Lecture 8

Outlier Detection and Predictive Analysis

Assoc. Prof. Dr. Burkay Genç

2024-04-30

Seed used in these slides

```r
set.seed(1024)
```

Libraries used in these slides

```
library(fpc)
library(dplyr)
library(ggplot2)
library(DMwR2)
```

Anomaly Detection

Anomaly Detection

- Has clear ties with clustering
  - Clustering: find and group similar items
  - Anomaly Detection: find items which do not belong to any groups
- Types of outliers
  - Point outliers: a point out of the normal
  - Contextual outliers: a point out of the specific context
    - It is normal to have a heart rate of 80bpm
    - …unless you are dead.
  - Collective outliers: multiple points where only a few is ok
    - Multiple failed login attempts

Univariate Outlier Detection

- the boxplot rule

$$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$$

- Grubb's test

$$z = \frac{|x - \bar{x}|}{s_x}$$

$$\tau = t^2_{\alpha/(2N),N-2}$$

$$z \geq \frac{N-1}{\sqrt{N}} \sqrt{\frac{\tau}{N-2+\tau}}$$

case is an outlier if this inequality holds.

- implemented in package `outliers` as `grubbs.test()`

Univariate Outlier Detection

- For categorical variables there is no simple formula
- We need expert knowledge to compare the distribution of values
  - Then, we can label anomalies

Multi-Variate Outlier Detection

- Types of detection
  - Supervised
  - Unsupervised
  - Semi-supervised

Multi-Variate Outlier Detection

- Unsupervised
  - DBSCAN (we had covered last week)

```
dbscan.outliers <- function(data, ...) {
  require(fpc, quietly=TRUE)
  cl <- dbscan(data, ...)
  posOuts <- which(cl$cluster == 0)
  list(positions = posOuts,
       outliers = data[posOuts,],
       dbscanResults = cl)
  }
```

Unsupervised

- house.data

```
load("house.data")   # loads houseData from file
names(houseData)
```

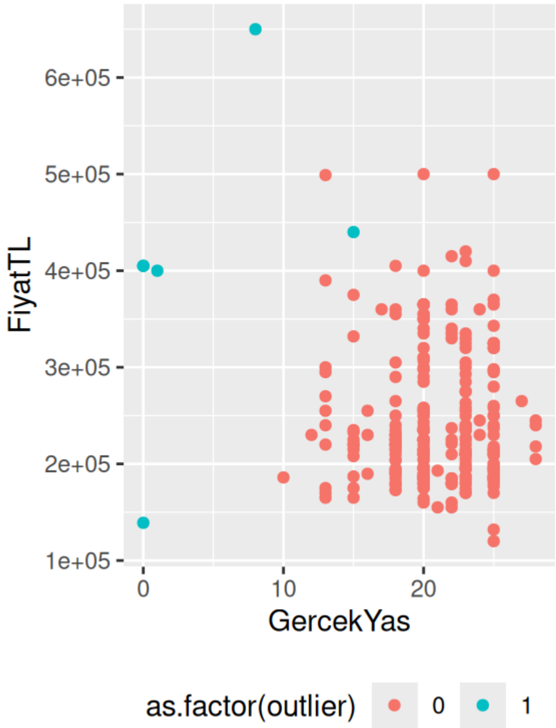
```
## [1] "MustakilMi"   "OrijinalAlan"
"BanyoSayisi"  "OdaSayisi"    "SalonSayisi"
## [6] "ToplamKat"    "GercekYas"    "FiyatTL"
```

```
outs <- dbscan.outliers(houseData,
                        eps = 3,
                        scale=TRUE)
outs$positions
```

```
## [1]  24  65 100 174 190 271
```

```
houseData$outlier = 0
houseData$outlier[outs$positions] = 1
```

```
ggplot(houseData, aes(
  x = GercekYas,
  y = FiyatTL,
  color = as.factor(outlier))) +
  geom_point() +
  theme(legend.position="bottom")
```



