Numeric features

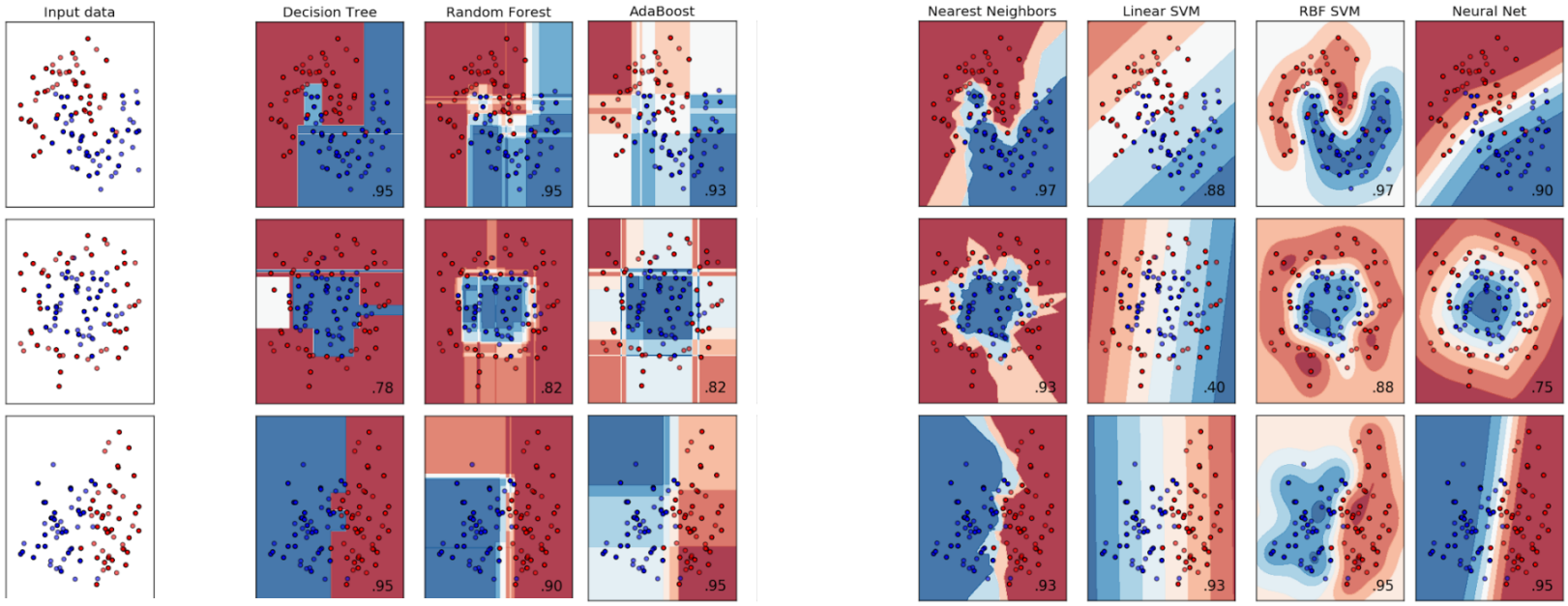
Numeric

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

Preprocessing

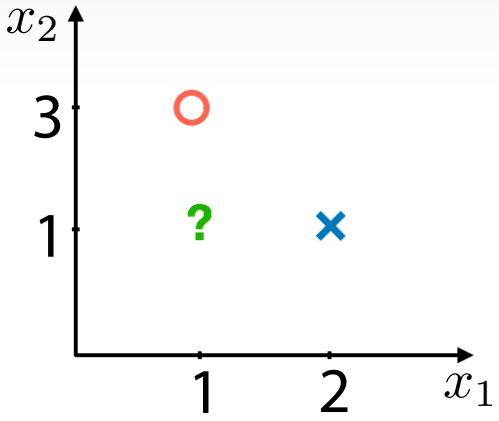
Tree-based models

Non-tree-based models

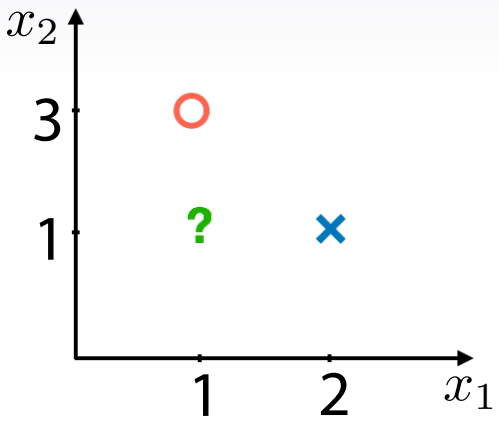


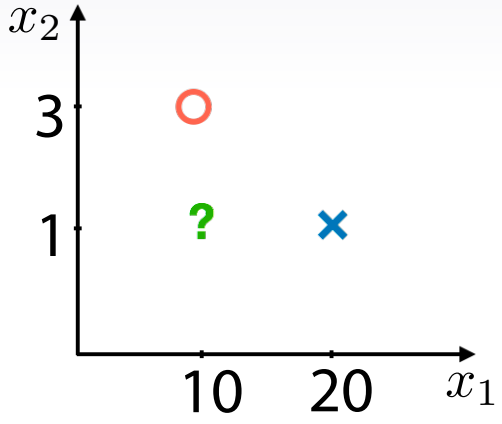
Classifier comparison¶, http://scikit - learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Preprocessing: scaling

