Department of Computer Science

IT3212 - Data-Driven Software

Assignment 1

Authors:
Birk Strand Bjørnaa
Carl Edward Storlien
Christian Stensøe
Åsmund Løvoll

01.10.2024

Table of Contents

List of Figures

List of Tables

1 Data Exploration

a)

Details of dataset

Using dataset `smoking_drinking_dataset.csv`. Details of the dataset:

First few rows

The first five rows are of the dataset[1] shown below in tabular form over three lines.

sex	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_right	SBP	DBP
Male	35	170	75	90.0	1.0	1.0	1.0	1.0	120.0	80.0
Male	30	180	80	89.0	0.9	1.2	1.0	1.0	130.0	82.0
Male	40	165	75	91.0	1.2	1.5	1.0	1.0	120.0	70.0
Male	50	175	80	91.0	1.5	1.2	1.0	1.0	145.0	87.0
Male	50	165	60	80.0	1.0	1.2	1.0	1.0	138.0	82.0

BLDS	tot_chole	HDL_chole	LDL_chole	triglyceride	hemoglobin	urine_protein	serum_creatinine
99.0	193.0	48.0	126.0	92.0	17.1	1.0	1.0
106.0	228.0	55.0	148.0	121.0	15.8	1.0	0.9
98.0	136.0	41.0	74.0	104.0	15.8	1.0	0.9
95.0	201.0	76.0	104.0	106.0	17.6	1.0	1.1
101.0	199.0	61.0	117.0	104.0	13.8	1.0	0.8

SGOT_AST	SGOT_ALT	gamma_GTP	SMK_stat_type_cd	DRK_YN
21.0	35.0	40.0	1.0	Y
20.0	36.0	27.0	3.0	N
47.0	32.0	68.0	1.0	N
29.0	34.0	18.0	1.0	N
19.0	12.0	25.0	1.0	N

Table 1: First five rows of dataset

[1] Soo.Y, Smoking and Drinking Dataset with body signal, Kaggle

Data types and unique values

Table 2 shows the data types of the columns.

The unique values for the categorical columns, `sex`, `hear_left`, `hear_right`, `urine_protein`, `SMK_stat_type_cd` and `DRK_YN` are shown in Table 1. This was confirmed using `Pandas'` built-in function `unique()` on the dataset. For clarification, the unique values of `urine_protein` are `1, 2, 3, 4, 5, 6`.

Column	Data Type
sex	object
age	int64
height	int64
weight	int64
waistline	float64
sight_left	float64
sight_right	float64
hear_left	float64
hear_right	float64
SBP	float64
DBP	float64
BLDS	float64
tot_chole	float64
HDL_chole	float64
LDL_chole	float64
triglyceride	float64
hemoglobin	float64
urine_protein	float64
serum_creatinine	float64
SGOT_AST	float64
SGOT_ALT	float64
gamma_GTP	float64
SMK_stat_type_cd	float64
DRK_YN	object

Table 2: Data types of columns

b)

Missing values and summary statistics

The `smoking_drinking_dataset.csv` dataset has no missing values. The Pandas method `isna()` was used to inspect this.

There are `991346` rows in the dataset. Summary statistics of the columns are shown in Table 3:

	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_right	SBP	DBP
mean	47.61	162.24	63.28	81.23	0.98	0.98	1.03	1.03	122.43	76.05
std	14.18	9.28	12.51	11.85	0.61	0.60	0.17	0.17	14.54	9.89
min	20.00	130.00	25.00	8.00	0.10	0.10	1.00	1.00	67.00	32.00
25%	35.00	155.00	55.00	74.10	0.70	0.70	1.00	1.00	112.00	70.00
75%	60.00	170.00	70.00	87.80	1.20	1.20	1.00	1.00	131.00	82.00
max	85.00	190.00	140.00	999.00	9.90	9.90	2.00	2.00	273.00	185.00

