

## Report - Machine Learning (HWS 2024)

# Assignment 2: Logistic Regression

Arne Huckemann (ahuckema), Elise Wolf (eliwolf)

November 1, 2024

hier noch abstract einfügen

## 1. Introduction

hier noch introduction eindügen

## 2. Task 1: Dataset Statistics

In this task, the goal was to explore and preprocess the Spambase dataset, which is structured to classify emails as spam or non-spam based on 57 of features. The corresponding outputs, visualizations, and detailed data exploration results are available in the Appendix (see Section B).

### 2.1. Task 1a: Feature Distribution Analysis Using Kernel Density Estimation (KDE)

The purpose of Task 1a was to examine the distribution of the dataset's features using kernel density plots and to interpret the results of these visualizations, particularly in identifying patterns or potential outliers.

The summary statistics provided by the `describe` function from `scipy.stats` were analyzed to better understand the distributions and variability across features.

Word and character frequency features generally have low mean values, suggesting that these elements appear infrequently in emails. In contrast, the capital run length features have notably higher means, indicating that capitalized sequences are more prevalent. Variance varies significantly, with features such as `word_freq_you` and `capital_run_length_total` exhibiting the highest variability.

High skewness across most features implies right-skewed distributions, particularly for words like "parts" and the character #, indicating infrequent but notable spikes in

specific emails. Similarly, high kurtosis values (e.g., 851 for `char_freq_#` and 1348 for `capital_run_length_longest`) suggest heavy tails, highlighting the presence of outliers.

To further examine the feature distributions, KDE plots were generated to visualize density across values for each feature. These plots, provided in Figures 1 through 4 in the appendix, enable a more intuitive understanding of feature concentration and outlier potential beyond summary statistics. Key findings from the KDE plots include:

- **Density Concentration around Zero:** The majority of features exhibit high density at or near zero, indicating that these features are often absent or have minimal values in many emails. This is consistent with the nature of text data, where certain words or characters may appear sparingly across documents, thus contributing minimally to most email vectors in the dataset.
- **Capital Run Length Features:** Capitalization-related features have density distributions extending further along the x-axis, reaching the dataset’s maximum values. This distinct pattern suggests that capitalized runs are more spread out and can reach high values, possibly reflecting emphasis in specific types of emails. High values in these features could serve as strong indicators of spam, given their rarity and prominence when present.
- **Prevalence of Lower Feature Values:** Most density curves decline sharply after an initial peak, suggesting that the majority of feature values are relatively low. For instance, as shown in Figure 4, which provides a zoomed-in view of the KDE plot, most feature values are below 10. This indicates that features are typically concentrated at lower values, with only a few instances reaching higher levels, which supports the potential presence of outliers.

The KDE analysis and summary statistics reveal that the dataset exhibits substantial sparsity and concentration at low feature values, with heavy tails in select features that suggest outliers or rare events in specific emails. This distributional insight is crucial for model preparation, suggesting that certain features, especially those related to capital run length, may be particularly indicative in distinguishing spam emails. Given the sparsity and right-skewness, transformations or scaling techniques may improve feature representation in subsequent modeling stages.

## 2.2. Task 1c: Gaussian KDE after Normalization

In this task, we revisited the kernel density plot after normalizing the dataset to assess the impact of the normalization process on the feature distributions. The results of this analysis can be visualized in Figures 7 and 8 in the appendix.

The normalized kernel density plot reveals that all features exhibit density peaks closely centered around zero. This centralization indicates that after normalization, the majority of feature values are situated near the mean of zero, with a significantly reduced range of variance. The density peaks are notably sharp and high, suggesting that the distribution of feature values is highly concentrated around this mean. This sharpness

arises from the normalization process, which has effectively reduced the spread of data values across the features.

Moreover, the x-axis range in the normalized plot is more uniform and confined, spanning approximately from -1 to 3, in stark contrast to the unnormalized data that extended from -1 to 45. This consistency across features simplifies the comparison of density distributions, allowing for clearer visualization of patterns without the distortions introduced by varying scales in the original data.

In the unnormalized kernel density plot, the features displayed a broader spread of densities, with peaks varying widely and some features extending up to high maximum values, such as 15841. This variability resulted in flattened density peaks, especially for features with high variance. The skewness in the unnormalized distributions, with many features concentrated near zero and long tails extending to higher values, highlighted the influence of large maximum values on the overall distribution.

The normalized kernel density estimate (KDE) significantly improves the interpretability of the feature distributions. With the skewness caused by extreme values removed, each feature's density distribution appears sharper and more centralized around the mean. The normalization process has effectively standardized the spread of values across features, allowing for more meaningful comparisons and enhancing the clarity of the underlying patterns within the dataset.

Additionally, we analyzed the correlation matrix of the normalized features, as depicted in Figure 9. The diagonal elements of the correlation matrix are all equal to one, which is expected since each feature is perfectly correlated with itself. However, the matrix also reveals high positive correlations among some off-diagonal elements, indicating that certain features tend to increase together. This might suggest that these features capture similar information or represent related patterns within the data. Conversely, we observe some high negative correlations, signifying that as one feature increases, the other tends to decrease, indicating opposing trends.

In conclusion, the normalization process has transformed the feature distributions, centering them around zero and enhancing the concentration of density peaks. This allows for clearer interpretation and comparison of the features. The correlation matrix further underscores important relationships among features, revealing both strong and weak correlations that can inform future analysis and model selection. Overall, the insights gained from the normalized data contribute significantly to understanding the underlying patterns within the dataset, enabling more effective machine learning modeling strategies.

### 3. Task 2

#### 3.1. Task 2a: ML estimate stays the same

Consider a logistic regression model with a bias term:

$$\text{logit}(P(y = 1|x)) = \beta_0 + \beta^\top \mathbf{x}$$

where  $\mathbf{x} = (x_1, x_2, \dots, x_p)^\top$  is the feature vector, and  $\beta = (\beta_1, \beta_2, \dots, \beta_p)^\top$  is the vector of coefficients.

Now, let each feature  $x_j$  be rescaled and shifted to  $x'_j = a_j x_j + b_j$ , where  $a_j$  and  $b_j$  are constants. Define  $\mathbf{x}'$  as the vector of transformed features, so:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$$

where  $\mathbf{A}$  is a diagonal matrix with elements  $a_j$  on the diagonal, and  $\mathbf{b} = (b_1, b_2, \dots, b_p)^\top$  is the vector of shifts.

The model with transformed features becomes:  $\text{logit}(P(y = 1|\mathbf{x}')) = \beta'_0 + \beta'^\top \mathbf{x}'$

Substituting  $\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$ :  $\text{logit}(P(y = 1|\mathbf{x})) = \beta'_0 + \beta'^\top (\mathbf{A}\mathbf{x} + \mathbf{b})$

Expanding the terms, we get:  $\text{logit}(P(y = 1|\mathbf{x})) = \beta'_0 + \beta'^\top \mathbf{A}\mathbf{x} + \beta'^\top \mathbf{b}$

Let  $\beta_0 = \beta'_0 + \beta'^\top \mathbf{b}$  and  $\beta = \beta'^\top \mathbf{A}$ . This redefinition works because we can absorb the effects of rescaling and shifting into the coefficients, allowing us to express the transformed model in the same form as the original. Then we have:  $\text{logit}(P(y = 1|\mathbf{x})) = \beta_0 + \beta^\top \mathbf{x}$

This shows that the transformed model can be written in the same form as the original model, with adjusted coefficients  $\beta_0$  and  $\beta$ . Thus, the likelihood function remains the same, and the ML estimates are invariant to rescaling and shifting of the features.

Computing z-scores standardizes the features to have a mean of 0 and a standard deviation of 1. This is important for several reasons. Standardizing features can improve the numerical stability of optimization algorithms used in ML estimation. Additionally, standardized coefficients are easier to interpret, as they represent the effect of a one standard deviation change in the feature. Furthermore, standardizing allows for direct comparison of the importance of different features.

## 3.2. Task 2e: Comparison of Gradient Descent and Stochastic Gradient Descent Methods

This task investigates the behavior of gradient descent (GD) and stochastic gradient descent (SGD) in optimizing a logistic regression model. Appendix C provides plots (Figures 10 to 13) that track the negative log-likelihood and learning rate dynamics over multiple epochs for each method.

### 3.2.1. Gradient Descent (GD) Behavior

Gradient descent (GD) updates parameters at each epoch based on the gradient calculated over the entire dataset:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \eta \nabla \ell(\mathbf{w}) \quad (1)$$

where  $\eta$  is the learning rate and  $\nabla \ell(\mathbf{w}) = \mathbf{X}^T(\mathbf{y} - \sigma(\mathbf{X}\mathbf{w}))$  denotes the gradient.

As shown in Figure 10, GD initially decreases the negative log-likelihood sharply, indicating rapid improvement. However, around epoch 10, there is a noticeable spike due to the bold driver heuristic, which adjusts the learning rate following increases in

the cost function. After this adjustment, the log-likelihood stabilizes around 700, and the learning rate plot reflects a decrease as GD fine-tunes near the minimum.

### 3.2.2. Stochastic Gradient Descent (SGD) Behavior

Stochastic gradient descent (SGD) updates parameters after each individual observation, using:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \epsilon_n \hat{\nabla} \ell(w|\mathbf{x}_i, y_i) \quad (2)$$

where  $\epsilon_n$  is the learning rate and  $\hat{\nabla} \ell(w|\mathbf{x}_i, y_i)$  is the gradient for a single data point.

The plot in Figure 11 shows that, while SGD also starts with a rapid reduction in the negative log-likelihood, its progression is smoother and lacks large corrective spikes. This is due to the smaller, more frequent updates. The learning rate gradually decreases from around 0.013 to a stable minimum near 0.001, allowing SGD to converge steadily around the same log-likelihood minimum as GD.