Unsupervised

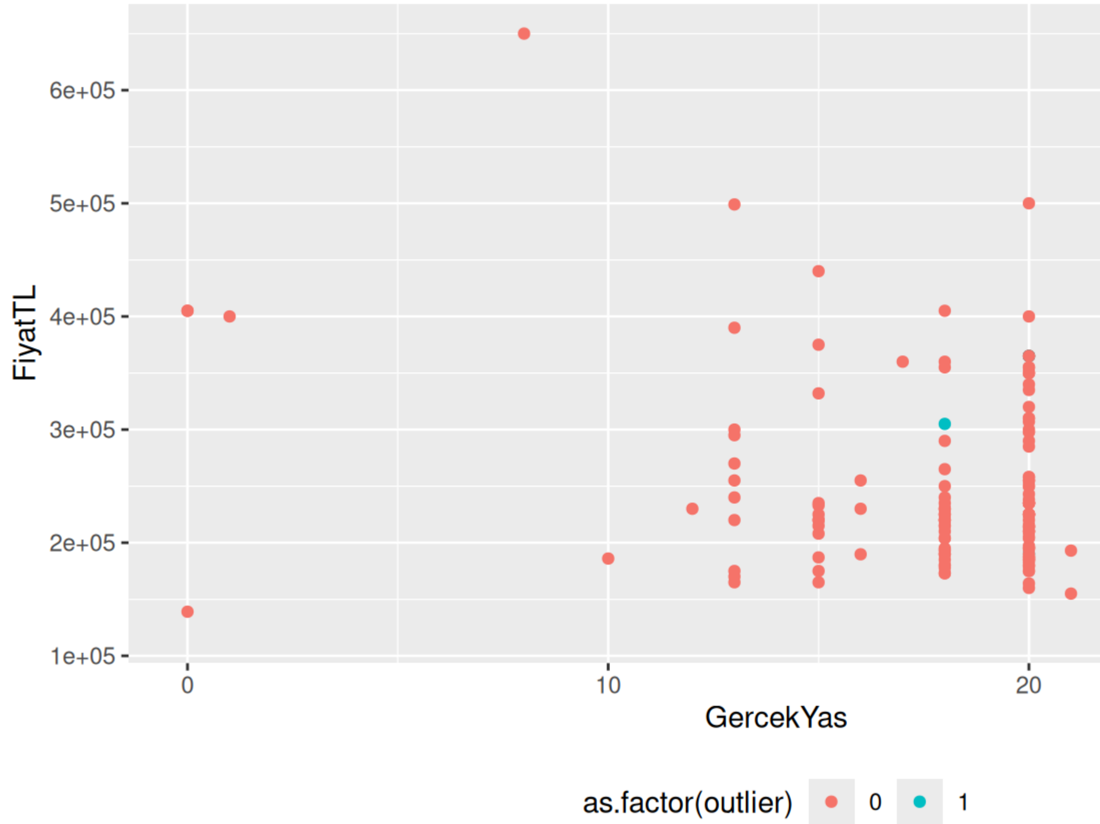
- Another method is OR_h by Torgo, 2007.
  - Uses the merge process of agglomerative hierarchical clustering technique

```
houseData$outlier = NULL
outs <- outliers.ranking(scale(houseData))
outs$rank.outliers[1:10]
```

```
##  [1]   2  46  56 133 180 198 241 251  45 204
```

```
houseData$outlier <- 0
houseData$outlier[
  outs$rank.outliers[1:10]] <- 1
```

```
ggplot(houseData, aes(
  x = GercekYas,
  y = FiyatTL,
  color = as.factor(outlier))) +
  geom_point() +
  theme(legend.position="bottom")
```