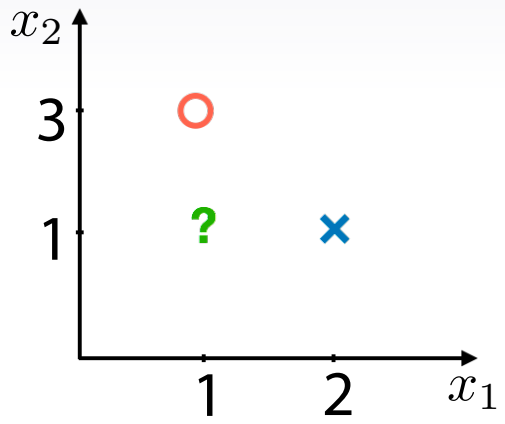


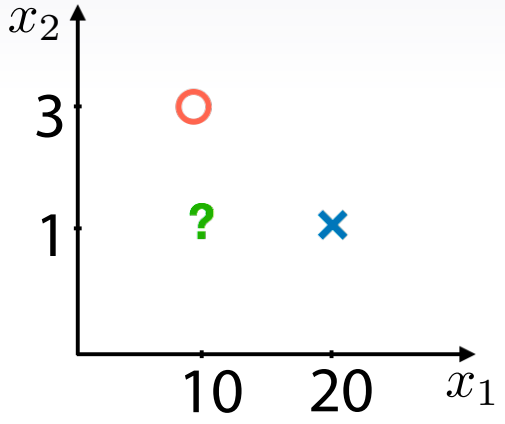
Preprocessing: scaling





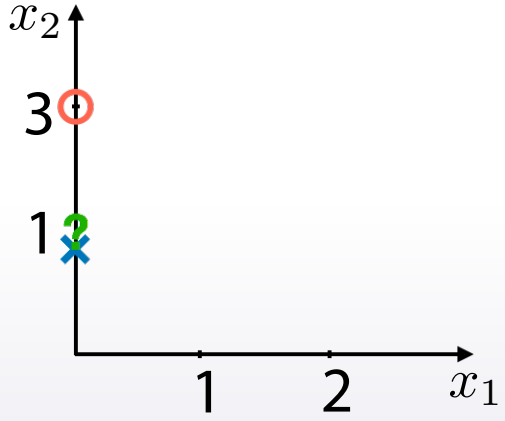
Preprocessing: scaling

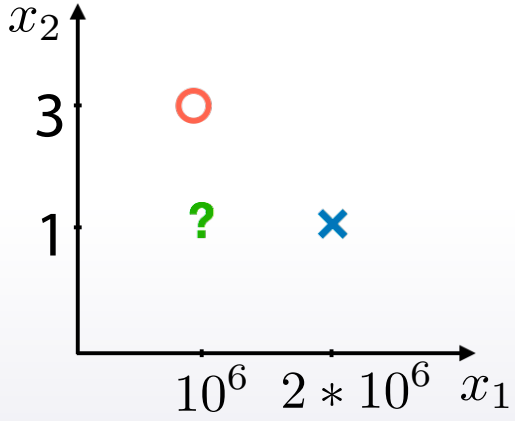




$$x_1 = x_1 * 0$$

$$x_1 = x_1 * 10^6$$





Preprocessing: scaling

1. To [0,1]