### 3.2.3. Comparative Analysis of GD and SGD

Both GD and SGD methods effectively reduce the negative log-likelihood to the same final plateau near 700, indicating convergence to a similar solution. However, each method's unique approach results in distinct optimization behaviors. By using full-batch gradients, GD has larger but less frequent parameter updates, requiring correction when a large step increases the cost. The bold driver heuristic triggers oscillations in the learning rate, leading to temporary spikes in the cost function, as seen around epoch 10. Despite the adaptive rate, GD's larger steps slow convergence near the minimum. With more frequent, incremental updates on individual observations, SGD avoids major spikes, resulting in a smooth convergence path. The gradual decrease in learning rate supports precision near the minimum, leading to faster initial convergence and steadier progress near the optimum without large corrections.

The comparison between GD and SGD highlights each method's unique strengths in solving the minimization problem. GD's deterministic path with full-batch updates provides stability but requires large, occasional corrections that slow convergence. In contrast, SGD's frequent small-batch updates allow for a rapid initial decrease in cost, with more controlled, steady adjustments near the minimum due to gradual rate decay.

In sum, both GD and SGD achieve similar final results, though SGD's stochastic nature and adaptive learning rate yield smoother convergence with faster initial progress, ideal for large datasets where quick approximate updates are beneficial.

## 3.3. Task 3: Predict and Classify Methods for Spam Classification

In this task, we completed the `predict` and `classify` methods, which were used to predict the spam probability and classify emails as spam or non-spam based on the models fitted in Task 2. The complete set of results, including confusion matrices, metrics, and performance curves, is provided in the Appendix D, where Figure 13 is

given for the precision-recall curve and the results as well as an intuitive explanation for the Confusion Matrix and the calculated Evaluation Metrics is provided.

Overall, the metrics in Appendix D indicate that the model effectively balances spam detection with minimizing false positives. With both GD and SGD showing a similar 90% accuracy, the choice between them depends on whether the priority is higher spam detection (SGD) or minimizing false positives (GD).

### 3.3.1. Precision-Recall Curve Analysis

The precision-recall curve (see Figure 13 in the Appendix) was plotted to visualize the trade-off between precision and recall for GD and SGD. Both GD and SGD start with high precision at low recall, suggesting that the models are initially conservative and only classify the most likely spam messages. As recall increases, precision drops due to a rise in false positives. This pattern reflects a standard trade-off in binary classification tasks. Both GD and SGD converge to similar performance levels, with SGD displaying minor fluctuations due to its stochastic updates. GD’s curve is smoother, reflecting the consistency of full-batch updates. This curve suggests both models effectively balance precision and recall, with GD offering stability and SGD offering faster convergence—useful for tasks requiring real-time or iterative training.

### 3.3.2. Exploration of Model Weight Composition

The composition of the weight vectors for GD and SGD was analyzed to determine which features are most influential in predicting spam. Features with high positive weights increase the likelihood of an email being classified as spam, while those with high negative weights reduce it.

- **Most Influential Features:** Features like `word_freq_3d`, `capital_run_length_longest`, and `char_freq_$` have high positive weights in both GD and SGD. These features intuitively align with common spam characteristics, such as the frequent use of capital letters and specific characters (e.g., the dollar sign).
- **Less Influential Features:** Features like `word_freq_make`, `word_freq_650`, and `word_freq_labs` have minimal impact on model predictions, with near-zero weights. This consistency in low weights across both models suggests that these features do not significantly differentiate between spam and non-spam, likely due to commonality in both types.
- **Negative Weight Features:** Features such as `word_freq_hp` (frequently associated with legitimate emails) have high negative weights, reducing the likelihood of an email being classified as spam. This suggests that terms common in legitimate emails serve as strong indicators for the non-spam class.

The weight vector analysis confirms that both GD and SGD models emphasize similar features for spam classification, highlighting intuitive factors like capital letters and

certain character frequencies. The consistency in high and low-weighted features across both models reflects a robust selection of relevant spam indicators, suggesting the models are effectively tuned to distinguish spam. Overall, the evaluation of the weight vectors, confusion matrices, precision-recall curves, and evaluation metrics demonstrates that both GD and SGD are effective for spam classification. SGD may be slightly preferable when the primary objective is to maximize recall (detect more spam), whereas GD offers stability with marginally higher precision in identifying non-spam. This task’s analyses demonstrate the effectiveness of both models for spam classification, and the choice between GD and SGD ultimately depends on specific application requirements.

## 4. Task 4

Task 4 focuses on analyzing the behavior and impact of regularization on the weight vector of a logistic regression model by varying the regularization parameter,  $\lambda$ . This exploration aims to understand how different  $\lambda$  values influence the model’s complexity, generalization ability, and convergence behavior across both gradient descent (GD) and stochastic gradient descent (SGD) methods. This section refers to the plots provided in the Appendix E, namely Figures 14 to 17, which illustrate the effect of  $\lambda$  on model performance.

### 4.1. Task 4b: Effect of the L2 Regularization Hyperparameter $\lambda$ on Model Performance

In Task 4b, we investigate how the choice of the L2 regularization hyperparameter  $\lambda$  impacts the effectiveness of logistic regression in a MAP estimation context. Here, the value of  $\lambda$  controls the strength of the Gaussian prior (or equivalently, the degree of L2 regularization) applied to the model parameters.

#### 4.1.1. Impact on Training and Test Log-Likelihoods

The training and test log-likelihoods display distinct trends as  $\lambda$  varies. As seen in Figure 17, the training log-likelihood initially begins at a lower value than the test log-likelihood, indicating a closer fit to the training data when  $\lambda$  is minimal. With an increase in  $\lambda$ , both log-likelihoods decrease, suggesting that the regularization begins to mitigate overfitting by penalizing large weights, which are often specific to the training set.

For low values of  $\lambda$ , we observe minimal change in the log-likelihoods, indicating that the model can fit the training data closely without significant regularization constraints. However, as  $\lambda$  grows to moderate levels, we see a gradual decline in both the training and test log-likelihoods, with a more substantial reduction around  $\lambda = 10$  and  $\lambda = 100$ . This reduction reflects an improvement in the generalization ability, as regularization imposes a limit on the model complexity. Notably, when  $\lambda$  becomes excessively large, the regularization penalty dominates, leading to a significant decrease in training log-likelihood and a further drop in test log-likelihood. This observation demonstrates how

excessive regularization can restrict the model’s capacity to capture meaningful patterns in the data.

The divergence between training and test log-likelihoods across  $\lambda$  values shows that higher  $\lambda$  values promote better generalization, as the test log-likelihood stabilizes closer to the training log-likelihood. This effect supports the conclusion that increasing  $\lambda$  improves generalization but, beyond a critical threshold, leads to underfitting due to excessive penalization of parameter values.

#### 4.1.2. Effect on Classification Accuracy

Classification accuracy shows a non-linear dependence on  $\lambda$ , as indicated in Figure 17. Initially, with  $\lambda \approx 0$ , accuracy remains around 0.9, reflecting high accuracy with minimal regularization. In the range of  $\lambda = 1$  to  $\lambda = 10$ , accuracy improves slightly, reaching a peak of approximately 0.91, suggesting that moderate regularization supports an optimal balance between model complexity and generalization. However, as  $\lambda$  increases beyond this point, accuracy declines sharply, reaching approximately 0.61 for very large  $\lambda$  values. This decline illustrates that, while moderate  $\lambda$  values enhance generalization, excessive regularization impairs the model’s predictive power by oversimplifying the model and reducing its ability to differentiate between classes effectively.

#### 4.2. Precision, Recall, F1, and AUC Analysis

Appendix E.1 (Figure 17) presents precision, recall, F1, and AUC results across varying  $\lambda$  values.

At lower  $\lambda$  values, precision, recall, and F1 remain steady, peaking around  $\lambda$  values of 1 to 10 where regularization optimally balances overfitting and generalization. For higher  $\lambda$  values, these scores decline sharply, highlighting reduced classification effectiveness under strong regularization.

The AUC score shows minor variation across  $\lambda$ , slightly improving around moderate  $\lambda$  values and decreasing slightly at higher  $\lambda$  levels, indicating stable classification quality despite the changes in other metrics.

Optimal performance is achieved with  $\lambda$  values between 1 and 10, providing a balance between model flexibility and generalization. Extreme  $\lambda$  values limit performance across all metrics.

#### 4.3. Task 4c: Study of the Composition of the Weight Vector for Varying Choices of $\lambda$

In this task, we investigate the impact of different regularization strengths on the composition of the weight vector in a linear model. By systematically increasing the regularization parameter  $\lambda$ , we observe how the weight vector changes in response to stronger penalties. The goal is to identify how increasing  $\lambda$  affects the model’s capacity to capture patterns in the data and to determine an optimal balance between complexity and simplicity in the model. The plots illustrating these weight compositions for each tested  $\lambda$  value are provided in the appendix.



For this relatively low value of  $\lambda$ , the weights are large and diverse, indicating that the model captures substantial patterns in the data. Although the penalty is applied, it is not overly restrictive, allowing the model to retain a level of complexity that fits the training data well. With low regularization, the model retains substantial complexity, as indicated by larger weight values. This allows the model to fit well to the training data, but there is an increased risk of capturing noise, leading to potential overfitting.

As  $\lambda$  increases, the weights become smaller, reflecting the model's reduced complexity due to stronger regularization. This moderate  $\lambda$  penalizes the model enough to reduce the risk of overfitting without fully compromising the model's ability to capture meaningful data patterns. Moderate regularization produces smaller weights, providing a balance between fitting the data and preventing overfitting. At this level, the model generalizes well to new data, capturing essential patterns without being overly complex.

For a high  $\lambda$ , the weights are substantially reduced, demonstrating a pronounced regularization effect. The model's complexity is limited, making it less effective at fitting the finer patterns in the data. This simplification, while useful in avoiding overfitting, leads to a reduction in model performance as underfitting begins to appear. With stronger regularization, the model's weights become very small, indicating reduced complexity. This simplification often leads to underfitting as the model fails to capture more nuanced relationships within the data.

With an extremely high value of  $\lambda$ , the weights are nearly zero. This severe penalty drastically limits the model's flexibility, causing it to underfit the data as it is unable to capture even the fundamental structures within the dataset. Excessive regularization results in weights near zero, creating an overly simplistic model that severely underfits the data, failing to capture even the primary relationships within the dataset.

