Assignment Report

Assignment 2: Logistic Regression

Artem Bisliouk (abisliou), Elizaveta Nosova (enosova)

November 2, 2024

1. Dataset Statistics

a) Kernel density plot

Due to the large overlap of the density functions, it is challenging to distinguish between individual feature distributions shown in the left plot in Figure 1. However, some features are highly skewed with heavy tails, indicating sparse data and the presence of outliers. Many values are clustered near zero, with a few high values stretching the distributions, reflecting the low frequency of certain words or characters in emails.

Based on the statistics of feature distributions in Figure 7 in Appendix A, features with high skewness, kurtosis, and maximum values, such as *capital_run_length_total*, *capital_run_length_longest*, and *capital_run_length_average*, show strong asymmetry and heavy tails in the kernel density plot, significantly affecting the overall distributions shape. These features have extreme maximum values, leading to sparse distributions with outliers extending beyond the main data cluster.

b) Data normalization: Implemented in the Jupiter notebook. For normalization of test data, statistics of train data were used.

c) Kernel density plot of normalized data

In the kernel density plot of the normalized data on the right plot of Figure 1, the distributions for all features have been scaled to a comparable range, centered around 0 and mostly constrained within a few standard deviations. This transformation reduces the extreme values and long tails observed in the unnormalized data, making the densities more compact and similarly spread.

However, some features still exhibit minor skewness, and the same features as before normalization have subtle tails extending beyond the main cluster, suggesting the presence of residual outliers even after normalization.

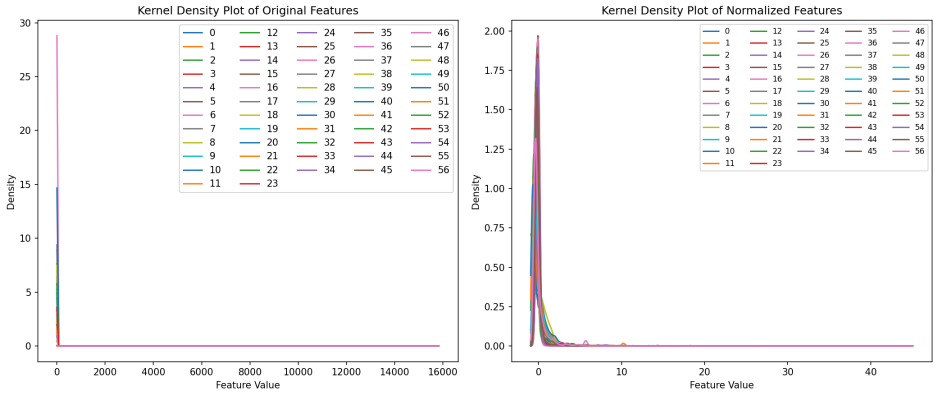

Figure 1: Kernel density plot of original and normalized features

2. Maximum Likelihood Estimation

a) Invariance of Maximum Likelihood Estimates Under Rescaling and Shifting with Bias Term

1. The likelihood function for logistic regression over a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}$ is:

$$\mathcal{L}(\mathbf{w}, b) = \prod_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)})^{y^{(i)}} (1 - P(y^{(i)}|\mathbf{x}^{(i)}))^{1-y^{(i)}}$$

where the logistic regression model is defined as:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+w_0)}}$$

where σ is the sigmoid function, $\mathbf{w}$ is the weight vector, and w_0 is the bias term.

2. Now consider a transformation of the features, where each feature x_j is rescaled by a constant a_j and shifted by a constant c_j. Define the transformed features as:

$$x'_j = a_j x_j + c_j \quad \text{for each } j.$$

Thus, the transformed feature vector $\mathbf{x}'$ can be written as:

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

where $\mathbf{A}$ is a diagonal matrix with a_j on the diagonal, and $\mathbf{c}$ is a vector of constants c_j.

3. Using the transformed feature vector, the model becomes:

$$P(y = 1|\mathbf{x}') = \sigma(\mathbf{w}'^T\mathbf{x}' + w'_0)$$

where $\mathbf{w}'$ and w'_0 are the new weight and bias terms to be determined.

