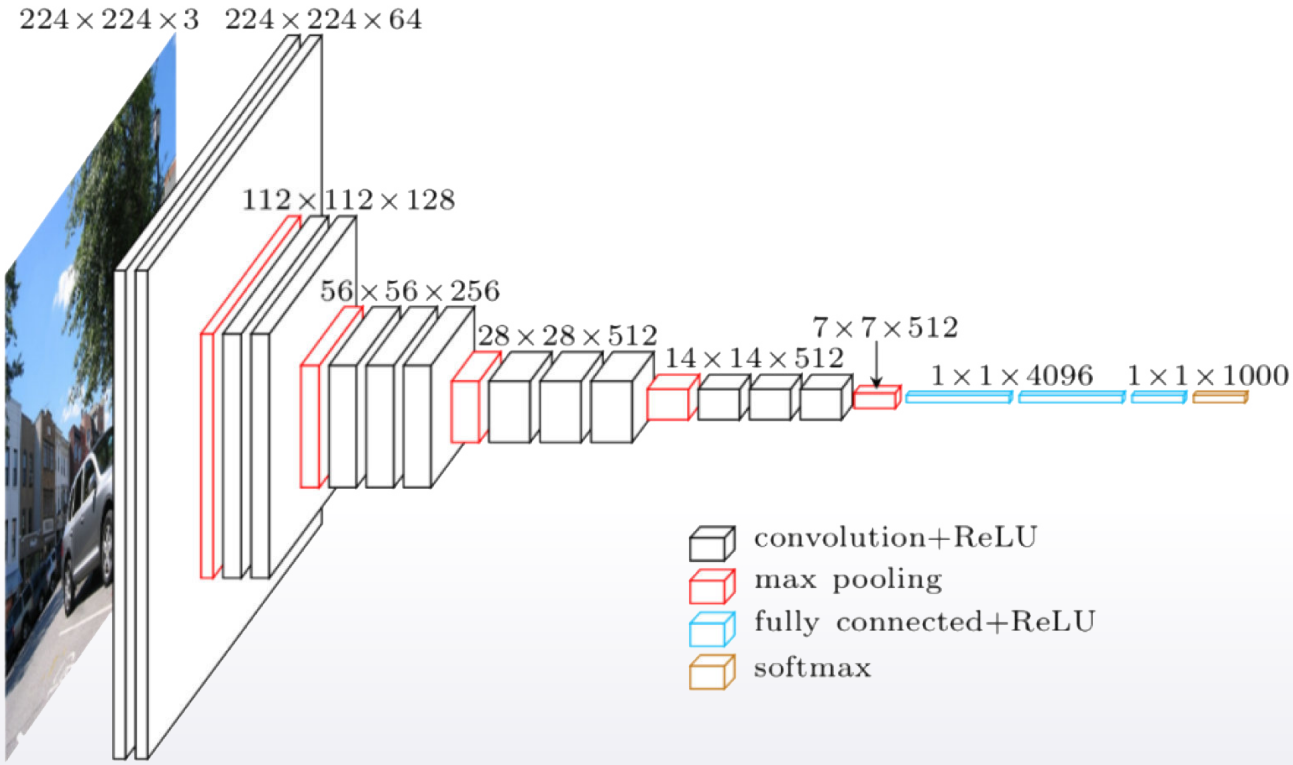
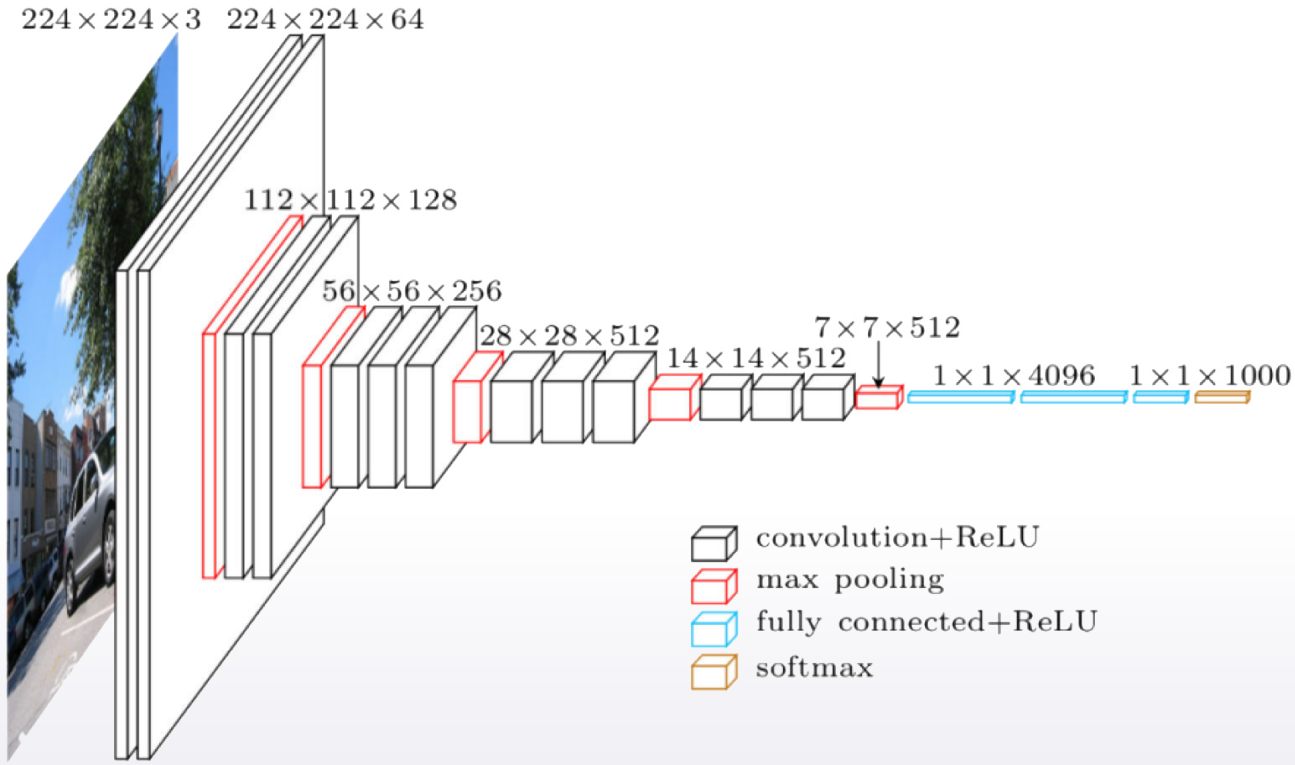