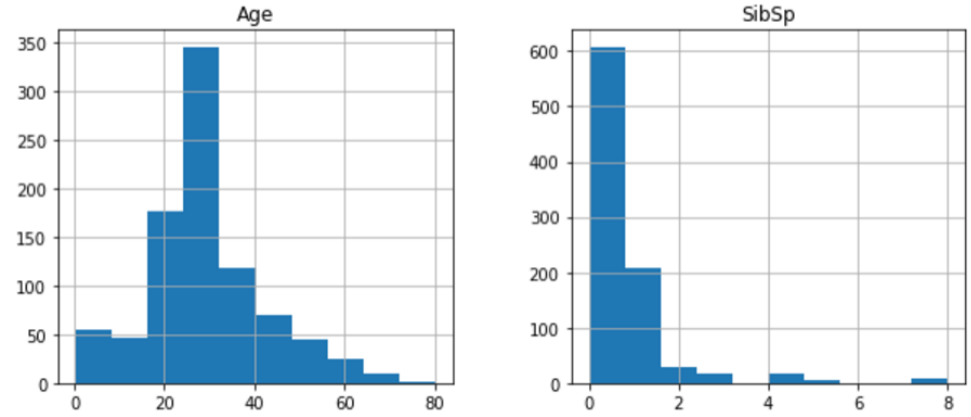
   `sklearn.preprocessing.MinMaxScaler`

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

Preprocessing: scaling

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

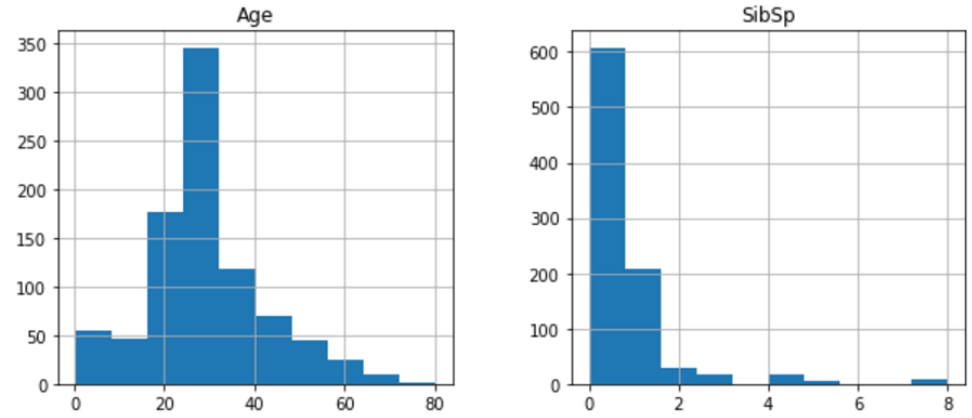
```
train[['Age', 'SibSp']].hist(figsize=(10,4));
```