Let's substitute $\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{c}$ into the model:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}'^T(\mathbf{A}\mathbf{x} + \mathbf{c}) + w_0')$$

$$= \sigma((\mathbf{w}'^T\mathbf{A})\mathbf{x} + (\mathbf{w}'^T\mathbf{c} + w_0'))$$

To make this equivalent to the original model $\sigma(\mathbf{w}^T\mathbf{x} + w_0)$, set:

$$\mathbf{w}'^T\mathbf{A} = \mathbf{w}^T \quad \text{and} \quad \mathbf{w}'^T\mathbf{c} + w_0' = w_0$$

Solving these equations:

$$\mathbf{w}' = \mathbf{w} \odot \mathbf{A}^{-1} \quad \text{and} \quad w_0' = w_0 - \mathbf{w}'^T\mathbf{c}$$

where $\odot$ denotes element-wise division.

Thus we have shown that having the scaled and shifted features $\mathbf{x}'$ we can find such a weight vector $\mathbf{w}'$ and a bias term w_0' that will give us the same likelihoods.

Since $\mathbf{w}'^T\mathbf{x}'^{(i)} + w_0' = \mathbf{w}^T\mathbf{x}^{(i)} + w_0$, it follows that:

$$\sigma(\mathbf{w}'^T\mathbf{x}'^{(i)} + w_0') = \sigma(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)$$

$$P(y = 1|\mathbf{x}') = P(y = 1|\mathbf{x})$$

Thus, $P(y = 1|\mathbf{x})$ remains unchanged under the transformation, meaning that the likelihood $\mathcal{L}(\mathbf{w}', w_0')$ is identical to the original likelihood $\mathcal{L}(\mathbf{w}, w_0)$, showing that the MLE remains the same with the transformed parameters $\mathbf{w}'$ and w_0'. This is also a consequence of the property of exponential family functions.

Computing z-scores helps standardize each feature, bringing them to a similar scale. This prevents any one feature from dominating the model due to scale differences and improves the convergence of optimization algorithms like gradient descent by ensuring numerical stability and thus mitigating the problems of gradient explosion and vanishing. Taking scale into account is especially crucial for MAP estimation, as it influences the prior density and ensures consistent penalties when using l2 regularization. Thus, standardizing features helps the model train more efficiently, yielding better results.

b) Log-likelihood and gradient of the log-likelihood

c) Gradient Descent (GD)

d) Stochastic Gradient Descent (SGD) Tasks 2b-d are implemented in the Jupiter notebook.

e) Comparison of GD and SGD on given parameters

In Figure 2, we analyze the performance of Gradient Descent (GD) and Stochastic Gradient Descent (SGD) in optimizing the negative log-likelihood for logistic regression. The left plot illustrates that both algorithms effectively reduce the negative log-likelihood on the training data within the first 30 epochs, with SGD showing quicker initial improvements already by the 10th epoch. After the 50th epoch, changes in the objective

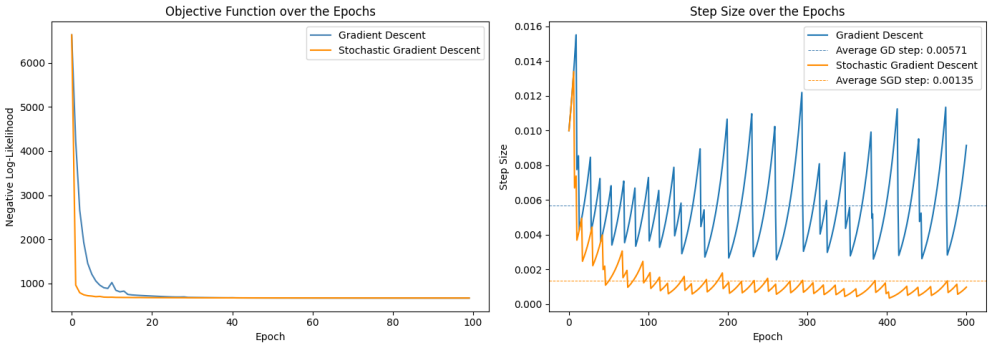

Figure 2: GD and SGD on Objective Function and Step Size Convergence

function are minimal, occurring only at the floating-point level in each epoch. Instead of running the algorithms for the preset 500 epochs, early stopping techniques could be employed to prevent overfitting.

The right plot depicts the learning rate fluctuations over the epochs. Both GD and SGD start with a default learning rate of 0.1, which peaks shortly after the initial iterations and subsequently decreases. On average, GD maintains a higher learning rate (0.00571) compared to SGD (0.00135), with less frequent fluctuations. More frequent variability in SGD, potentially influenced by the bold driver heuristic, suggests that increases in the objective function after updates occur more often. These oscillations in SGD highlight its adaptability, which can be beneficial in navigating the parameter space, especially in scenarios involving larger datasets where the computational cost of full-batch updates becomes prohibitive.