	BLDS	tot_chole	HDL_chole	LDL_chole	triglyceride	hemoglobin	urine_protein
mean	100.42	195.56	56.94	113.04	132.14	14.23	1.09
std	24.18	38.66	17.24	35.84	102.20	1.58	0.44
min	25.00	30.00	1.00	1.00	1.00	1.00	1.00
25%	88.00	169.00	46.00	89.00	73.00	13.20	1.00
75%	105.00	219.00	66.00	135.00	159.00	15.40	1.00
max	852.00	2344.00	8110.00	5119.00	9490.00	25.00	6.00

	serum_creatinine	SGOT_AST	SGOT_ALT	gamma_GTP	SMK_stat_type_cd
mean	0.86	25.99	25.76	37.14	1.61
std	0.48	23.49	26.31	50.42	0.82
min	0.10	1.00	1.00	1.00	1.00
25%	0.70	19.00	15.00	16.00	1.00
75%	1.00	28.00	29.00	39.00	2.00
max	98.00	9999.00	7210.00	999.00	3.00

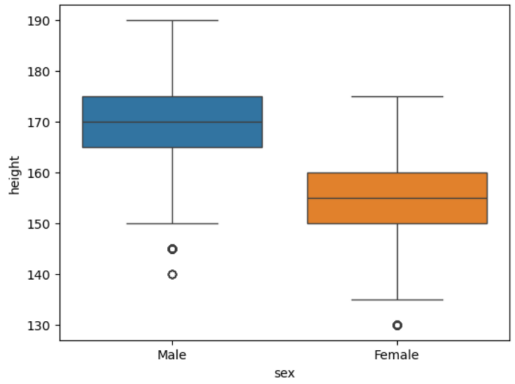
Table 3: Summary statistics of dataset

Outliers

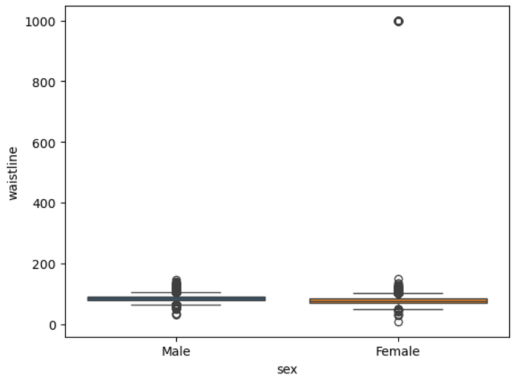
The univariate outliers in the dataset `smoking_drinking_dataset.csv` can be identified by using a box plot for the features. For this task, the `seaborn` library in Python was used to generate the box plots.

Univariate outliers have been identified in almost all of the features in this dataset: `height`, `weight`, `waistline`, `sight_left`, `sight_right`, `SBP`, `DBP`, `BLDS`, `tot_chole`, `HDL_chole`, `LDL_chole`, `triglyceride`, `hemoglobin`, `serum_creatinine`, `SGOT_AST`, `SGOT_ALT` and `gamma_GTP`.

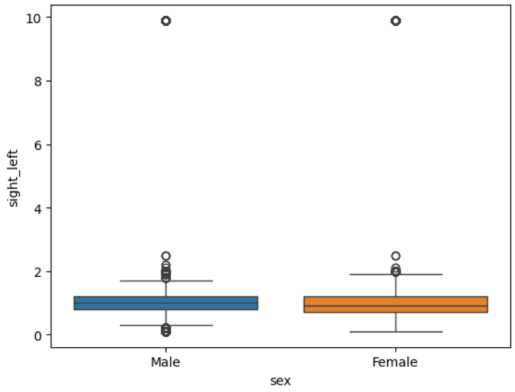
Figure 1 shows some of the box plots indicating univariate outliers.



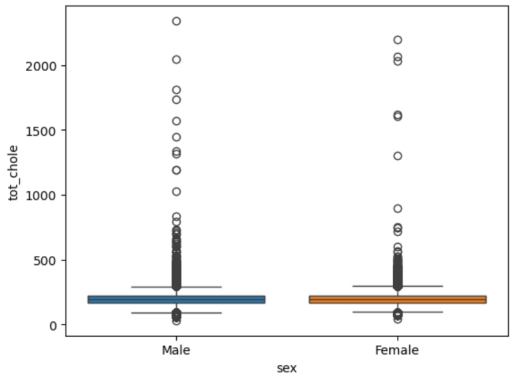
(a) Box plot of `height`



(b) Box plot of `waistline`



(c) Box plot of `sight_left`



(d) Box plot of `tot_chole`

Figure 1: Examples of univariate outliers identified with box plots.