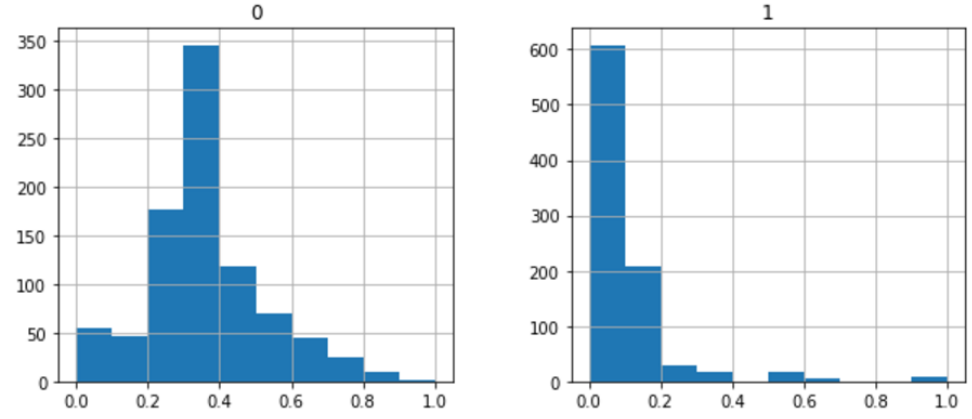
Preprocessing: scaling

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

```
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`

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