3. Prediction

Both classifiers, trained using GD and SGD for weight optimization respectively, were applied to the test split and achieved nearly identical performance metrics: an accuracy of 0.92, recall of 0.94 on the negative class, and 0.88 on the positive class. The only performance difference lies in precision, where the GD model achieved 0.93 for the negative class and 0.91 for the positive class, with the SGD model scoring 0.01 lower on each, a minor and likely negligible difference. The confusion matrices in Figure 8 in Appendix B confirm that the models' predictions differ only by 1 false positive and 1 false negative for the SGD model. This suggests that both GD and SGD are comparably effective and suitable optimization methods for given data and parameters, with similar generalization to unseen data.

From the composition of the feature vector in Figure 3 we observe that both models agree that the most significant feature by far is *word_freq_3d*. Notably, this feature shows the largest difference in weight between the two models: in the GD model, its weight is 8.086, while in SGD it is less than half, at 3.957. The feature *word_freq_650* has the lowest weight, close to 0: -0.0014 in GD and 0.0110 in SGD, indicating that its

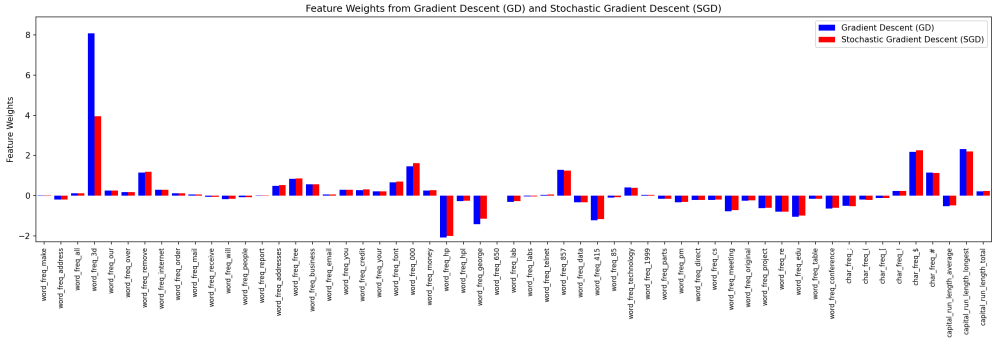

Figure 3: Weights attributed to the features by GD and SGD

contribution to the classification decision is minimal. For the remaining features, the weight differences between models are minor, with absolute values ranging between 0 and 2.

Table 1 in Appendix C presents the ten most important features based on the absolute values of the weights for both GD and SGD. While both models include the same features, their rankings may differ; however, the discrepancies in the values assigned by GD and SGD are minimal, existing only at the level of floating points, with exception of top feature *word_freq_3d* discussed in previous paragraph. This suggests a general agreement between the two classifiers.

Although the occurrence of the pattern "3d" may not seem like an obvious indicator of spam emails, some other features contributing to spam classification are more intuitive. For instance, emails containing words like "meeting" and "conference" or proper nouns such as "george" and "hp" relatively frequently are less likely to be spam, e.g. "Hi team, George has scheduled a meeting with the HP representative we met at the tech conference". At the same time, those with extended capitalized phrases, symbols like "$" and "#," and words such as "free" are more likely to be spam, e.g. "CONGRATULATIONS! You've been selected to WIN $1000 CASH! #CLICK HERE# TO CLAIM YOUR FREE PRIZE NOW!". However, the significance of certain numbers in the text, such as the positive weight for 867 and negative weight for 415, is less interpretable. This discrepancy reflects the challenge in feature interpretability for numerical tokens, which may relate more to dataset-specific patterns than to semantic meaning.