Unsupervised

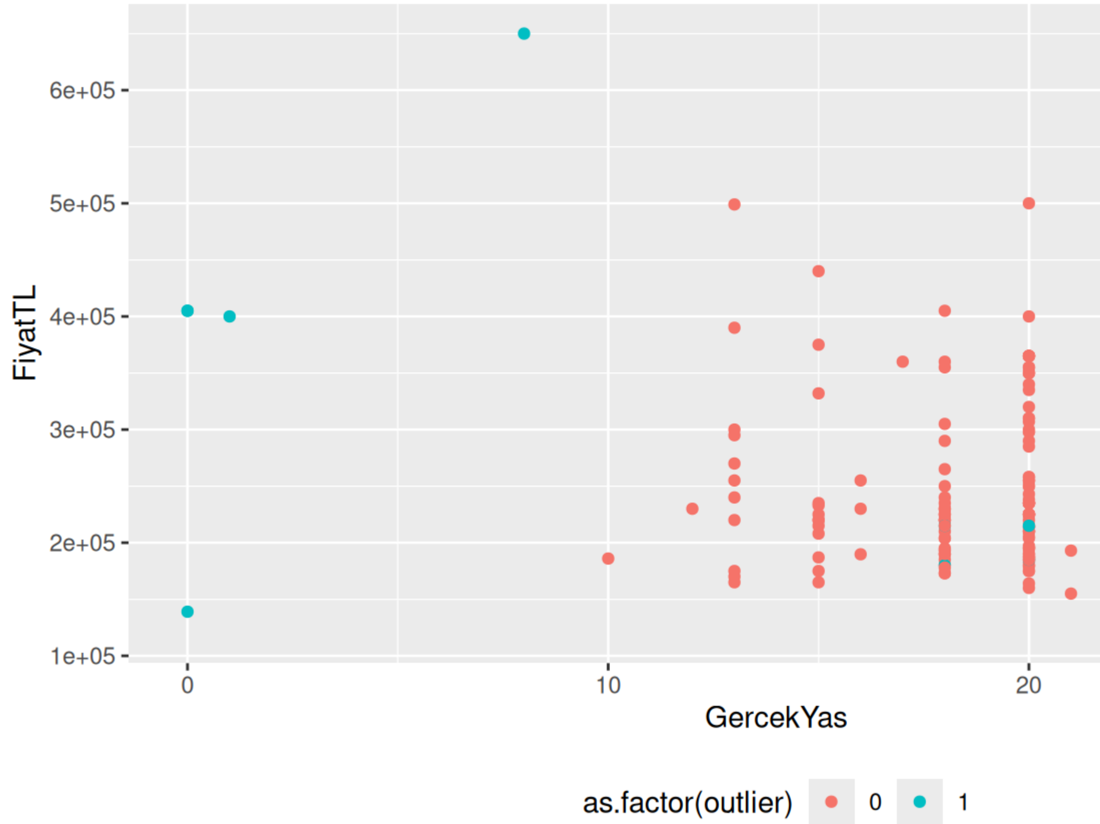
- Another method is LOF by Breunig et al., 2000
- It is implemented as `lofactor` in the book package

```
houseData$outlier = NULL
out.scores <- lofactor(scale(houseData), 15)
top_outliers <- order(out.scores, decreasing =
T)[1:10]
top_outliers
```

```
##  [1] 243  24 100 132 266 174 190  65 248  57
```

```
houseData$outlier <- 0
houseData$outlier[top_outliers] <- 1
```

```
ggplot(houseData, aes(
  x = GercekYas,
  y = FiyatTL,
  color = as.factor(outlier))) +
  geom_point() +
  theme(legend.position="bottom")
```



Supervised

- Training data with manually labeled outliers is required
- Train a classification model with outliers being the target variable
- Use the model for detecting outliers in *new* training data

Major problem : **Imbalance**!

- Outliers are **outliers**, so they will be **out numbered**
- This imbalance creates problems for learning algorithms
  - If outliers are 2% in the set, labeling everything as normal has an accuracy of 98% !
  - Models usually ignore outliers: they are designed to detect regularities, not irregularities

Supervised

- To fix imbalance
  - over sample outliers
  - under sample regulars
  - if supported by the ML method, use biased cost matrices

Predictive Analysis

Predictive Analysis

Using the data at hand, build a model which can be used to predict the value of a response variable based on the values of input variables.

- Almost all ML models are basically curve fitting algorithms
- If you fit a curve to the existing data points, you can use this curve to compute unknown/unobserved points

Predictive Analysis

Mainly two types:

- Classification: nominal target variable
- Regression: numeric target variable

Ordinals may go into one of these categories.

Predictive Analysis

Mostly, predictive analysis is **curve fitting**.

$f(X_1, X_2, \ldots, X_k) \rightarrow Y$

Overall approach:

1. First assume the shape of f (the type of model)
   - linear, logical, probabilistic, complex, ensemble
2. Based on the data, optimize f
3. Evaluate results

Predictive Analysis

Why choose one model over another?