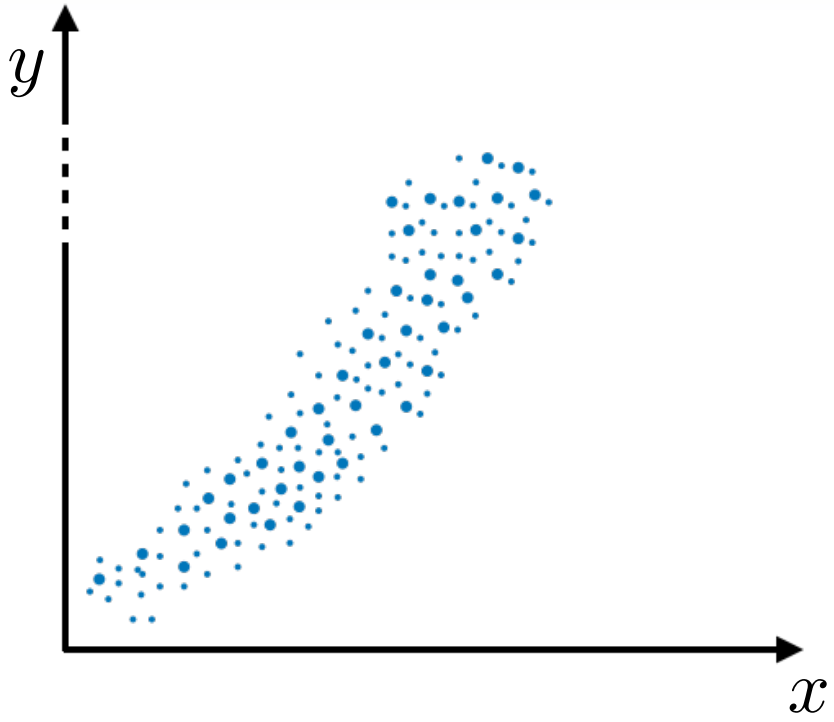
2. To mean=0, std=1

   `sklearn.preprocessing.StandardScaler`

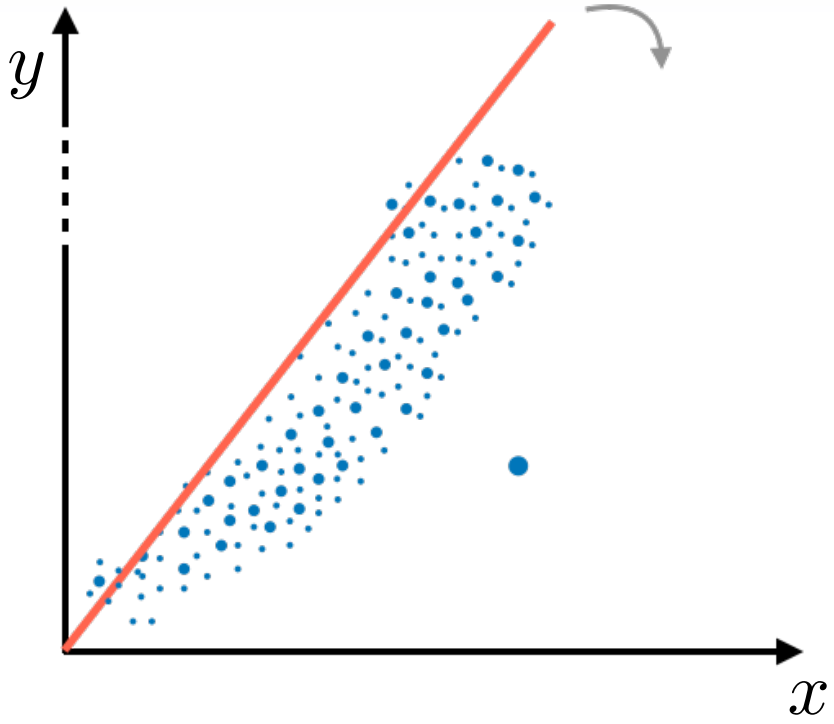
$$X = (X - X.mean())/X.std()$$

Preprocessing: outliers