Image -> vector

1. Descriptors
2. Train network from scratch
3. Finetuning



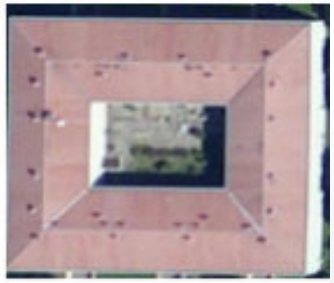
Finetuning example

Category 1:
North-South orientation

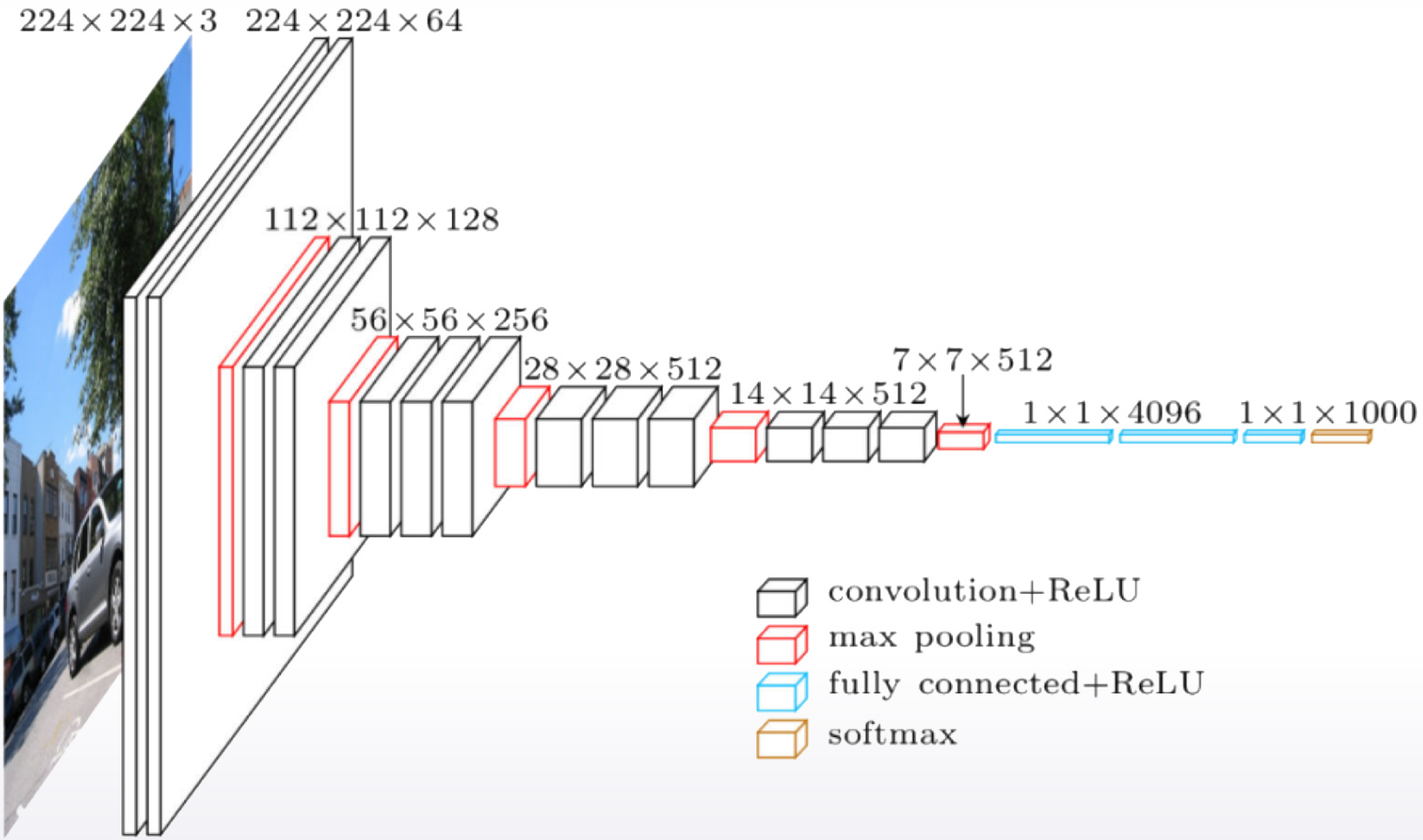
Category 2:
East-West orientation

Category 3:
Flat roof

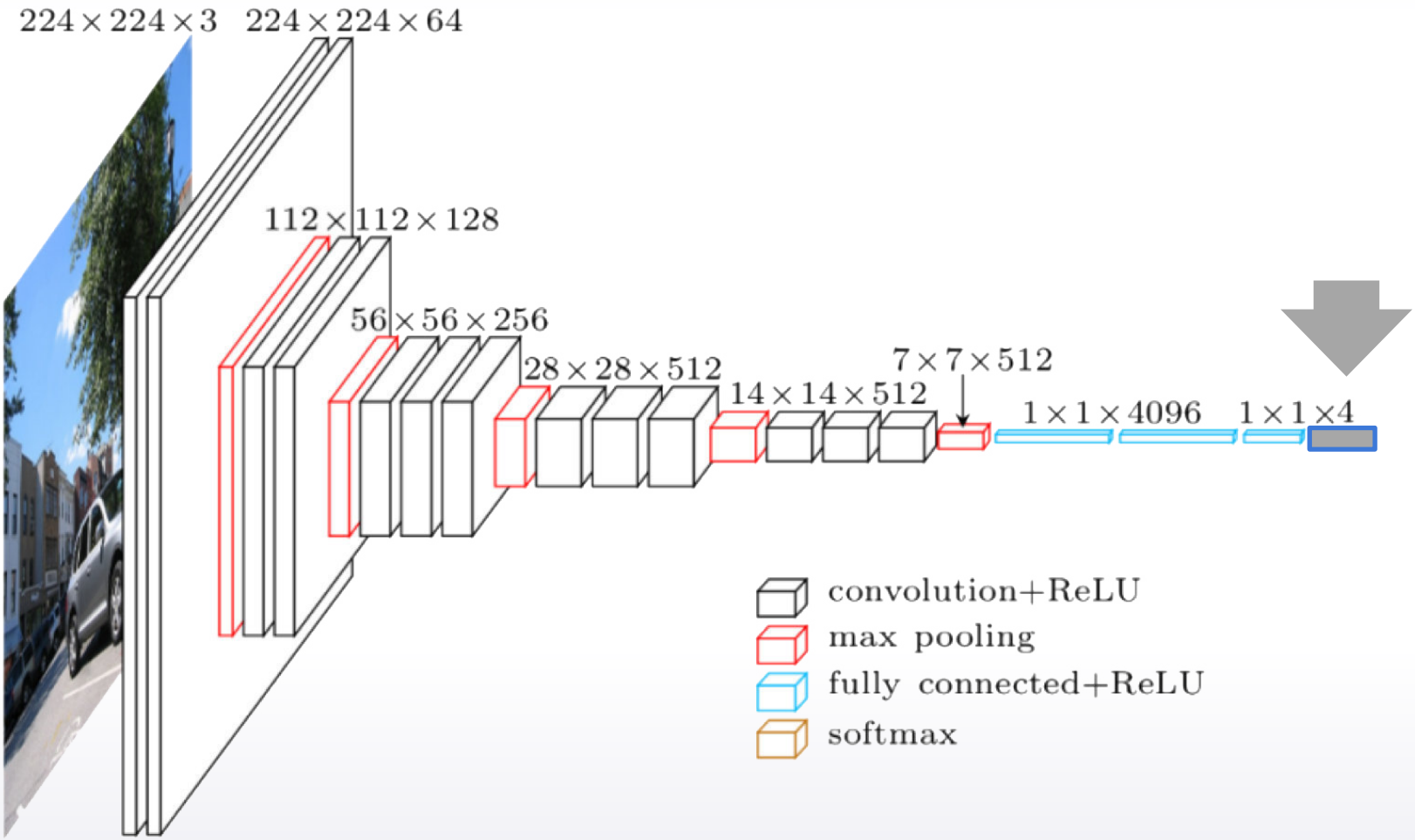
Category 4:
Other



Finetuning example



Finetuning example



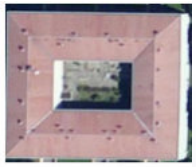
Augmentation

Category 1:
North-South orientation

Category 3:
Flat roof

Category 2:
East-West orientation

Category 4:
Other



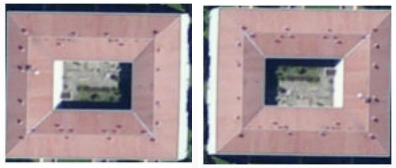
Augmentation

Category 1:
North-South orientation

Category 3:
Flat roof

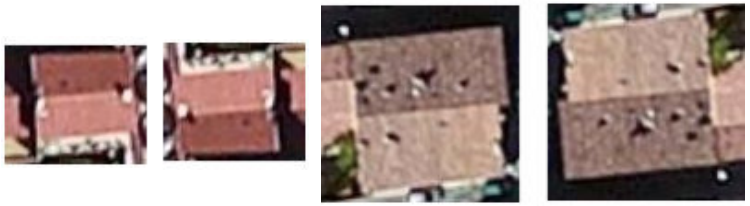