- **Understandability** / Readability
- **Speed** / Complexity
- **Accuracy** / Success of prediction

Evaluation Metrics

Classification

- **Confusion matrix**
  - A matrix displaying frequencies of observations for an interaction of predictions and *ground truth*
  - The predictions are the columns and the actual values are the rows

	c_1	c_2	c_3
c_1	a	b	c
c_2	d	e	f
c_3	g	h	i

- a: the actual value is c_1 and the prediction is c_1
- b: the actual value is c_1 but the prediction is c_2
- d: the actual value is c_2 but the prediction is c_1

Classification

- **Error rate** (aka. the 0/1 loss)

$$L_{0/1} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I(\hat{h}(x_i) \neq y_i)$$

where,

- N_{test} is the number of test cases.
- $I(x)$ is an indicator function:
  - x is false $\rightarrow I(x) = 0$
  - x is true $\rightarrow I(x) = 1$
- $\hat{h}(x_i)$ is the prediction for x_i
- y_i is the actual target value for observation i

Classification

- Accuracy

$$Acc = 1 - L_{0/1}$$

	c_1	c_2	c_3
c_1	a	b	c
c_2	d	e	f
c_3	g	h	i

$$Acc = \frac{a + e + i}{N_{test}}$$

Classification

- **Cost/benefit matrix**

	c_1	c_2	c_3
c_1	$B_{1,1}$	$C_{1,2}$	$C_{1,3}$
c_2	$C_{2,1}$	$B_{2,2}$	$C_{2,3}$
c_3	$C_{3,1}$	$C_{3,2}$	$B_{3,3}$

- Provides **flexible cost and benefit** values for each type of prediction
  - Especially useful in **imbalanced** datasets
  - Also, **fraud detection**, **outlier detection**, etc.

Utility

- Utility is computed as

$$U = \sum_{i=1}^{n_c} \sum_{k=1}^{n_c} CM_{i,k} \times CB_{i,k}$$

- CM: Confusion matrix
- CB: Cost/benefit matrix

Classification

Standard CB matrix:

	outlier	normal
outlier	1	0
normal	0	1

An example CB matrix for outlier detection:

	outlier	normal
outlier	5	-5
normal	-1	0.1

- Consider 98% regular, 2% outlier
  - If we mark everything as normal
    - standard utility : 98
    - modified utility : -10 + 9.8 = -0.2
- You can normalize by maximum utility possible
  - standard utility : 98 / 100 = 0.98
  - modified utility : -0.2 / 19.8 = -0.0101

Classification

- When you have a binary classification

	T	F
T	TP	FN
F	FP	TN

- Precision: rate of correctly identified trues to all predicted as true.

Prec = $\frac{TP}{TP+FP}$

- Recall: rate of correctly identified trues to all actual trues.

Rec = $\frac{TP}{TP+FN}$

Classification

- You can aggregate precision and recall into one metric, the F-measure:

$$F_\beta = \frac{(\beta^2+1)\times Prec\times Rec}{\beta^2\times Prec+Rec}$$

Regression

- For numeric target variables, one frequently used metric is the *mean squared error*:

$$MSE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (\hat{y}_i - y_i)^2$$

- Or for the sake of unit compliance, use *root mean squared error*:

$$RMSE = \sqrt{MSE}$$

- Or, alternatively use *mean absolute error*:

$$MAE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} |\hat{y}_i - y_i|$$

Regression

- You can use a baseline method to produce relative error metrics.
- A baseline method is something naive, such as the mean y value
- *Normalized mean squared error*:

$$NMSE = \frac{\sum_{i=1}^{N_{test}} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N_{test}} (\bar{y}_i - y_i)^2}$$

- We expect NMSE to be close to 0. A value of 1 means a performance as bad as the baseline.
- Also, *Normalized mean absolute error*

$$NMAE = \frac{\sum_{i=1}^{N_{test}} |\hat{y}_i - y_i|}{\sum_{i=1}^{N_{test}} |\bar{y}_i - y_i|}$$

Implementations

- There are many implementations of these metrics
  - function `mmetric` in package `rminer` (Cortez, 2015)
  - functions `classificationMetrics` and `regressionMetrics` in package `performanceEstimation` (Torgo, 2014a)
  - function `performance` in package `ROCR` (Sing et al., 2009)
  - function `performance` in package `mlr` (Bischl et al., 2016)
- And, you can always compute them on the fly.

~~In-class~~ At-home Activity

- Load house.data into R
- Apply clustering to the data
  - How many clusters seems to be the optimal?
- Apply anomaly detection to the data
  - Do you catch a few or many anomalies?