The observations indicate that an optimal range for  $\lambda$  exists where the model effectively balances complexity and generalization. From the analysis,  $\lambda$  values around 10000 appear to provide an ideal balance, where the weights are regularized enough to prevent overfitting while retaining the ability to capture relevant patterns. This range aligns with the principles of regularization, wherein higher values of  $\lambda$  simplify the model by shrinking weights. However, excessive regularization (very high  $\lambda$  values) results in a loss of critical data structure, leading to underfitting.

In summary, regularization offers a valuable tool for managing model complexity through  $\lambda$ . The optimal choice for  $\lambda$  lies in achieving a level where weights are neither too large (avoiding overfitting) nor too small (avoiding underfitting), thereby supporting robust performance on both training and unseen data.

## 5. Conclusions

? writes:

The closing section, or summary, is used to draw together the topics discussed in the paper. It should include a concise statement of the paper's important results and an explanation of their significance. This is an appro-

appropriate place to state (or restate) any limitations of the work: shortcomings in the experiments, problems that the theory does not address, and so on.

The conclusions are an appropriate place for a scientist to look beyond the current context to other problems that were not addressed, to questions that were not answered, to variations that could be explored. They may include speculation, such as discussion of possible consequences of the results.

A *conclusion* is that which concludes, or the end. *Conclusions* are the inferences drawn from a collection of information. Write "Conclusions", not "Conclusion". If you have no conclusions to draw, write "Summary". (?)

## A. Mathematical Groundwork

Nr.1 a): In the code that was given, a gaussian density function was fitted to the Data. Therefore, clearly the data looks gaussian distributed. There are two Problems though. The person who assigned the task was very untransparent in the simplification of the problem. To explain this consider the following notation:

Let for  $n = 3065$  and for each feature  $j = 1, \dots, 57$   $X^j := (X_1^{(j)}, \dots, X_n^{(j)})$  denote  $n$  samples from a distribution  $F_j$  on  $\mathbb{N}$ . We can simplify the problem such that we say that we only consider the distribution on  $\{0, \dots, K\}$  where  $L = \max_{i \leq n, j \leq 57} X$  and  $X = (X^1, \dots, X^n)^T$ . If we were exact we would have the following problem: for all  $j \leq 57$  we have  $X^j \sim \text{Multinom}(K; p_1, \dots, p_K)$  is the multi-binomial distribution and where  $K$  is as above. The Binomial distribution is a special case of the Multinomial, when choosing  $p_1, \dots, p_K$  accordingly. In the exercise this simplification was not communicated. Further, the plots that are given are misleading, as the data looks normally distributed, but we are clearly on a discrete space.

Therefore, the data is actually multinomial distributed, but we simplify it to a binomial distribution.

b) The assignment is again stated misleading. We can only do the normalization empirically and not exact.

c) As the normalization is only empirically we see slight deviations in each density function, because the expectation is not exactly zero and not the variance is not exactly one.

Nr.2 b):

**Definition A.1.** Eine Familie von Verteilungen heißt Exponentialfamilie mit Parameterraum  $\Theta$ , falls es Funktionen  $a : \mathbb{R} \times \mathbb{R}_+ \rightarrow \mathbb{R}$  und  $b : \Theta \rightarrow \mathbb{R}$  sowie ein Maß  $\mu$  und einen Störparameter  $\tau > 0$  gibt, so dass alle Elemente der Familie  $\{f_\theta \mid \theta \in \Theta\}$  eine Dichte bzgl.  $\mu$  sind und es gilt

$$f_\theta(y) = e^{\tau^{-2}(y\theta + a(y, \tau) - b(\theta))}$$

Der Parameter  $\Theta$  besitze mindestens 2 Elemente und es gelte

$$\Theta = \left\{ \theta \in \mathbb{R} : \int_{\mathbb{R}} e^{\tau^{-2}(y\theta + a(y, \tau))} \mu(dy) < \infty \right\}$$

Binomialverteilung mit Erfolgswahrscheinlichkeit  $p$  bei fixen  $n$  Ziehungen:  $\mu$  ist das Zählmaß auf den Zahlen  $0, 1, \dots, n$  und

$$f_p(k) = \binom{n}{k} p^k (1-p)^{n-k} = e^{k \log(p) + (n-k) \log(1-p) + \log(\frac{n!}{k!(n-k)!})}$$

Ausgrund der folgenden Eigenschaft

$$\log(1-p) = -\log\left(\frac{1}{1-p}\right) = -\log\left(\frac{1-p+p}{1-p}\right) = -\log\left(1 + \frac{p}{1-p}\right) \quad (*)$$

gilt

$$\begin{aligned} f_p(k) &= e^{k \log(p) + (n-k) \log(1-p) + \log(\frac{n!}{k!(n-k)!})} \\ &= e^{k \log(p) + n \log(1-p) - k \log(1-p) + \log(\frac{n!}{k!(n-k)!})} \\ &= e^{k \log(p) - n \log(1 + \frac{p}{1-p}) + k \log(1 + \frac{p}{1-p}) + \log(\frac{n!}{k!(n-k)!})} \end{aligned}$$

Wegen

- Es gilt  $\log(a \cdot b) = \log(a) + \log(b)$
- $p \cdot (1 + \frac{p}{1-p}) = p + \frac{p^2}{1-p} = \frac{(1-p)p + p^2}{1-p} = \frac{p}{1-p}$

können wir schließlich zeigen:

$$\begin{aligned} f_p(k) &= e^{k \log(p) - n \log(1 + \frac{p}{1-p}) + k \log(1 + \frac{p}{1-p}) + \log(\frac{n!}{k!(n-k)!})} \\ &= e^{\overbrace{k \log(\frac{p}{1-p})}^{=: k\theta} + \overbrace{\log(\frac{n!}{k!(n-k)!})}^{=: a(k,n)} - \overbrace{n \log(1 + \frac{p}{1-p})}^{=: b(\theta)}} \end{aligned}$$

Wir definieren  $\theta := \log(\frac{p}{1-p})$  und  $b(\theta) := n \log(1 + \frac{p}{1-p}) = n \log(1 + e^\theta)$ .

**Theorem A.1.** Sei eine Exponentialfamilie  $(\mathbb{P}_\theta)_{\theta \in \Theta}$  gegeben,  $\Theta$  offen und  $Y \sim \mathbb{P}_\theta$ . Sei  $\mathbb{E}_\theta Y^2 < \infty$  für alle  $\theta \in \Theta$  und  $b : \Theta \rightarrow \mathbb{R}$  sei zweimal stetig differenzierbar mit  $b''(\theta) > 0$  für alle  $\theta \in \Theta$ . Dann gilt

$$\mathbb{E}Y = b'(\theta), \quad \text{Var } Y = \tau^2 b''(\theta)$$

**Definition A.2.** Sei  $b$  zweimal stetig differenzierbar und  $b'' > 0$ . Dann heißt die Inverse  $g := (b')^{-1}$  von  $b'$ , definiert auf  $G = \{b'(\theta) : \theta \in \Theta\}$ , natürliche Linkfunktion.

**Proposition A.1.** Es gelten die Voraussetzungen von Satz 2.7. Falls das verallgemeinerte lineare Modell die natürliche Linkfunktion besitzt, so gilt  $(\theta_1, \dots, \theta_n)^\top = X\beta$ .

Seien  $Y_i, i = 1, \dots, n$ , unabhängige Zufallsvariablen mit Dichte

$$f_{\theta_i}(y) = \exp \{ \tau^{-2} (y\theta_i + a(y, \tau) - b(\theta_i)) \}$$

D.h. die Familie von Verteilungen der  $Y_i$  sind eine Exponentialfamilie. Damit können wir den Satz verwenden, der aussagt, dass  $\mathbb{E}[Y] = b'(\theta)$ , d.h.

$$\forall i = 1, \dots, n : \mathbb{E}[Y_i] = b'(\theta_i)$$

und wegen der Definition des GLM gilt

$$\forall i = 1, \dots, n : \mathbb{E}[Y_i] = g^{-1}(x_i^T \beta)$$

Also gilt insgesamt

$$b'(\theta_i) = g^{-1}(x_i^T \beta) \iff \theta_i = (b')^{-1}(g^{-1}(x_i^T \beta))$$

Das können wir auch schreiben als

$$\theta_i = (b')^{-1}(m_i)$$

und  $m_i := m_i(\beta) := g^{-1}(x_i^T \beta)$ .

Da die  $Y_i$  unabhängig voneinander sind können wir die gemeinsame Dichte faktorisieren. Dann gilt also für die log-Likelihood-Funktion:

$$l(\beta) \stackrel{\text{Def.}}{=} \log(f_{\theta}(y)) \stackrel{\text{unabh.}}{=} \log\left(\prod_{i=1}^n f_{\theta_i}(y)\right) = \sum_{i=1}^n \log(f_{\theta_i}(y)) = \tau^{-2} \sum_{i=1}^n (Y_i \theta_i + a(Y_i, \tau) - b(\theta_i)).$$

Unser Ziel ist es, das  $\operatorname{argmax}_{\beta \in \mathbb{R}} l(\beta) =: \hat{\beta}$  zu bestimmen. Dies ist dann der MLE Schätzer.

**Definition A.3.**

1. Sei

$$U_j(\beta) = \frac{\partial l(\beta)}{\partial \beta_j}, \quad j = 1, \dots, p$$

Dann heißt die  $(p \times p)$ -Matrix  $I = I(\beta)$  mit

$$I_{jk} = \mathbb{E}[U_j(\beta)U_k(\beta)], \quad j, k = 1, \dots, p$$

Dann ist  $(I_{jk})_{j,k \leq n}$  die Fisher-Informationsmatrix

Erinnerung  $S_{\theta} := \nabla_{\theta} \log(p_{\theta})$ :  $I = \int S_{\theta}(x) \cdot S_{\theta}^T(x) d\mathbb{P}_{\theta}(x)$

2. Die  $(p \times p)$ -Matrix  $W(\beta)$  mit

$$W_{jk}(\beta) = \frac{\partial^2 l(\beta)}{\partial \beta_j \partial \beta_k}, \quad j, k = 1, \dots, p$$

heißt Hesse-Matrix.