Category 2:
East-West orientation

Category 4:
Other



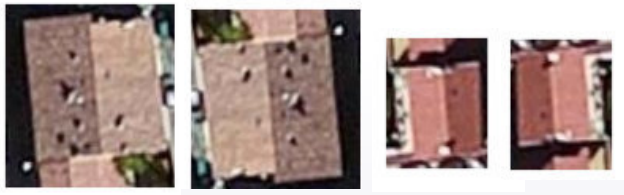
Augmentation

Category 1:
North-South orientation

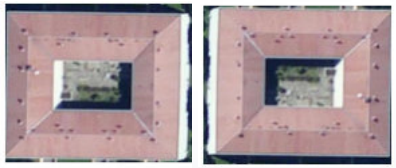


Category 3:
Flat roof

Category 2:
East-West orientation

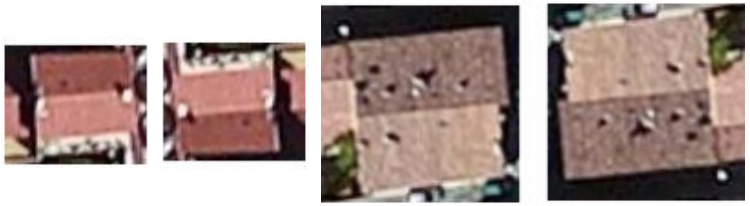


Category 4:
Other



Augmentation

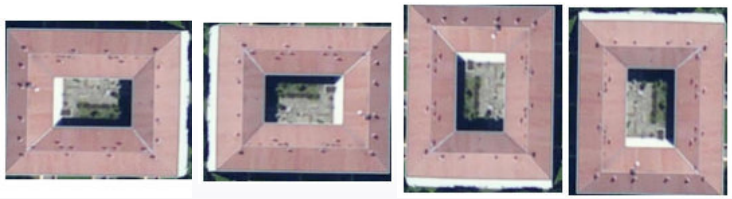
Category 1:
North-South orientation



Category 3:
Flat roof

Category 2:
East-West orientation

Category 4:
Other



Feature extraction from text and images

1. **Texts**
    a. Preprocessing
        i. Lowercase, stemming, lemmarization, stopwords
    b. Bag of words
        i. Huge vectors
        ii. Ngrams can help to use local context
        iii. TFiDF can be of use as postprocessing
    c. Word2vec
        i. Relatively small vectors
        ii. Pretrained models

Feature extraction from text and images

1. **Texts**
    a. Preprocessing
        i. Lowercase, stemming, lemmarization, stopwords
    b. Bag of words
        i. Huge vectors
        ii. Ngrams can help to use local context
        iii. TFiDF can be of use as postprocessing
    c. Word2vec
        i. Relatively small vectors
        ii. Pretrained models
2. **Images**
    a. Features can be extracted from different layers
    b. Careful choosing of pretrained network can help
    c. Finetuning allows to refine pretrained models
    d. Data augmentation can improve the model