2 Data Cleaning

Handling missing values
There are generally two ways of dealing with missing data in a dataset: If the dataset is large, the missing values can be ignored. Alternatively, one can fill in the missing values.

In the dataset `food-bank`, and the csv-file `live1.csv`, we chose to remove all the null values. This is because we found that only 1.88% of the elements contained null values, and we we considered this percentage small enough that the most logical solution was deletion. The deletion was performed by creating a dataframe with `pandas`-function `read`, and then using the `numpy` library's `dropna`-function.

Upon inspection of the dataset, we learned that the missing values were contained to certain areas of the dataset. An example is Ducks in Albania. As the only data available was from 1994 to 2011, we decided that this data range was insufficient to assess trends in duck stock for the missing years (1961-1993 and 2011-2020). Furthermore, the data were very similar when looking past the stock and year, which means that the consequence of removing the rows with missing data did not lead to losing other important data. If the dataset had been more complex, such that deletion of rows had lead to loss of other relevant data, we would be more inclined to use another method for handling missing values.

We initially considered handling the missing values with *interpolation*, which is used to estimate the values of unknown data points between existing ones. As we learned upon inspection that the missing values were contained to certain areas, and were several in a row, it was deemed unfit for this dataset.
Numerical imputation was also considered as an approach, but as we wouldn't be able to take changing trends over time into account, we deemed this approach unfit as well. There was no clear relationship between the datapoints, and thus we were not confident that either mean or median interpolation, or other regression models would yield correct results.

	Area	Item	Element	Year	Unit	Value
823	Albania	Ducks	Stocks	2004	1000 Head	527.0
824	Albania	Ducks	Stocks	2005	1000 Head	528.0
825	Albania	Ducks	Stocks	2006	1000 Head	500.0
826	Albania	Ducks	Stocks	2007	1000 Head	800.0
827	Albania	Ducks	Stocks	2008	1000 Head	1000.0
828	Albania	Ducks	Stocks	2009	1000 Head	1100.0
829	Albania	Ducks	Stocks	2010	1000 Head	1100.0
830	Albania	Ducks	Stocks	2011	1000 Head	1000.0
831	Albania	Ducks	Stocks	2012	1000 Head	NaN
832	Albania	Ducks	Stocks	2013	1000 Head	NaN
833	Albania	Ducks	Stocks	2014	1000 Head	NaN
834	Albania	Ducks	Stocks	2015	1000 Head	NaN
835	Albania	Ducks	Stocks	2016	1000 Head	NaN
836	Albania	Ducks	Stocks	2017	1000 Head	NaN
837	Albania	Ducks	Stocks	2018	1000 Head	NaN
838	Albania	Ducks	Stocks	2019	1000 Head	NaN
839	Albania	Ducks	Stocks	2020	1000 Head	NaN
840	Albania	Geese and guinea fowls	Stocks	1961	1000 Head	NaN
841	Albania	Geese and guinea fowls	Stocks	1962	1000 Head	NaN
842	Albania	Geese and guinea fowls	Stocks	1963	1000 Head	NaN

(a) Part of dataset before deletion.

	Area	Item	Element	Year	Unit	Value
823	Albania	Ducks	Stocks	2004	1000 Head	527.0
824	Albania	Ducks	Stocks	2005	1000 Head	528.0
825	Albania	Ducks	Stocks	2006	1000 Head	500.0
826	Albania	Ducks	Stocks	2007	1000 Head	800.0
827	Albania	Ducks	Stocks	2008	1000 Head	1000.0
828	Albania	Ducks	Stocks	2009	1000 Head	1100.0
829	Albania	Ducks	Stocks	2010	1000 Head	1100.0
830	Albania	Ducks	Stocks	2011	1000 Head	1000.0
873	Albania	Geese and guinea fowls	Stocks	1994	1000 Head	240.0
874	Albania	Geese and guinea fowls	Stocks	1995	1000 Head	240.0
875	Albania	Geese and guinea fowls	Stocks	1996	1000 Head	244.0
876	Albania	Geese and guinea fowls	Stocks	1997	1000 Head	225.0
877	Albania	Geese and guinea fowls	Stocks	1998	1000 Head	205.0
878	Albania	Geese and guinea fowls	Stocks	1999	1000 Head	200.0
879	Albania	Geese and guinea fowls	Stocks	2000	1000 Head	250.0
880	Albania	Geese and guinea fowls	Stocks	2001	1000 Head	280.0
881	Albania	Geese and guinea fowls	Stocks	2002	1000 Head	275.0
882	Albania	Geese and guinea fowls	Stocks	2003	1000 Head	270.0
883	Albania	Geese and guinea fowls	Stocks	2004	1000 Head	352.0
884	Albania	Geese and guinea fowls	Stocks	2005	1000 Head	353.0