**Example A.1.** *Binomialverteilung  $\text{Bin}(n, \beta)$*

$$\begin{aligned} U(\beta) &= X^\top (Y - \beta) \\ I(\beta) &= X^\top \text{diag}(p_i (1 - \beta_i)) X \end{aligned}$$

c)

Now we can do stochastic gradient descent on the log likelihood function  $l(\beta)$ . It is as follows for  $\alpha > 0$ :

$$\beta_{n+1} := \beta_n + \alpha \nabla l(\beta) = \beta_n + \alpha U(\beta).$$

**Remark A.1.** *A sufficient condition is that  $U(\beta^*) = 0$  and a necessary condition that we have reached a global minimum is  $I(\beta^*) = 0$ . This is due to the following relation:  $W = -I$  if we consider the natural Link function (which we do).*

**Theorem A.2.** *Die Einträge der Hesse Matrix von der Maximum-Likelihood Funktion gegeben durch:*

$$W_{jk} = \sum_{i=1}^n x_{ij} x_{ik} \left( (Y_i - m_i) \nu_i - \frac{1}{(g'(m_i))^2} \sigma_i^{-2}(\beta) \right), \quad j, k = 1, \dots, p$$

mit

$$\nu_i = \tau^{-2} \frac{\partial^2 (b')^{-1}(g^{-1}(\eta_i))}{\partial^2 \eta_i}, \quad \eta_i = x_i^\top \beta$$

Ist  $g$  die natürliche Link-Funktion, so

$$W(\beta) = -I(\beta)$$

## B. Plots and Outputs for Task 1

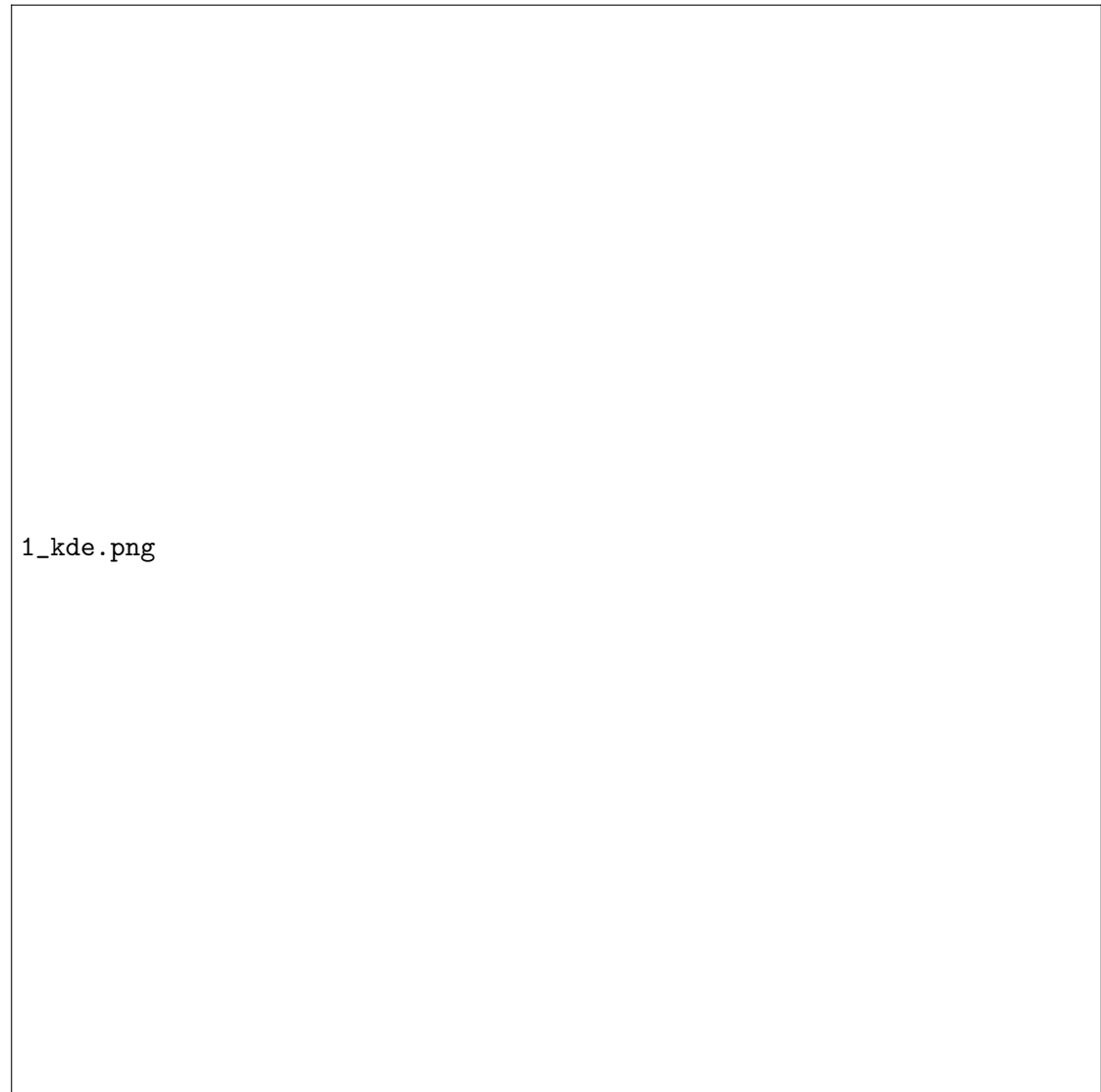
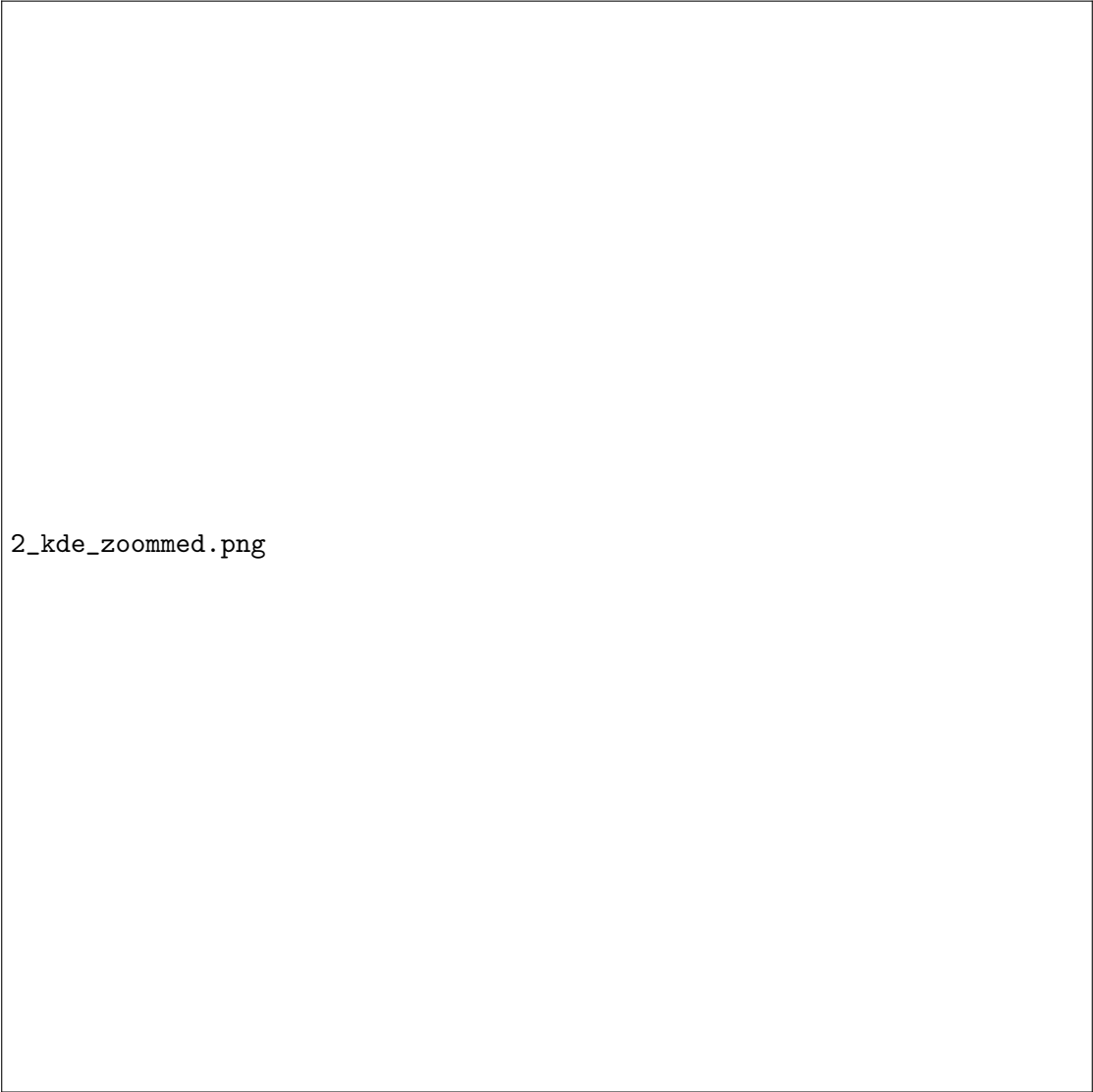


Figure 1: Gaussian Kernel Density Estimation (KDE) plot.



2\_kde\_zoommed.png

Figure 2: Zoomed in Gaussian KDE plot.