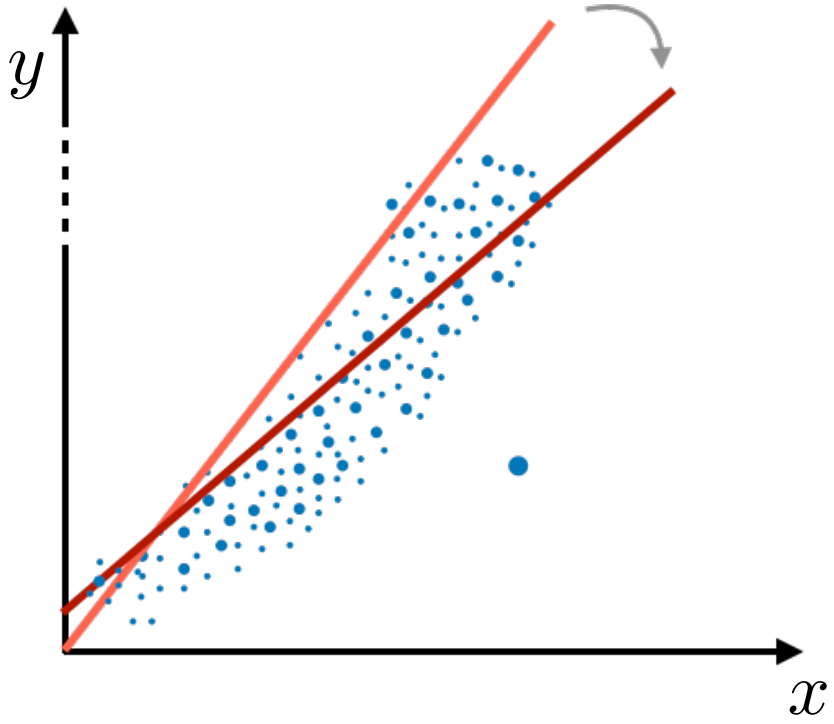
Preprocessing: outliers



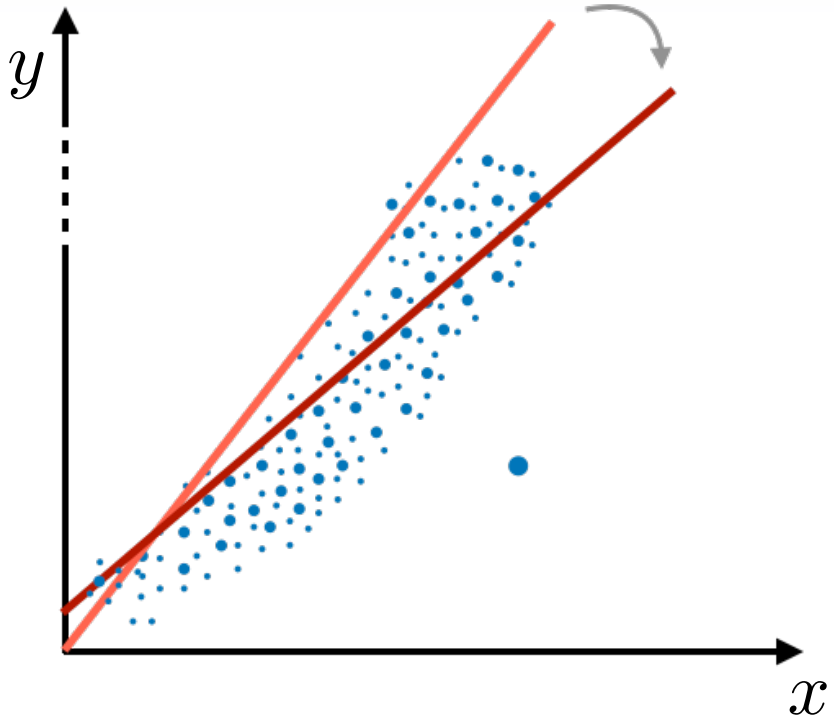
Preprocessing: outliers

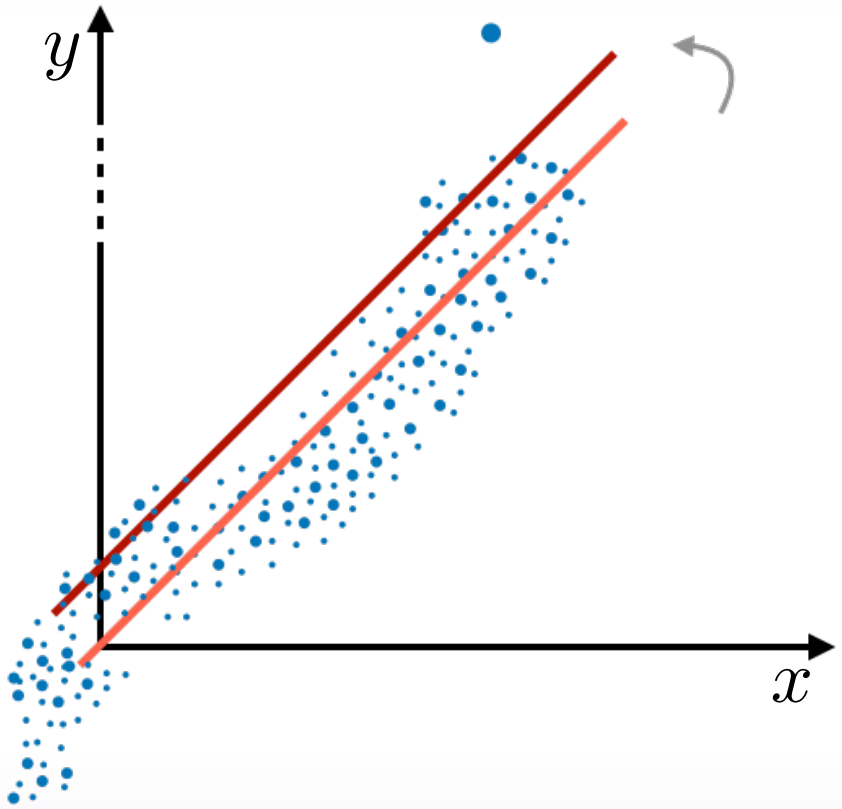


Preprocessing: outliers

