AI and ML for Cybersecurity
Midterm Exam
Teimuraz Chachava

Finding the Correlation

1. Introduction

Correlation analysis is a statistical method used to determine the strength and direction of the relationship between two variables. In this task, we were asked to find Pearson's correlation coefficient for the given dataset and visualize the results on a scatter plot.

---

2. Dataset

The dataset consists of two variables XXX and YYY:

X	Y
-5	2
-3	-1
-4	-4
-1	1
1	-2
3	1
5	-3
7	-2

---

3. Methodology

The **Pearson correlation coefficient** (rrr) measures the linear relationship between two continuous variables. Its formula is:

r=cov(X,Y)σX⋅σYr = \frac{\text{cov}(X,Y)}{\sigma_X \cdot \sigma_Y}r=σX⋅σYcov(X,Y)

Where:

- cov(X,Y)\text{cov}(X,Y)cov(X,Y) = covariance between XXX and YYY
- σX,σY\sigma_X, \sigma_YσX,σY = standard deviations of XXX and YYY

We computed it using the `pearsonr()` function from **SciPy**, which also provides the **p-value** (the probability of observing the correlation by chance if the null hypothesis of no correlation is true).

---

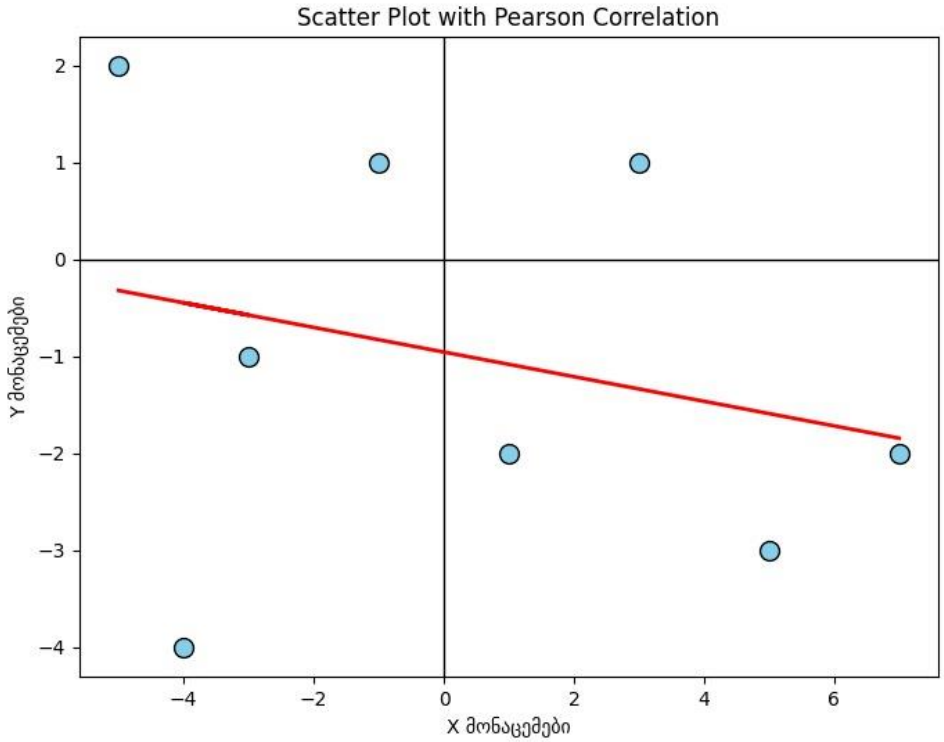
4. Results

The computed values are:

- **Pearson correlation coefficient (r): -0.3029**
- **P-value: 0.4520**

Interpretation:

- The value of rrr is approximately **-0.30**, which indicates a **weak negative linear relationship** between XXX and YYY.
- The p-value is greater than 0.05, meaning the correlation is **not statistically significant** at the 95% confidence level. In other words, the observed weak correlation could be due to random chance.


Scatter Plot with Pearson Correlation
2
1
0
−1
−2
−3
−4
Y მონაცემები
−4
−2
0
2
4
6
X მონაცემები

1. Load and Inspect the Data

- The CSV file `t_chachava2024_615387.csv` is read into a Pandas DataFrame.
- A preview (`head()`) and the dataset's shape are printed so you can check what the data looks like and how many rows/columns it has.

---

2. Select Features and Target

- Four input features are chosen:
  - `words` → total number of words in an email
  - `links` → number of links in an email
  - `capital_words` → number of all-uppercase words
  - `spam_word_count` → number of "spammy" keywords (like *free*, *win*, *prize*, etc.)
- The target column is `is_spam` (1 for spam, 0 for legit).

---

3. Split into Training and Test Sets

- Data is split:
  - 70% → training
  - 30% → testing
- `stratify=y` ensures the spam/legit ratio is preserved.
- `random_state=42` makes the split reproducible.

---

4. Train Logistic Regression Model

- A **Logistic Regression** classifier is created with `max_iter=1000`.
- The model is trained (`fit`) on the training data.
- The trained model is saved with `joblib.dump()` so it can be reused later without retraining.

---

5. Inspect the Model

- The coefficients (weights) for each feature are printed, showing how much each contributes to predicting spam.
- The model intercept is also shown.

---

6. Evaluate the Model

- The model makes predictions on the test set.
- A **confusion matrix** is created to compare predicted vs actual labels.
- Accuracy is calculated and printed.
- A graphical **Confusion Matrix Display** is plotted with Matplotlib.

---

7. Feature Extraction Function

- A helper function `extract_features_from_text()` is defined.
  It takes a raw email string and calculates the same four features:
  - Word count
  - Number of links
  - Count of all-uppercase words
  - Number of spammy keywords (from a predefined list)
- The function outputs these values as a Pandas DataFrame so they can be fed into the model.

---

8. Test on Sample Emails

- Two example emails are defined:
  - **Spam email** → contains many "FREE", "WIN", "cash", "offer" words and links.
  - **Legit email** → a normal work-related message.
- For each:
  - Features are extracted with the helper function.
  - The model predicts whether it's spam (`1`) or legit (`0`).
  - The probability of being spam is also shown.