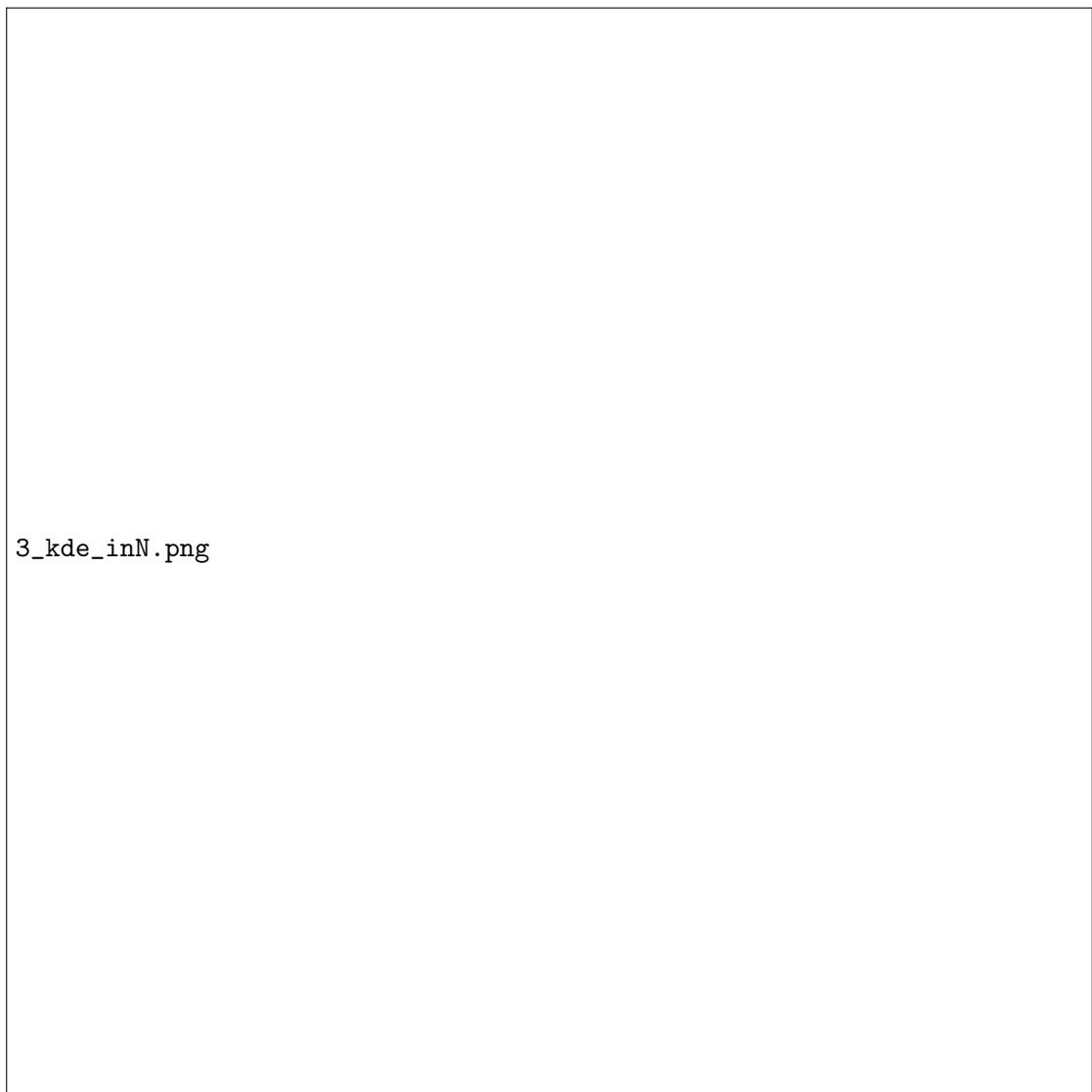


Figure 3: Gaussian KDE plot in more detailed stepsize.



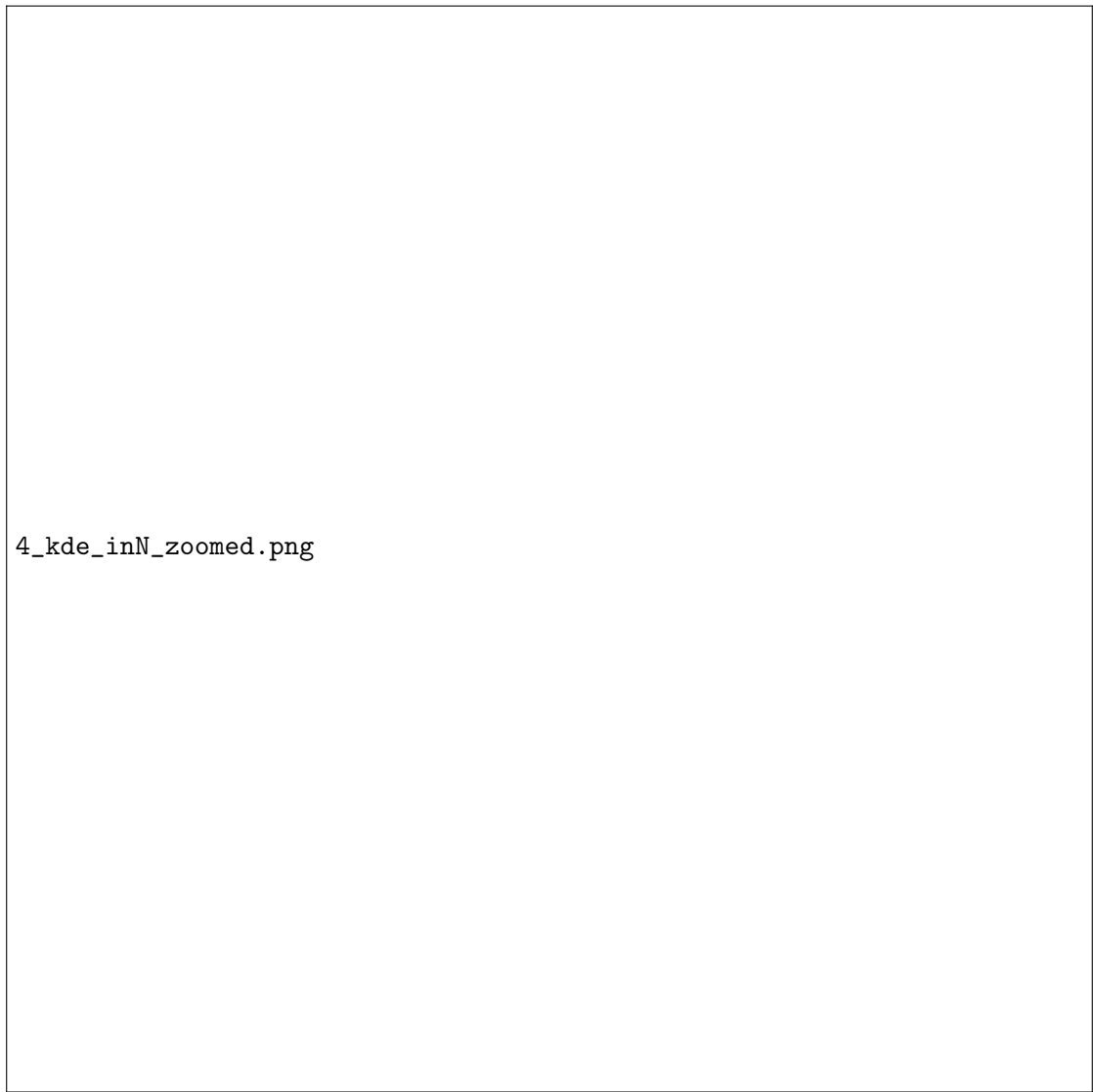


Figure 4: Zoomed in Gaussian KDE plot in more detailed stepsize.



Figure 5: Overall sum of feature occurency plotted in histogram.

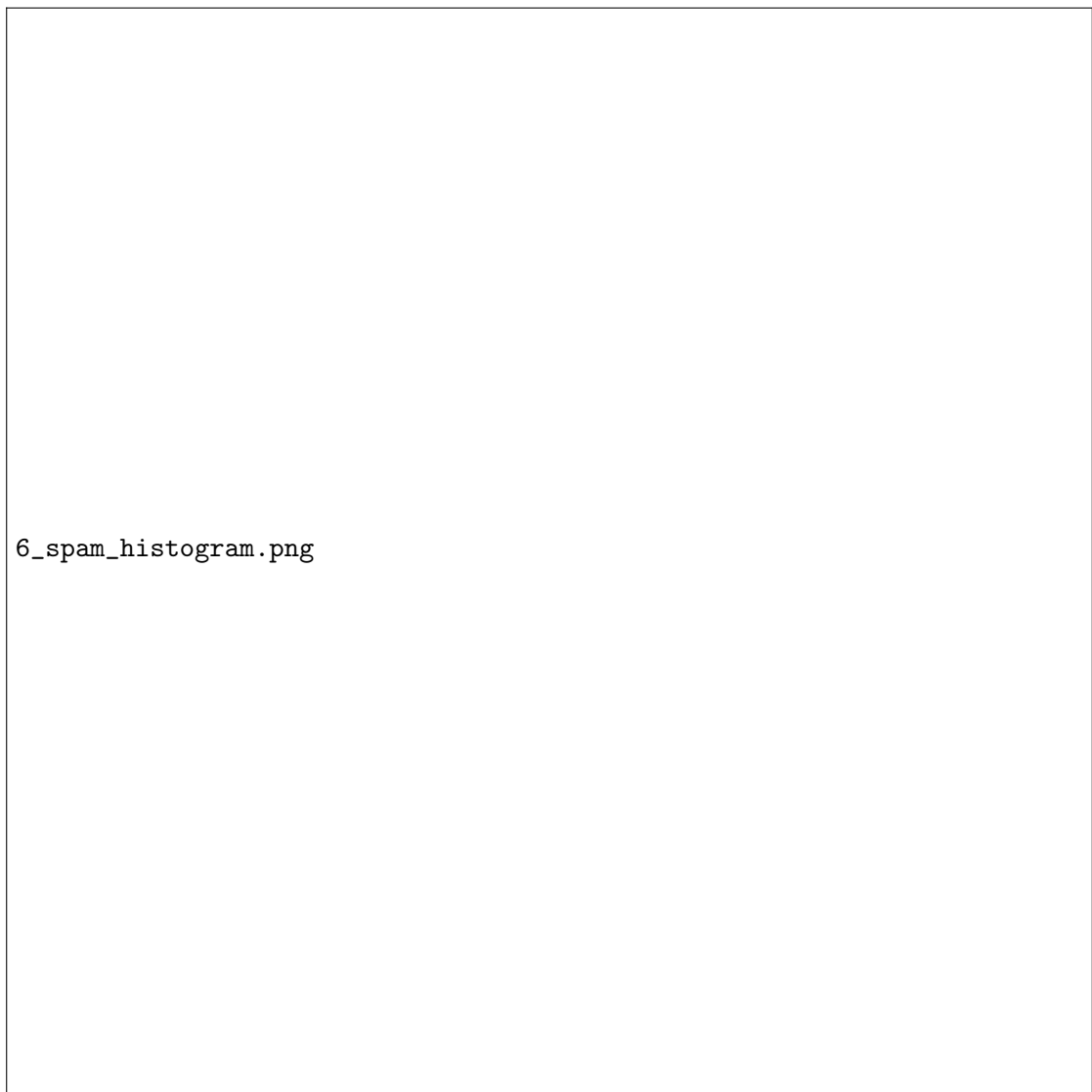


Figure 6: Sum of feature occurency plotted in histogram including all spam and including all non-spam features separately.

