MSAI Statistics Home Assignment 8
soft deadline: 30/05/2024 19:00 GMT+3
hard deadline: 30/05/2024 19:00 GMT+3

As announced earlier, grading for HWs consists of points and bonus points. Solving bonus (indicated with a star) problems is not required, but recommended. Solving all homeworks' normal problems correctly will give you a score of 7, solving all homeworks' bonus problems correctly will give you additional 2 points to the score.

Hand-written solutions are accepted if the handwriting is clear enough and scanned with sufficient quality, but LaTeX is always preferable. This homework includes a python task, which can be solved in Google Colab or in a local Jupyter Notebook. It is thus handy to solve everything (both LaTeX and code) in a single Jupyter Notebook.

Solutions obtained with the use of ChatGPT and similar models can be accepted if the solution is clearly indicated as such, and model version and prompt is provided. If the solution is found to be from ChatGPT and similar models without indication and model/prompt detail, the teachers will evaluate the problem as zero points.

Problem 1. (2 points) Remember Spearman's rank correlation coefficient, which can be written as:

$$\rho_S = \frac{12}{n^3 - n} \sum_{i=1}^{n} \left(i - \frac{n+1}{2} \right) \left(T_i - \frac{n+1}{2} \right)$$

where (R_i, S_i) are the original ranks and (i, T_i) are the ranks sorted by first component.

Prove that we can rewrite this as:

$$\rho_S = 1 - \frac{6}{n^3 - n} \sum_{i=1}^{n} (i - T_i)^2 = 1 - \frac{6}{n(n-1)(n+1)} \sum_{i=1}^{n} (R_i - S_i)^2$$

Hint: expand the series

$$\sum \left[\left(i - \frac{n+1}{2} \right) - \left(T_i - \frac{n+1}{2} \right) \right]^2$$

Problem 2. (4 points) Computer experiment. Use the following code to load the data and get acquainted with it.

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
data = load_diabetes(as_frame=True)
print(data["DESCR"])
df = data["frame"]
df_train, df_test = train_test_split(df, test_size=0.2)
```

1. (2 point) Test independence hypothesis for every feature with target (10 tests total). Remember the normality assumption! Don't forget to account for multiple testing! Fit a linear regression model using features for which we reject the independence hypothesis. Measure RMSE on testing dataset.

2. (2 point) Train a regularized regression model with all features considered. Remember the normality assumption! Read the summary of your fit. Find the confidence intervals for every coefficient. Fit a new ordinary linear regression model excluding all features that have zero in the confidence interval. Measure RMSE on testing dataset.

Problem 3. (4 bonus points) Computer experiment. Use the following code to load the data and get acquainted with it.

```
from sklearn.datasets import load_wine
data = load_wine(as_frame=True)
df = data["frame"]
X = df["color_intensity"]
Y = df["hue"]
Z = df["flavanoids"]
```

We can test for independence between X and Y using Pearson's correlation coefficient:

$$\rho = \frac{\frac{1}{N}\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\mathbb{V}(X)\mathbb{V}(Y)}}$$

The test is

$$H_0: \quad X, Y \text{ - independent} \equiv p(x, y) = p(x)p(y)$$

If X, Y are normally distributed, under the null-hypothesis of independence, the t-statistic,

$$T = \frac{\rho\sqrt{n-2}}{\sqrt{1-\rho^2}} \sim T_{n-2}$$

has Student's t-distribution with $n-2$ degrees of freedom. So you can use the CDF of this distribution to find the p-value.

But, of course, all these tests are already implemented in Python – you can use scipy.stats.pearsonr to find both the value of ρ and the p-value of this test.

1. (1 point) Find $\rho(X, Y)$ and p-value with SciPy's built-in test
2. (1 point) Find $\rho(X, Y)$ with formula and p-value from the CDF of the t-distribution

This test, however, only checks linear dependence. If there is some non-linear dependence, line we have between X and Z, it would not distinguish it from independence. There exists, for example, Spearman's correlation coefficient (and an independence test for it) – but it works well only for monotone dependencies.

What works for non-monotone nonlinear dependencies is Kendall's rank correlation coefficient, τ. Under H_0 being true (independence of X and Y), Kendall's tau is asymptotically normal, allows to do a test:

$$\frac{\tau}{\sqrt{\mathbb{V}\tau}} \xrightarrow{d} \mathcal{N}(0, 1)$$

3. (1 point) Find $\rho_S(X, Z)$ and p-value with 'scipy.stats.spearmanr'
4. (1 point) Find $\tau(X, Z)$ and p-value with 'scipy.stats.kendalltau'