(b) Part of dataset after deletion.

Figure 2: Comparison of dataset before and after deletion.

3 Handling Outliers

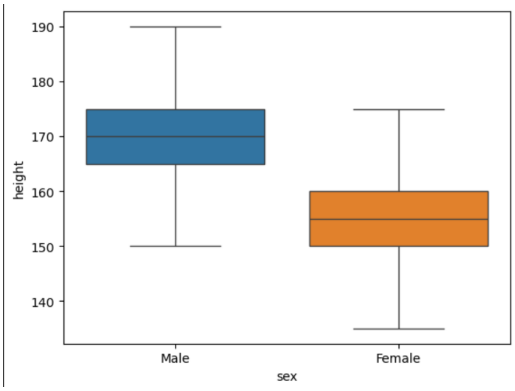
To detect and handle outliers in the dataset `smoking_drinking_dataset.csv`, the IQR (Interquartile Range) method is used in this task. The IQR method marks data points outside the desired range of inliers, or "normal" data points. Usually, a data point is marked as outlier if it lies more than 1.5 times the IQR below the first quartile, or more than 1.5 times the IQR above the third quartile. IQR is the range of values between the first and third quartile.

When first applying the IQR method, we noticed that males and females have different range of values in some of the features. Therefore, we modified the IQR method to calculate Q1 and Q3 for males and females separately.

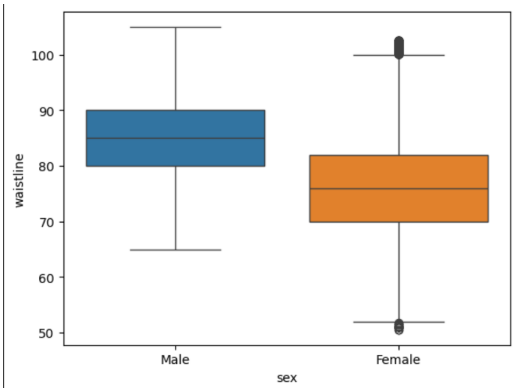
The IQR method marked **36%** of the entire dataset as outliers. Because this portion constitutes too large a part of the dataset, the outliers should not be removed. It would reduce the size of the dataset too much which can impact how well our final model is fit.

We considered two methods of handling the outliers that didn't involve removal. One method is capping the outlier values to the lower or upper bound of the IQR method (setting their value as close to the inlier-outlier border as possible). Another method is setting the outlier values to the mean values of the feature.

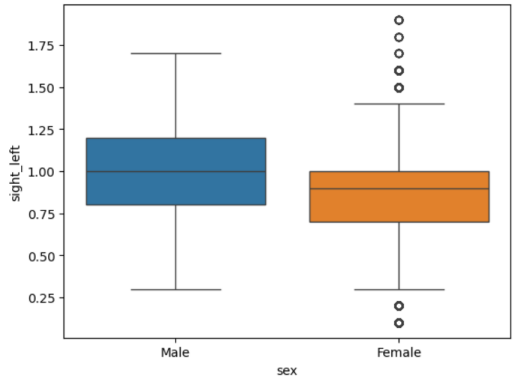
Since we do not know enough about this domain (body signals), we chose to set the outlier values to the mean of the feature, with respect to the sex. After exploring the dataset, it looked like a large portion of the outliers were measurement errors instead of naturally occurring extremes. It therefore makes sense to set the values to the mean instead of capping since it's not necessarily an actual extreme value.