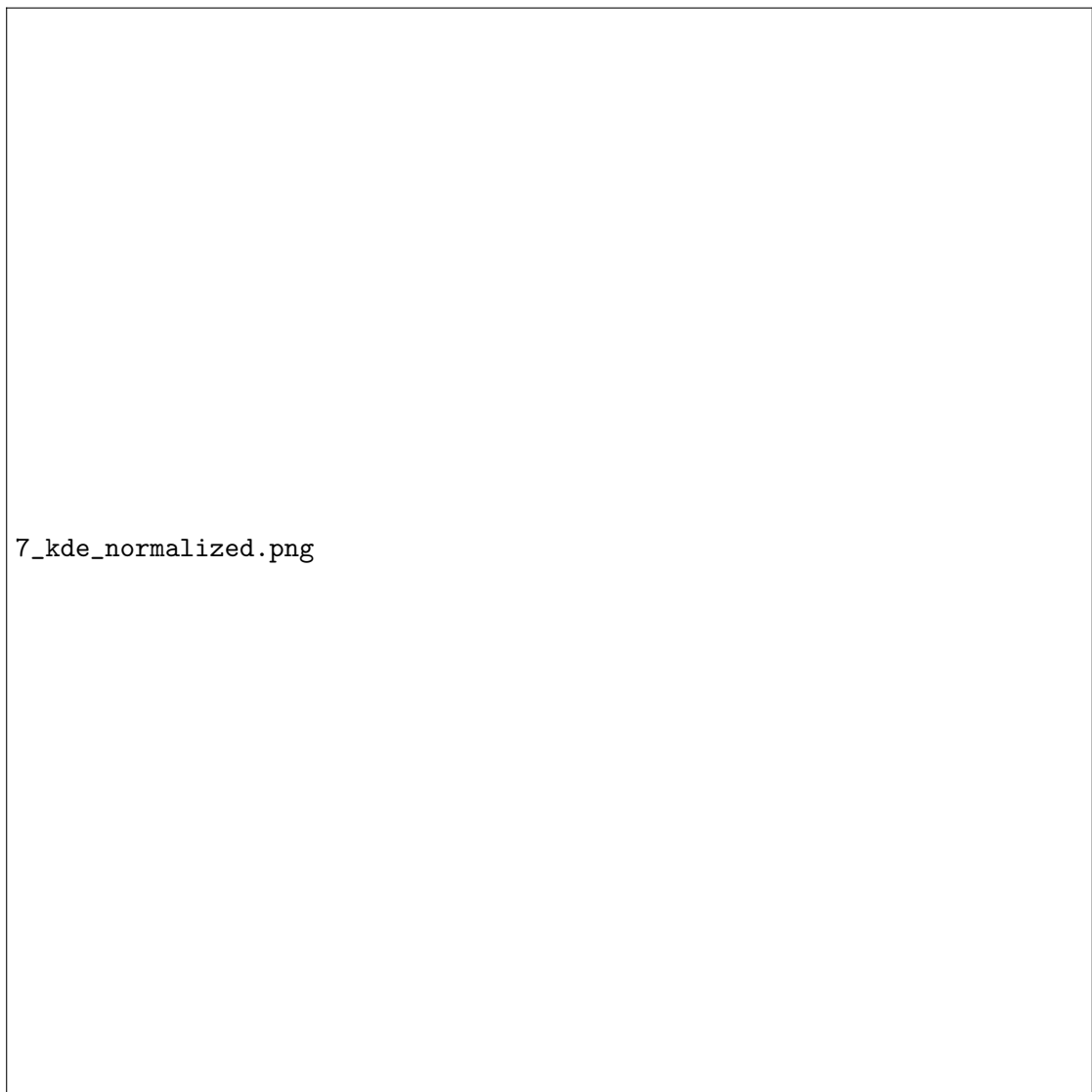


Figure 7: Gaussian KDE plot with normalized data.

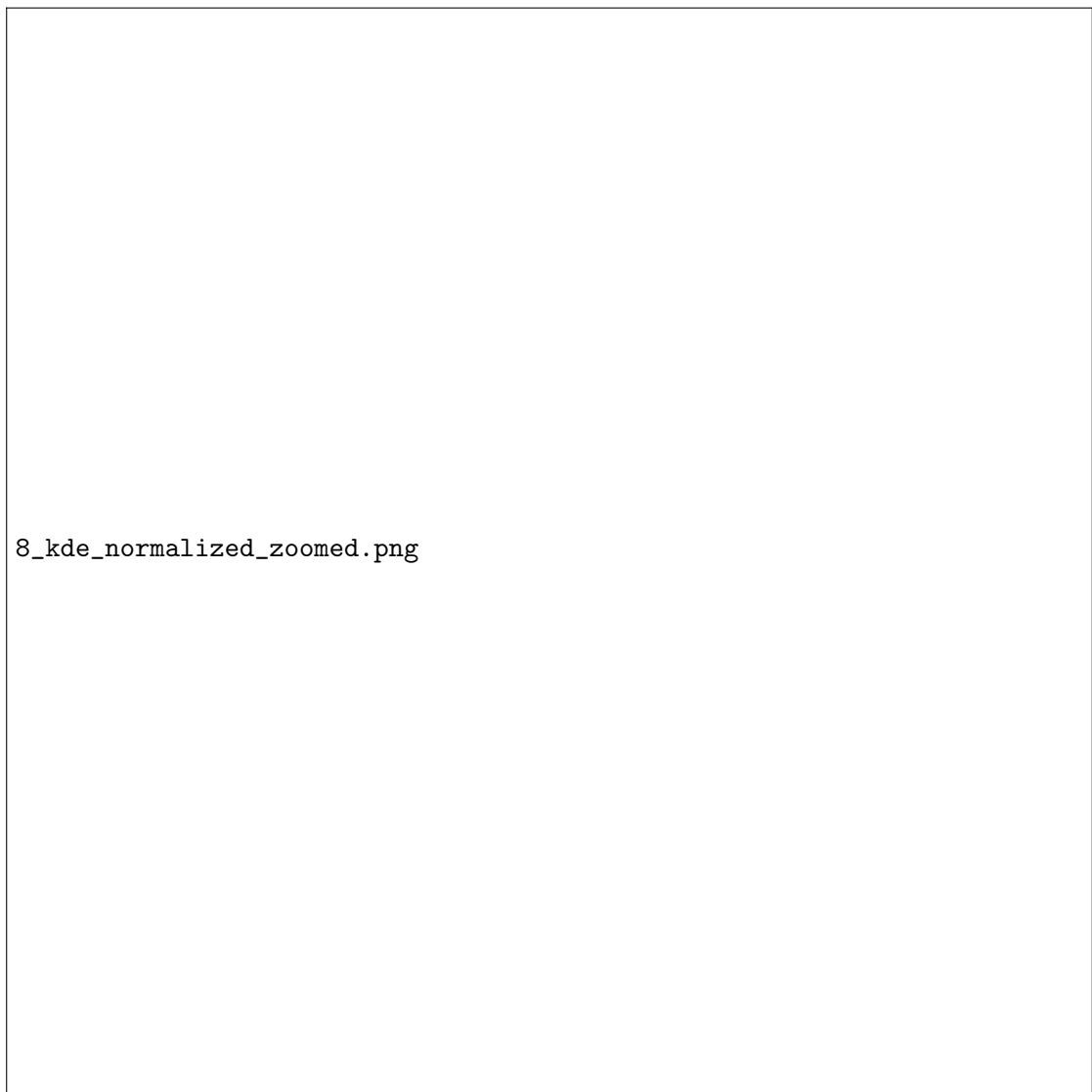
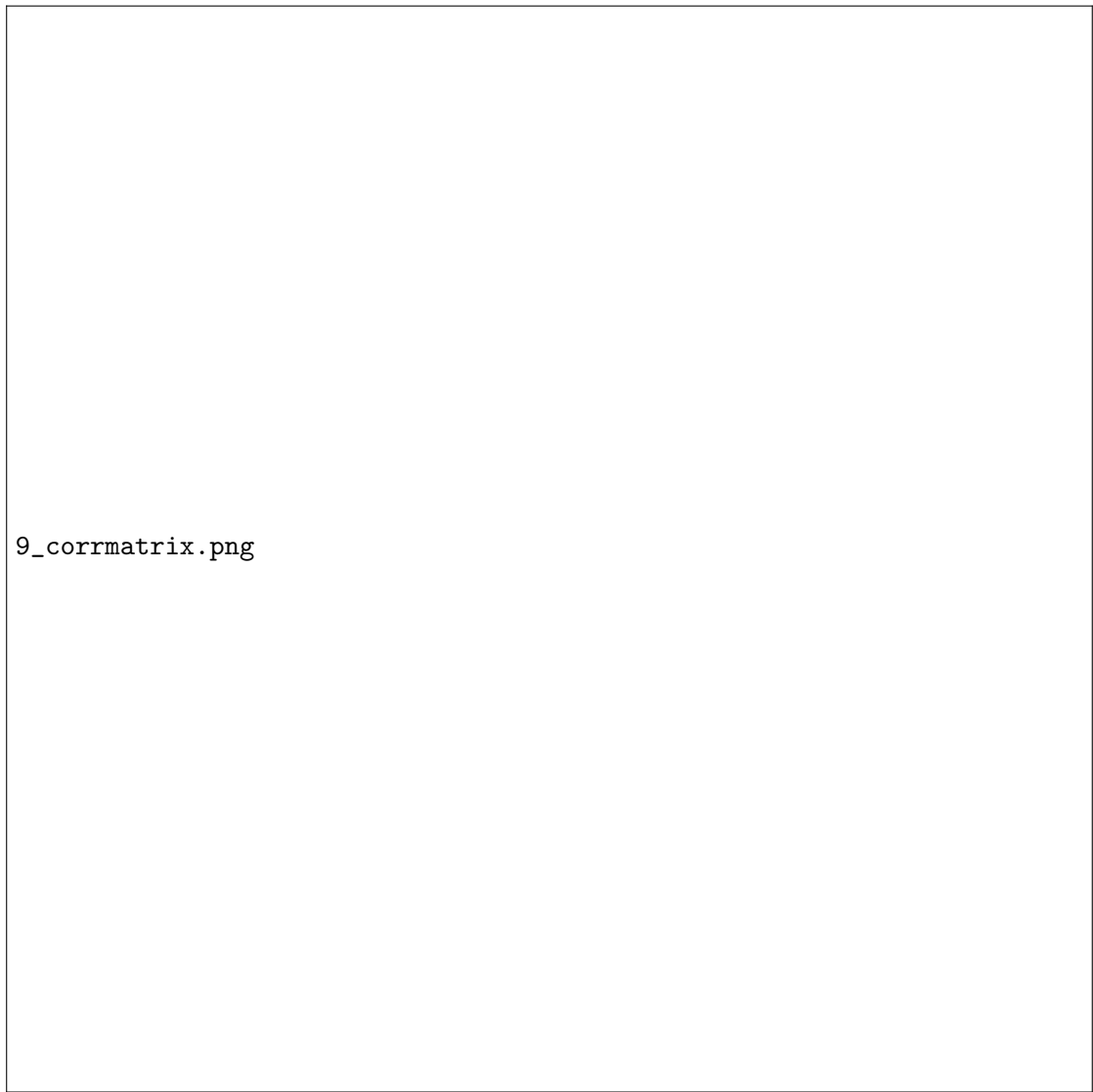