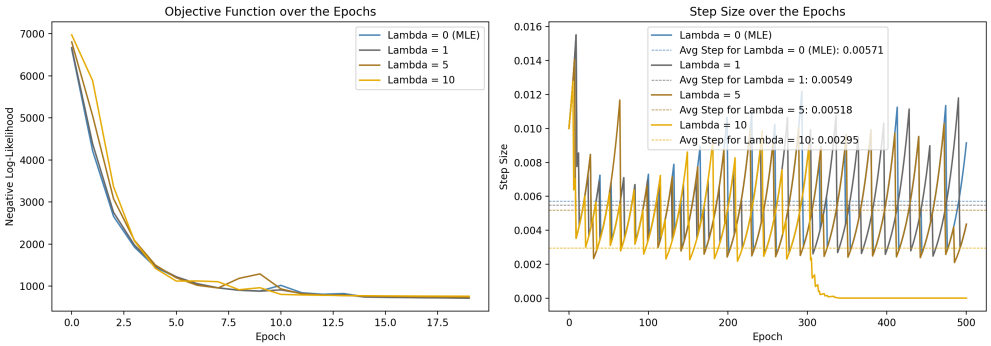

Figure 4: Objective Function and Step Size over Epochs for Small λ Values (0, 1, 5, 10)

4. Maximum Aposteriori Estimation

a) Gradient descent for MAP estimation: Implemented in the Jupiter notebook.

b) Effect of the prior

Small λ values: As shown in Figure 4, Maximum A Posteriori (MAP) estimation converges toward Maximum Likelihood Estimation (MLE) as $\lambda \to 0$, given that $\lambda = \frac{1}{\sigma^2}$ of the Gaussian prior. When $\lambda = 0$, the prior distribution is uniform, effectively making MAP estimation identical to MLE. This explains the similarity in behavior for GD with MLE at $\lambda = 0$. For $\lambda = 1$, the model penalizes large weights moderately, achieving a balance that encourages convergence and generalization, as seen in the smooth and stable objective function plot around epoch 10.

The plot on the right illustrates the learning rate change under the bold driver heuristics. The step size behaves similarly for λ equal to 0, 1 and 5, oscillating around the average of 0.005. A much lower average for $\lambda = 10$ does not mean a generally smaller learning rate size, but is caused an abrupt drop at epoch 300. With larger λ, the penalty from the regularization term gains strength, bounding the weights update by prior's decreasing variance. The updates become minimal and further adjustments have little to no effect on the model's performance.

High λ values: For high λ values (e.g., 50, 100, and 200), as shown in Figure 5, the initial objective values increase dramatically, exceeding 60,000 for $\lambda = 200$. This large initial penalty reflects the strong influence of the prior as $\lambda = \frac{1}{\sigma^2}$, which drives the weight vector w towards zero, acting as a strong regularizer to prevent overfitting. As λ grows, the penalty term $\frac{\lambda}{2}\|w\|^2$ dominates, constraining weight updates and slowing convergence. For instance, $\lambda = 50$ stabilizes within approximately 10 epochs, whereas higher λ values, such as 100 and 200, require 15 epochs to reach near-minimal objective values. On the right plot the same drop in learning rate as for $\lambda = 10$ occurs, happening earlier with increasing λ. High λ values illustrate how restrictive priors slow learning, prioritizing generalization over rapid adaptation to the data.

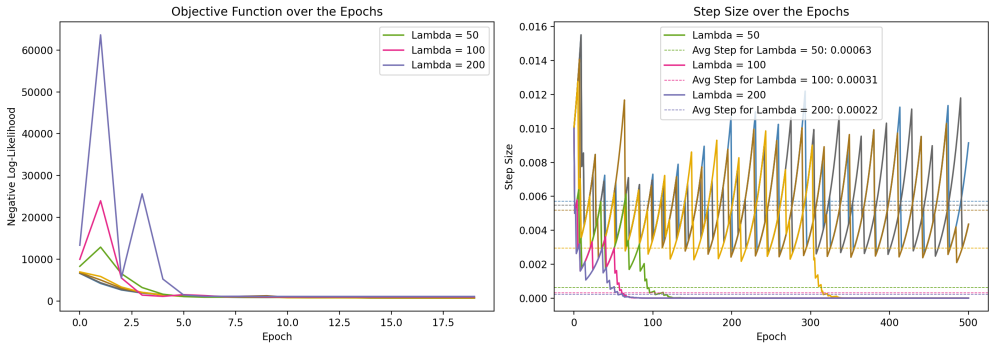

Figure 5: Objective Function and Step Size over Epochs for High λ Values (50, 100, 200)

Extreme λ values: For extreme λ values, particularly at $\lambda = 500$ and $\lambda = 1000$, the effects of the prior become overwhelmingly dominant, as shown in Figure 9 in Appendix D. With $\lambda = 500$, the objective function initially spikes above 1.2×10^6 due to the heavy penalty term $\frac{\lambda}{2}\|w\|^2$, but rapidly reduces within five epochs as the regularization constrains weight magnitudes. In contrast, for $\lambda = 1000$, we observe no curve for the objective function as gradient explosion occurs. This is evident from the permanently increasing step size, starting at 0.01 but escalating to over 3×10^8 by epoch 500. This rapid escalation renders the model unable to achieve numerical stability, resulting in the log-likelihood becoming NaN throughout the training process.