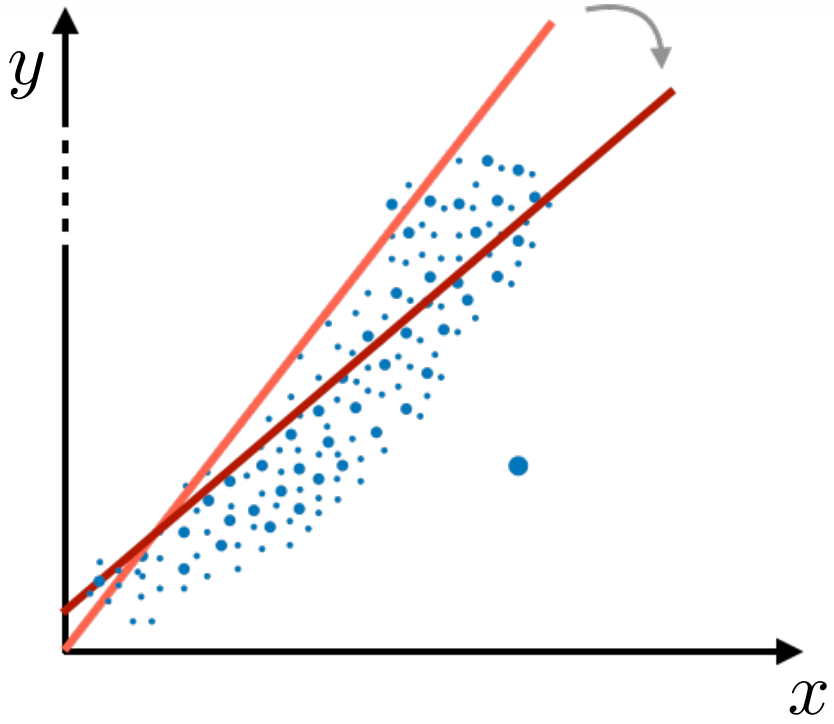


Preprocessing: outliers

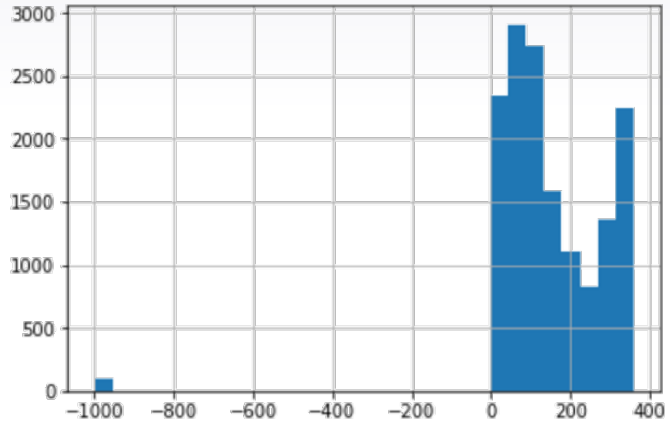




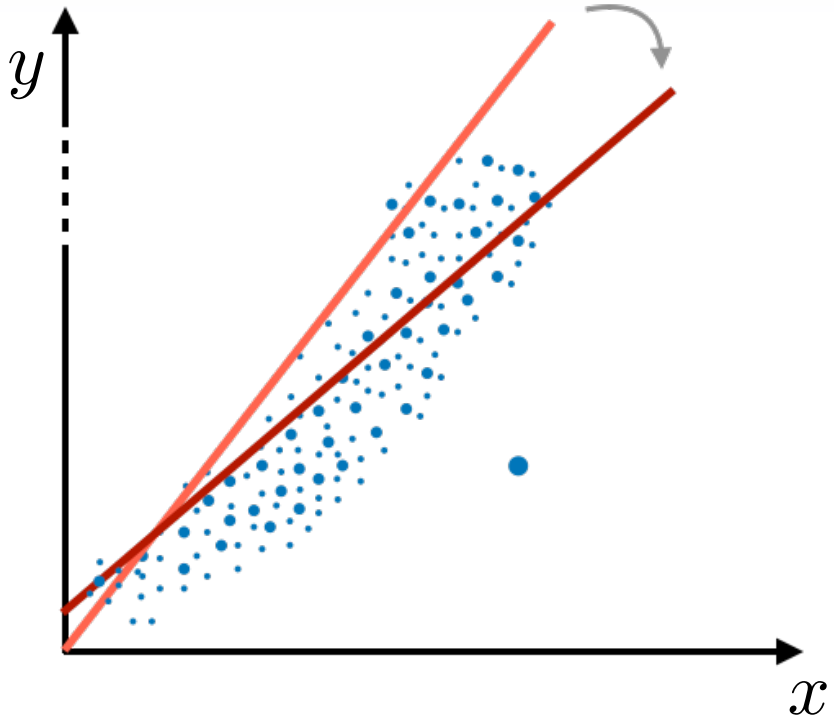
Preprocessing: outliers



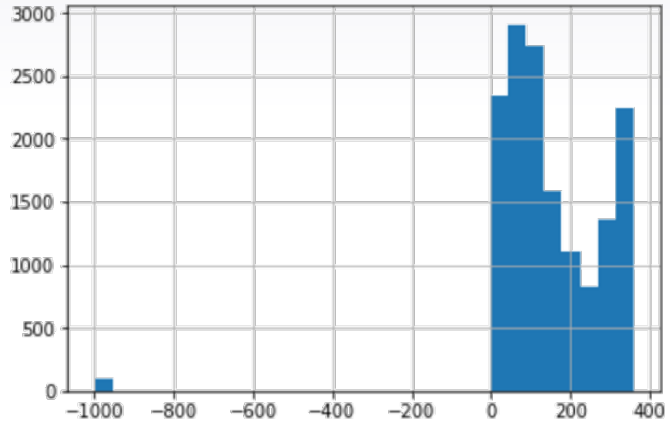