Figure 8: Zoomed in Gaussian KDE plot with normalized data.



9\_corrmatrix.png

Figure 9: Correlation matrix of normalized features.

## C. Plots and Outputs for Task 2



Figure 10: Gradient Descent progress visualized in the progress of the negative log-likelihood and the learning rate over the epochs.

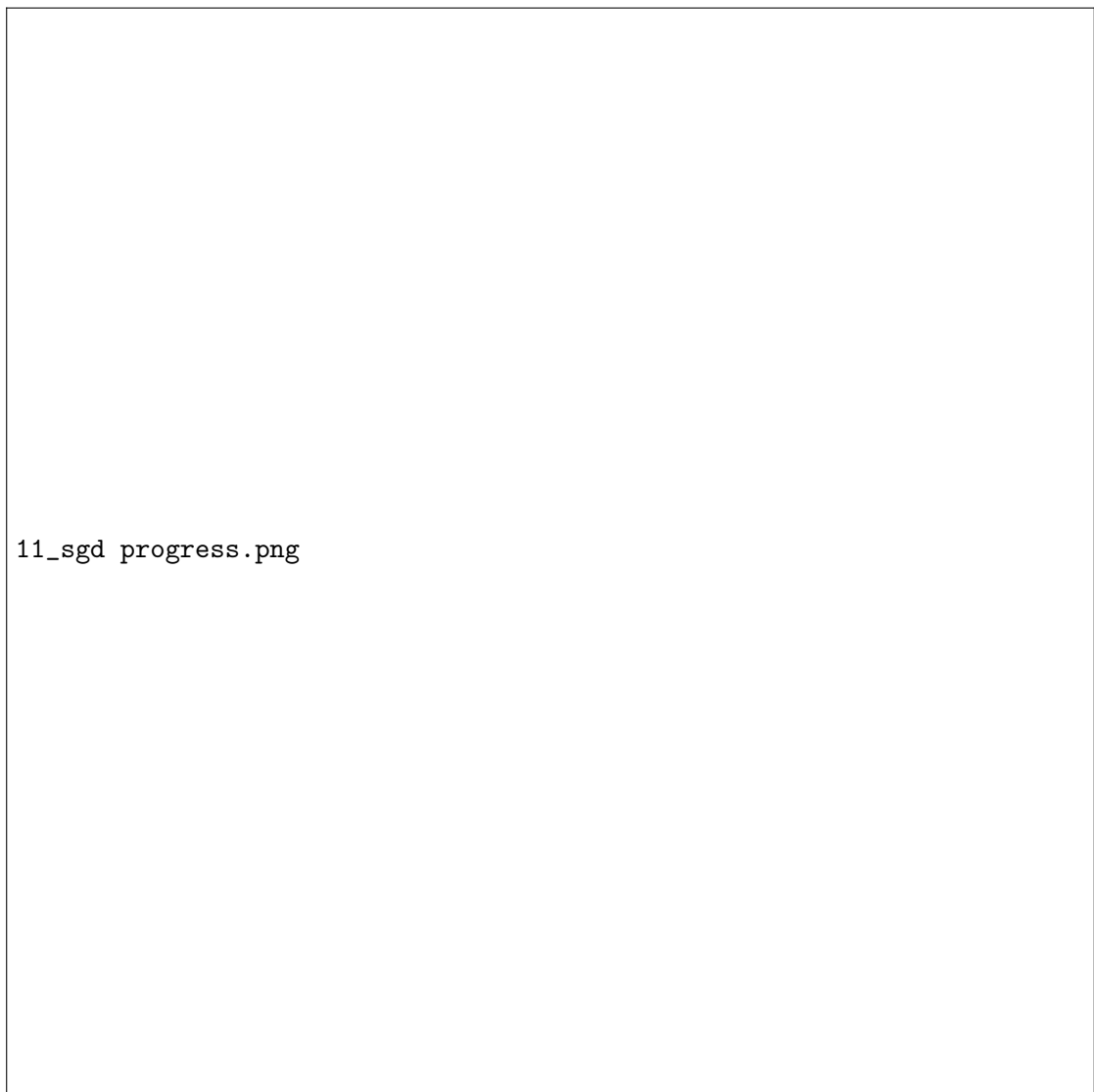
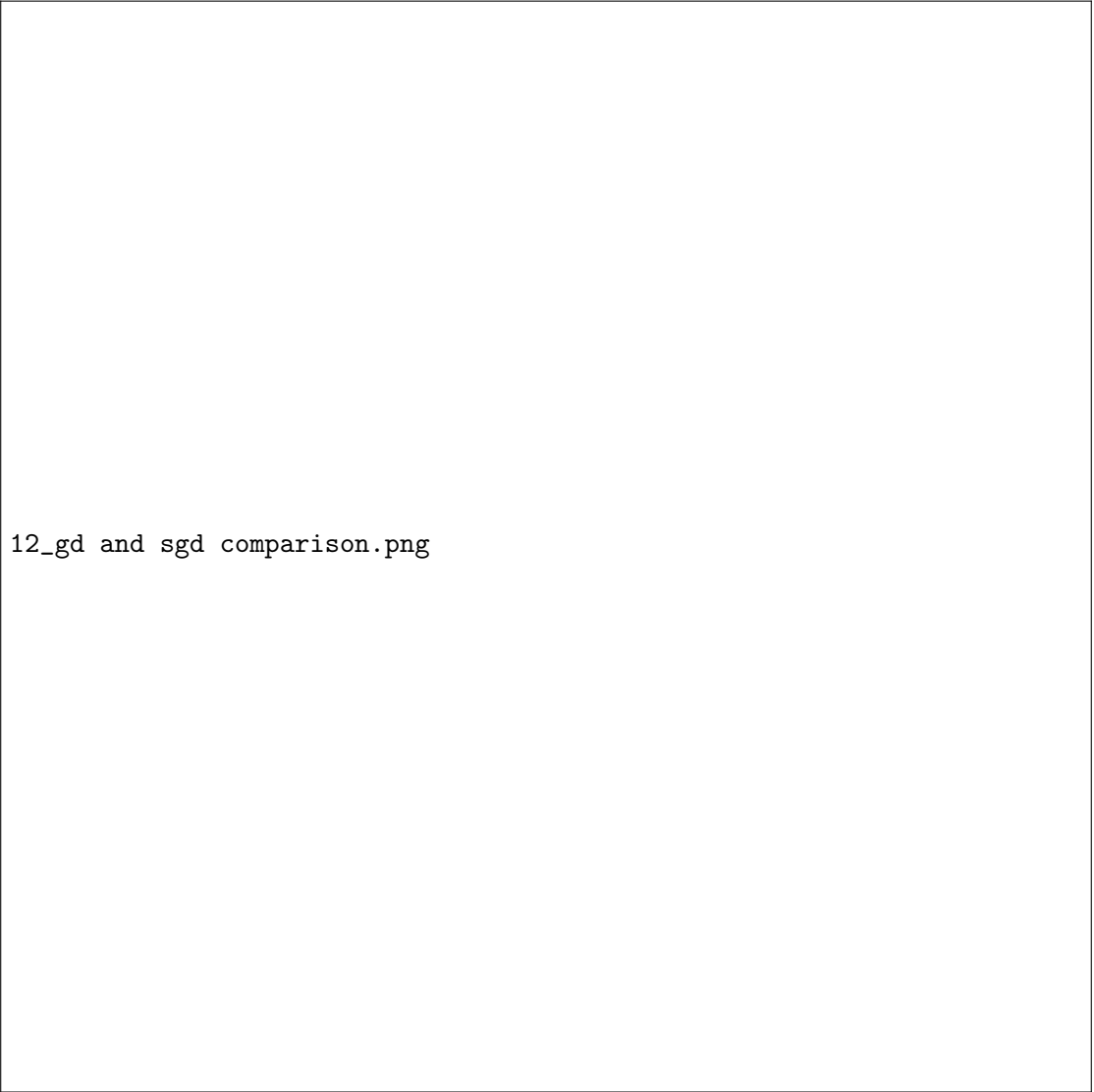


Figure 11: Stochastic Gradient Descent progress visualized in the progress of the negative log-likelihood and the learning rate over the epochs.