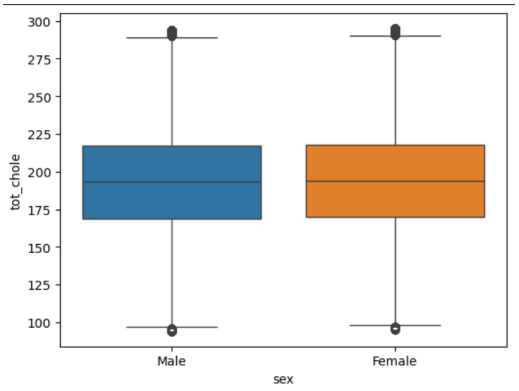
(a) Box plot of `height` after handling outliers



(b) Box plot of `waistline` after handling outliers



(c) Box plot of `sight_left` after handling outliers



(d) Box plot of `tot_chole` after handling outliers

Figure 3: Examples of features after handling outliers

Figure 3 shows the same box plots as in Figure 1 after handling the outliers. Table 5 shows the same summary statistics as Table 4 after handling the outliers.

	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_right	SBP	DBP
mean	47.61	162.25	62.60	80.96	0.95	0.94	1.03	1.03	121.87	75.71
std	14.18	9.28	11.25	9.15	0.32	0.28	0.17	0.17	13.34	9.27
min	20.00	135.00	35.00	50.50	0.10	0.30	1.00	1.00	80.00	49.00
25%	35.00	155.00	55.00	74.50	0.70	0.80	1.00	1.00	112.00	70.00
75%	60.00	170.00	70.00	87.00	1.20	1.20	1.00	1.00	130.00	81.00
max	85.00	190.00	90.00	105.00	1.90	1.70	2.00	2.00	160.00	105.00

	BLDS	tot_chole	HDL_chole	LDL_chole	triglyceride	hemoglobin	urine_protein
mean	96.23	194.33	56.16	112.05	117.15	14.31	1.09
std	11.46	35.73	13.63	32.59	57.98	1.43	0.44
min	63.00	94.00	18.00	20.00	1.00	10.60	1.00
25%	89.00	169.00	46.00	89.00	73.00	13.20	1.00
75%	103.11	218.00	65.00	134.00	152.00	15.40	1.00
max	135.00	295.00	101.00	204.00	332.00	18.10	6.00

	serum_creatinine	SGOT_AST	SGOT_ALT	gamma_GTP	SMK_stat_type_cd
mean	0.85	23.57	22.11	28.30	1.61
std	0.19	6.56	10.22	18.09	0.82
min	0.30	4.00	1.00	1.00	1.00
25%	0.70	19.00	15.00	16.00	1.00
75%	1.00	28.00	28.00	35.00	2.00
max	1.50	47.00	60.00	103.00	3.00

Table 4: Summary statistics of dataset after handling outliers

From Table 5, one can quickly observe that all the extreme max values are gone, compared to the original summary in Table 4.

We also tried the `IsolationForest` class from `sklearn.ensemble` on the dataset. We set the hyperparameter `contamination` to 0.05 to identify **5%** of the dataset as anomalies. The machine learning model identifies both univariate and multivariate outliers. After using the methods `decision_function()` and `predict()`, we confirmed the model had identified the correct proportion of the dataset. However, setting the ideal hyperparameters becomes the hardest part. We chose not to explore it any further.

4 Data Transformation

a) In task 4 we have used the `live1.csv` dataset. Label encoding is commonly used when dealing with categorical variables in datasets that need to be processed by machine learning algorithms. Most machine learning models work with numerical input data, so categorical text data must be converted into numbers. Label encoding transforms each unique category in a column into an integer value, enabling models to process the information.

For this dataset we did not consider one-hot encoding as it is less efficient for this particular dataset. One-hot encoding creates additional binary columns for each unique category, which can increase the dimensionality of the dataset, especially when the categorical columns like Area or Item contain many unique values. This leads to columns and rows where most values are zero, consuming more memory. We therefore concluded that label encoding was the correct way to transform the data.