```
In [17]: pd.Series(x).hist(bins=30) ;
```



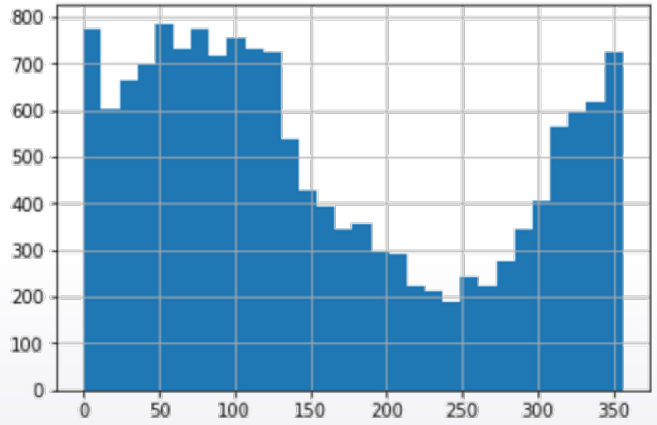
Preprocessing: outliers



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

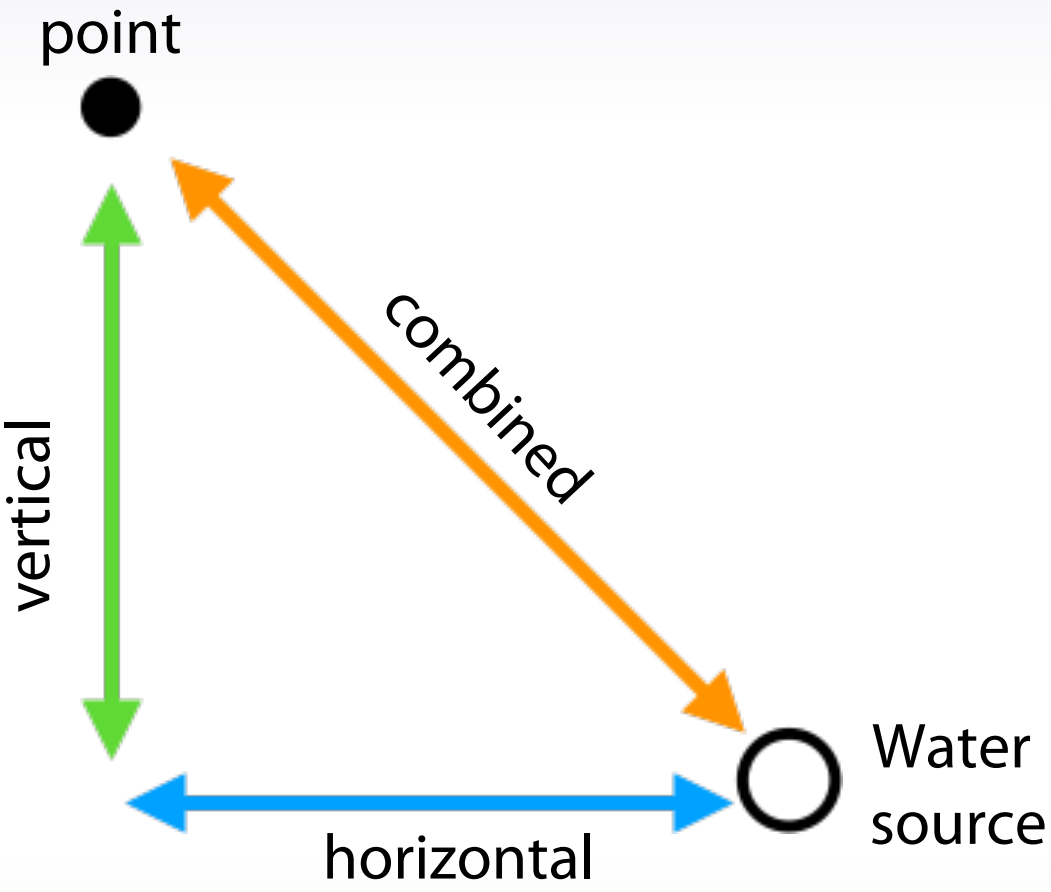
Ways to proceed:

- 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