12\_gd and sgd comparison.png

Figure 12: Comparison of GD and SGD via the progress of the negative log-likelihood and the learning rate.

## D. Outputs and Plot for Task 3

### D.1. Confusion Matrix and Model Comparison

The confusion matrices for both Gradient Descent (GD) and Stochastic Gradient Descent (SGD) models provide insight into the classification outcomes for the test set ( $\mathbf{y}_{\text{test}}$ ) using normalized input data ( $\mathbf{X}_{\text{testz}}$ ). In comparing the results:

- **True Positives (TP):** GD identified 507 true positives, while SGD identified 508. This indicates a slight advantage for SGD in correctly classifying spam emails.

- **True Negatives (TN):** GD identified 876 true negatives, while SGD identified 878. This difference reflects that SGD is slightly more effective in identifying legitimate (non-spam) emails.
- **False Positives (FP):** GD misclassified 65 non-spam emails as spam, while SGD had 63 false positives. This suggests that GD is marginally less precise in distinguishing non-spam emails.
- **False Negatives (FN):** GD had 88 false negatives, while SGD had 87, indicating that SGD misses fewer spam emails.

The minor differences between GD and SGD are consistent with the models' stochasticity and learning processes, as SGD's stochastic updates yield slightly better performance in spam detection (higher TP, lower FN), while GD slightly reduces the misclassification of legitimate emails (higher TN, lower FP).

## D.2. Evaluation Metrics

The following evaluation metrics, calculated for the GD model, provide insight into the model's accuracy, precision, recall, and overall balance. Similar results were observed for the SGD model.

- **Accuracy:** The accuracy for GD is calculated as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{507 + 876}{507 + 876 + 65 + 88} \approx 0.9004,$$

indicating that approximately 90.04% of instances were correctly classified.

- **Precision:** The precision for the spam class in GD is

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{507}{507 + 65} \approx 0.8863,$$


meaning that around 88.63% of instances predicted as spam are indeed spam.

- **Recall:** Recall, indicating the percentage of actual spam instances correctly identified, is

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{507}{507 + 88} \approx 0.8521.$$

- **F1 Score:** The F1 score for GD is the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \approx 0.8690.$$



13\_precision\_recall\_curve.png

Figure 13: Precision-Recall curve plotted for GD and SGD.


## E. Plots and Outputs for Task 4



Figure 14: Comparison of GD with L2 regularization with GD and SGD in the progress of the negative log-likelihood.



Figure 15: Zoomed in comparison of GD with L2 regularization with GD and SGD in the progress of the negative log-likelihood.



16\_gd\_l2 learning rate.png

Figure 16: Comparison of GD with L2 regularization with GD and SGD in the progress of the learning rate.

### E.1. Precision, Recall, and F1 Score Analysis

Precision, recall, and F1 score follow similar trends to accuracy, as shown in Figure 17. At lower  $\lambda$  values, these metrics remain stable, indicating balanced performance in both positive and negative classes. As  $\lambda$  approaches values around 1 to 10, precision, recall, and F1 scores reach their highest values, reflecting an optimal regularization effect that avoids overfitting while capturing key patterns in the data. For larger  $\lambda$  values, these scores decrease markedly, emphasizing that excessive regularization reduces the model's ability to identify positive class instances effectively. Thus, moderate values of  $\lambda$  yield the best balance across these classification metrics, while extreme values degrade

performance by imposing strong constraints on parameter values.

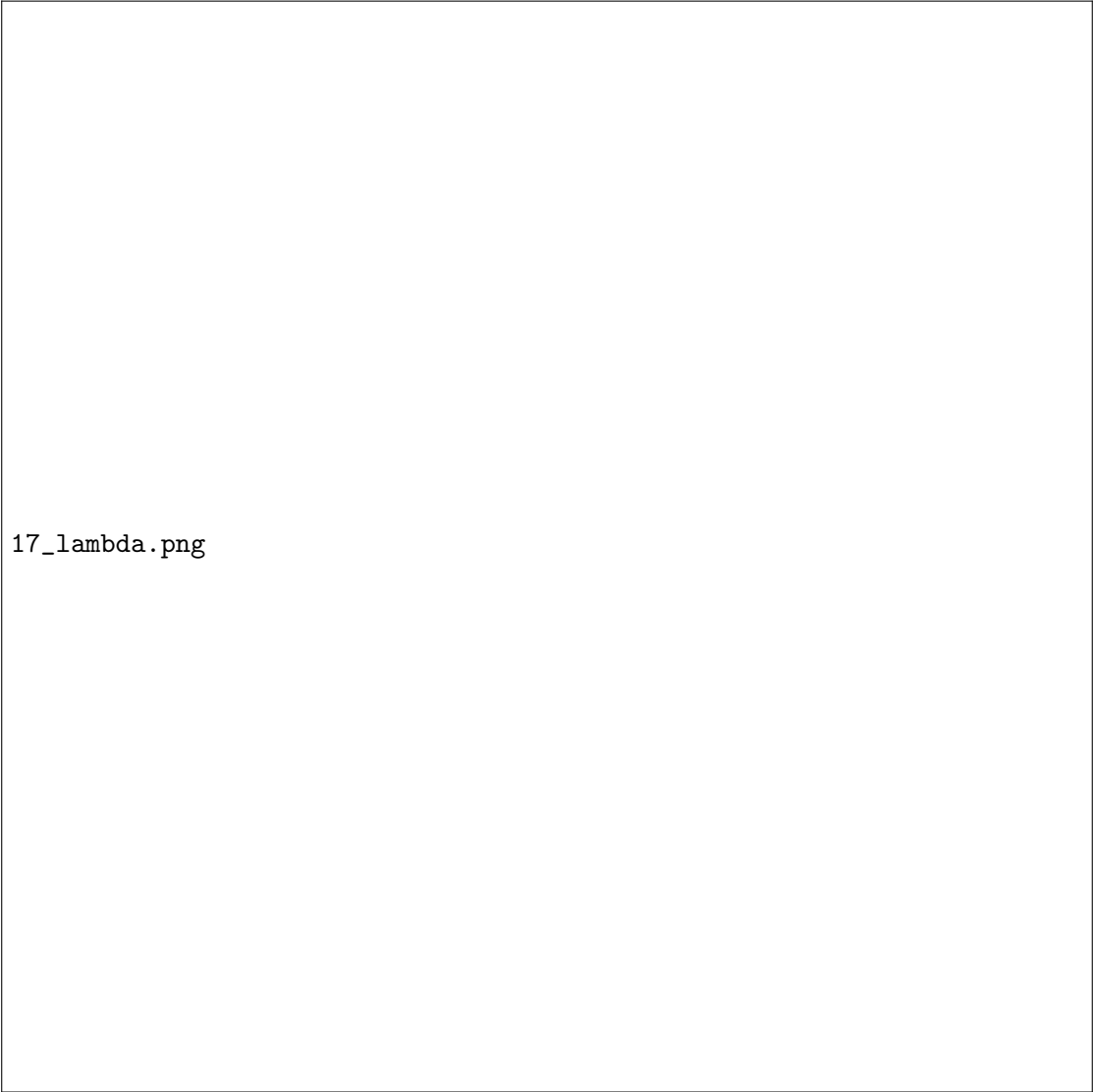
## E.2. AUC Score and $\lambda$ Sensitivity

The AUC score exhibits a smaller range of variation compared to other metrics, as shown in Figure 17. Starting from an initial value of approximately 0.958, AUC slightly increases with moderate  $\lambda$  values and reaches a peak near 0.959, indicating improved classification boundary separation. However, as  $\lambda$  grows further, AUC declines to around 0.952, reflecting a decrease in ranking quality when the model becomes overly regularized. The relatively stable AUC values across varying  $\lambda$  indicate that ranking quality remains robust even as accuracy and other metrics fluctuate more significantly.

## E.3. Summary of Findings

The analysis demonstrates that  $\lambda$  plays a crucial role in determining the model’s performance. A balanced  $\lambda$  value, in the range of 1 to 10, enables the model to achieve high classification accuracy and effective generalization without excessive overfitting. At very high values, the L2 penalty restricts the model’s flexibility, leading to significant underfitting and a drop in all performance metrics except for a comparatively less affected AUC score.

In conclusion, the results underscore that an optimal  $\lambda$  balances error minimization and generalization. While low  $\lambda$  values permit better fit on the training data, they risk overfitting, as indicated by disparities between training and test log-likelihoods. Moderate  $\lambda$  values effectively reduce overfitting while retaining the model’s flexibility, as evidenced by improvements in log-likelihoods, accuracy, and F1 score. Excessive  $\lambda$  values, however, impose excessive constraints, limiting the model’s ability to capture essential data patterns. These findings, illustrated in Figures 14 through 17, confirm that the choice of  $\lambda$  is essential to achieving a model that generalizes well without compromising predictive accuracy.



17\_lambda.png

Figure 17: Effect of different Lambda parameter size.



## Declaration of Honor

I hereby declare that I have written the enclosed project report without the help of third parties and without the use of other sources and aids other than those listed in the table below and that I have identified the passages taken from the sources used verbatim or in terms of content as such. or content taken from the sources used. This work has not been submitted in the same or a similar form been submitted to any examination authority. I am aware that a false declaration declaration will have legal consequences.

### Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
DeepL	Translation	Throughout	+
Github Copilot	Code generation	a02-lr.ipynb	+

Signatures

Mannheim, 01. November 2024