For this particular dataset we removed the feature named **Element** which had the same value in the entire dataset, a so-called dimensionality reduction. **Unit** was also removed after we scaled the **Values** feature with a factor of 1000 due to the **Unit** feature implying what factor the **Value** was multiplied with. This made the **Values** feature more scalable. As shown in Tables 6 and 7, we used the LabelEncoder() method from `sklearn.preprocessing` library to label the features. The method labels alphabetically, as the tables show.

Encoded Value	Area
0	Afghanistan
1	Africa
2	Albania
3	Algeria
4	Americas
...	...
241	Zambia
242	Zimbabwe

(a) Mapping of Area encoded values

Encoded Value	Item
0	Asses
1	Beehives
2	Buffaloes
3	Camels
4	Cattle
...	...
13	Sheep

(b) Mapping of Item encoded values

Area	Item	Year	Value
0.0	0.0	0.000000	0.000039
0.0	0.0	0.016949	0.000026
0.0	0.0	0.033898	0.0000305
0.0	0.0	0.050847	0.000035
0.0	0.0	0.067797	0.000039

(c) First five rows of scaled data

Figure 4: Mapping of Area and Item encoded values and scaled data

b) For our task, we used the normalizing technique which includes the use of MinMaxScaler(). We used the `.fittransform` method for scaling the features in the dataset.

Feature scaling is used in machine learning because many algorithms are sensitive to the range of features. Models that rely on distance metrics can be skewed by features with larger values, causing unequal contributions. Also, gradient-based methods converge faster and more reliably when features are on similar scales. Unscaled data can lead to inefficient learning and poor performance. Scaling also helps to fairly penalize features and improve model generalization. Summarized, feature scaling standardizes features, leading to faster, more accurate models.

5 Data Splitting

a) The method we used to split the data was firstly to randomize the data such that the data split would be fair, and unbiased. Secondly, we used the length of the dataset and a factor of 0.8 so split the set in a 80-20 train-test split. By just switching up the factor, we easily could split the set in a 70-30 split or other requested splits.

b) Splitting the data into training, validation, and test sets is important for building reliable machine learning models. The model is trained on one subset of data and evaluated on another subset, which helps prevent overfitting—where the model learns patterns specific to the training data but fails to generalize to new data. The test set acts as a final evaluation to check how well the model performs on completely unseen data, simulating real-world use. This separation ensures that performance metrics provide a more accurate reflection of how the model will behave in practice, preventing misleadingly high results due to memorization of the training data.

6 Bonus - Applying PCA to the dataset

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional space by identifying the principal components that capture the most variance. These components highlight the most impactful features, reducing noise and redundancy [2].

We applied PCA to the `smoking_drinking_dataset.csv` dataset, ensuring the features were scaled using `StandardScaler` to normalize values to a mean of 0 and standard deviation of 1 [3]. When applying PCA, accuracy may drop, though our dataset has only 23 features, suggesting all are somewhat important.

For demonstration, we used `LogisticRegression` to predict whether someone is a drinker, training models on both the full and reduced dimensions. We set PCA to retain 95% of features, finding little difference in predictions with 18 components. This suggests some features, like eyesight and hearing, have minimal impact and could be dropped. Waistline and weight, due to correlation, may also be redundant.

Reducing to 2 components dropped accuracy from 0.72 to 0.69 but made the model much lighter and easier to interpret.

[2]Warya, Aish (n.d.). Principal Component Analysis(PCA)
[3]Warya, Aish (n.d.). Principal Component Analysis(PCA)

References

Codest, Session (n.d.). *Anomaly detection with isolation forest in scikit-learn*. URL: https://medium.com/mlthinkbox/anomaly-detection-with-isolation-forest-in-scikit-learn-99417dcc3971.

Dey, Akash (n.d.). *How to handle outliers*. URL: https://www.kaggle.com/code/aimack/how-to-handle-outliers.

Soo.Y (n.d.). *Smoking and Drinking Dataset with body signal*. URL: https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset.

Warya, Aish (n.d.). *Principal Component Analysis(PCA)*. URL: https://www.geeksforgeeks.org/principal-component-analysis-pca/.