Intro to Data Science

Project G1: Spam email classification

Martin Kivisikk
Margaret Pütsepp
Ella Hiedel

Link to the public repository: github.com/martinkivisikk/IDS23Project

Task 2. Business understanding

Background: The project aims to address the pervasive issue of spam emails through the development of an effective classification model. Spam emails pose a threat to both individuals and organizations, leading to productivity loss, security concerns, and potential financial risks.

Business Goals:

1. **Reduce Email Disturbance:** Minimize the impact of spam emails on users' productivity by accurately identifying and filtering out unwanted emails.
2. **Enhance Security:** Improve the overall email security for users by reducing the likelihood of malicious content and phishing attempts.

Business Success Criteria:

- Achieve a classification accuracy of at least 95% on the testing dataset.
- Decrease the false positive rate to less than 2%, ensuring that legitimate emails are not incorrectly classified as spam.

Assessing the situation

Inventory of Resources:

- **Data:** Utilize the provided dataset (combined_data.csv) containing labeled emails.
- **Technology:** Employ machine learning frameworks such as scikit-learn and tools like Jupyter Notebooks for model development. Also using Flask as the back-end for the web application.

Requirements, Assumptions, and Constraints:

- **Requirements:** Access to high-quality labeled data, computational resources for model training, and expertise in machine learning.
- **Assumptions:** The assumption that the provided dataset is representative of the email environment and contains relevant features for spam classification.

- **Constraints:** Limited computational resources may restrict the complexity and size of the chosen machine learning model.

Risks and Contingencies:

- **Risk:** The model may perform poorly if the dataset is not representative or lacks diversity.
- **Contingency:** Implement robust data augmentation techniques and explore additional external datasets to address limitations in the provided dataset.

Terminology:

- **Spam Score:** A numerical value indicating the likelihood of an email being spam.
- **False Positive:** Legitimate emails incorrectly classified as spam.
- **False Negative:** Spam emails incorrectly classified as legitimate.

Costs and Benefits:

- **Costs:** Computational resources, time.
- **Benefits:** Improved user experience, enhanced email security, and increased productivity for individuals and organizations.

Defining the data-mining goals

Data-Mining Goals:

1. **Develop a Robust Classification Model:** Build a machine learning model capable of accurately classifying emails as spam or non-spam.
2. **Optimize Hyperparameters:** Fine-tune the model's hyperparameters to maximize performance.

Data-Mining Success Criteria:

- Achieve a high classification accuracy on both the training and testing datasets.
- Successfully optimize hyperparameters to improve model performance.

In summary, this project aims to leverage machine learning to create a robust spam email classification model, ultimately reducing email disturbance and enhancing overall email security for users. The success of the project will be measured based on the achieved classification accuracy and the model's ability to minimize false positives.

Task 3. Data understanding

Gathering data

For our project, we used a dataset that consisted of around 83 500 emails available on Kaggle, a popular platform for machine learning datasets (Spam Email Classification Dataset (kaggle.com)). The dataset has 83 446 rows and 2 columns, label and text. The label '1' indicates that the email is classified as spam and '0' denotes that the email is legitimate (ham). The text column contains the content of the email messages. On basic initial analysis, the spam mail to non-spam mail ratio was roughly 50/50.

Outline data requirements

- All emails need to be labelled.
- The database should be rather large (over 10 000 items) to ensure the robustness and generalizability of our model,
- The database should be balanced (spam to non-spam) to avoid any biases or skewing.
- The content of the test emails should be abstract enough to make broad conclusions that would perform well on real-world data.
- The data should be rather beginner friendly- no extensive metadata, complicated preprocessing etc.

Verify data availability

We chose a Kaggle dataset that was easy to access and work on. We also tested our models on similar databases of spam and non-spam to test performance on completely new data.

Define selection criteria

About the selection of database- we chose something that fit all of our requirements. This high rated Kaggle dataset emerged as a suitable choice, meeting our criteria for labeling, size, balance, and abstraction in testing. This made the initial stages of the project easy and set a solid foundation for subsequent tasks.

About selecting which emails were spam or not- the emails were pre-labelled, which saved us a lot of time.

Describing data

We worked with recognizing text patterns as we did not have any metadata (subject, sender, receiver, timestamp). The emails themselves are quite abstract. For example the following email is legitimate- “larry king live at escapenumber escapenumber p m et on friday june escapenumber…” and the next is spam- “dear sir / madam , we are glad to offer you a broad range of databases…”.

Exploring data

Among some interesting features, some emails had text in foreign languages such as spanish, the word “escapenumber” occurred over 10 000 times, there are different links, weird spellings (“c _ i _ a _ l _ i _ s”), symbols (“! #-”) and misspelt words.

Verifying data quality

We think that working with rather chaotic, but extensive and well formatted data makes it rather interesting to see what rules our models derive from it.

Task 4. Project plan

Tasks

1. **Data Preparation and Cleaning:**
    - *Tasks:*
        - Handle missing values, outliers, and duplicate records.
        - Perform text cleaning and preprocessing on email content.
    - *Time Allocation:*
        - Martin: 1 hour
        - Margaret: 2 hours
        - Ella:
2. **Model Development and Evaluation:**
    - *Tasks:*
        - Implement various machine learning models (Random Forest, Support Vector Machines, Naive Bayes).
        - Train and evaluate models using cross-validation.
    - *Time Allocation:*
        - Martin: 2 hours
        - Margaret: 2 hours
        - Ella: x hours
3. **Hyperparameter Tuning:**
    - *Tasks:*
        - Apply Grid Search or Random Search for hyperparameter optimization.
        - Cross-validate tuned models for robust performance.
    - *Time Allocation:*
        - Martin: 1 hour
        - Margaret: 1 hour
        - Ella:

4. **Web Application Development:**
    - *Tasks:*
        - Set up a Flask backend for the web application.
        - Implement the user interface for model selection and email input.
    - *Time Allocation:*
        - Martin: 2 hours
5. **Integration and Deployment:**
    - *Tasks:*
        - Integrate the trained model with the Flask web application.
        - Test the end-to-end functionality and fix any issues.
    - *Time Allocation:*
        - Martin: 2 hours

Methods and Tools

- **Data Preparation and Cleaning:**
    - *Methods:* Pandas for data manipulation, scikit-learn for preprocessing, Natural Language Toolkit (NLTK) for stemming and lemmatizing.
    - *Tools:* PyCharm or Jupyter Notebooks for development.
- **Model Development and Evaluation:**
    - *Methods:* Random Forest, Support Vector Machines, Naive Bayes for classification.
    - *Tools:* scikit-learn for model implementation and evaluation.
- **Hyperparameter Tuning:**
    - *Methods:* Grid Search or Random Search for hyperparameter optimization.
    - *Tools:* scikit-learn for implementing hyperparameter tuning.
- **Web Application Development:**
    - *Methods:* Flask for backend development, HTML/CSS for frontend and Jinja for templating.
    - *Tools:* PyCharm and WebStorm for development.