As $\lambda \to \infty$, the variance of the Gaussian prior $\sigma^2 = \frac{1}{\lambda}$ approaches zero, effectively enforcing a near-zero constraint on weights. This extreme regularization, as observed for $\lambda = 1000$, prevents the model from adapting to data, leading to numerical instability and failure to learn meaningful patterns. By comparison, while $\lambda = 500$ allows convergence, the high regularization still restricts weight flexibility, resulting in a constrained model. Thus, extreme λ values reveal the limitations of MAP estimation under strong priors, where excessive regularization causes gradient instability or overly suppresses model responsiveness.

Model performance with different λ values: The model's performance summary (Figure 6) reflects the influence of λ on the log-likelihood and accuracy metrics. As λ increases, both the training (Train LL) and test log-likelihoods (Test LL) progressively decrease, indicating the stronger regularizing effect imposed by MAP estimation. This decrease is due to the penalty term, $\frac{\lambda}{2}\|w\|^2$, which increasingly restricts the magnitude of the weights as λ grows, thereby reducing the model's capacity to fit both the training and test data closely. However, Test LL remains consistently higher than Train LL across all values of λ, which is expected in MAP estimation with regularization. The regularization helps prevent overfitting to the training data by constraining model complexity, which leads to a better generalization on the test data.

For lower λ values (e.g., 0 and 1), the model achieves high test accuracy, reaching a maximum of 0.92, as it benefits from a balance between data fit and minimal regular-

ization. However, as λ increases beyond 100, test accuracy begins to decline, notably dropping to 0.88 at $\lambda = 500$ and plummeting to 0.61 for $\lambda = 1000$, where we also observe NaN values in the log-likelihood due to numerical instability. While the accuracy score for $\lambda = 1000$ still suggests performance above a random guess, the F1-score—reflecting the harmonic mean of precision and recall—reveals severe underfitting, as the model predicts only the 0 (Non-spam) class, resulting in an F1-score of 0. This behavior aligns with the analysis of extreme λ values, where the heavy regularization limits model adaptability, resulting in underfitting and the loss of predictive capability. The results confirm that while moderate regularization (λ around 1 to 10) helps in reducing overfitting, excessively large λ values overly constrain the weight vector, hindering the model's ability to capture meaningful data patterns effectively.

	Lambda	Train LL	Test LL	Test Accuracy	Test F1-Score
0	0.0	-655.41	-427.34	0.92	0.89
1	1.0	-682.85	-447.83	0.92	0.89
2	5.0	-722.37	-473.22	0.92	0.89
3	10.0	-754.85	-492.75	0.92	0.9
4	50.0	-893.11	-575.72	0.92	0.89
5	100.0	-988.51	-632.24	0.92	0.89
6	200.0	-1108.95	-701.37	0.91	0.89
7	500.0	-1304.38	-805.99	0.91	0.88
8	1000.0	nan	nan	0.61	0.0

Figure 6: Model Performance Summary with Different λ Values.

c) Composition of the weight vector

The weight composition across varying λ values, as shown in Figure 10 in Appendix E, reveals how regularization strength directly impacts feature importance. For small λ values (e.g., 0 and 1), the weights exhibit a broad range, allowing certain features to have relatively large positive or negative values, which indicates that the model places significant importance on these features to maximize predictive accuracy. As λ increases, the weight magnitudes decrease uniformly, reflecting the stronger Gaussian prior (lower variance) that suppresses individual feature importance by shrinking all weights towards zero. This effect aligns with the penalty term $\frac{\lambda}{2}\|w\|^2$, which penalizes large weights to enforce simpler models. Notably, for very high λ values like 500 and 1000, almost all weights converge close to zero (for λ=1000 it's even not observable), indicating an over-regularized model that heavily restricts feature influence, resulting in an underfitted model with diminished predictive capacity. This transition demonstrates how MAP estimation balances model complexity and generalization, with high λ values imposing strong priors that minimize variance but also reduce the model's adaptability to the data, consistent with the trends in convergence behavior and objective function observed in previous analyses.

Appendix

A. Feature Distribution

Figure 7 shows statistics of original features.

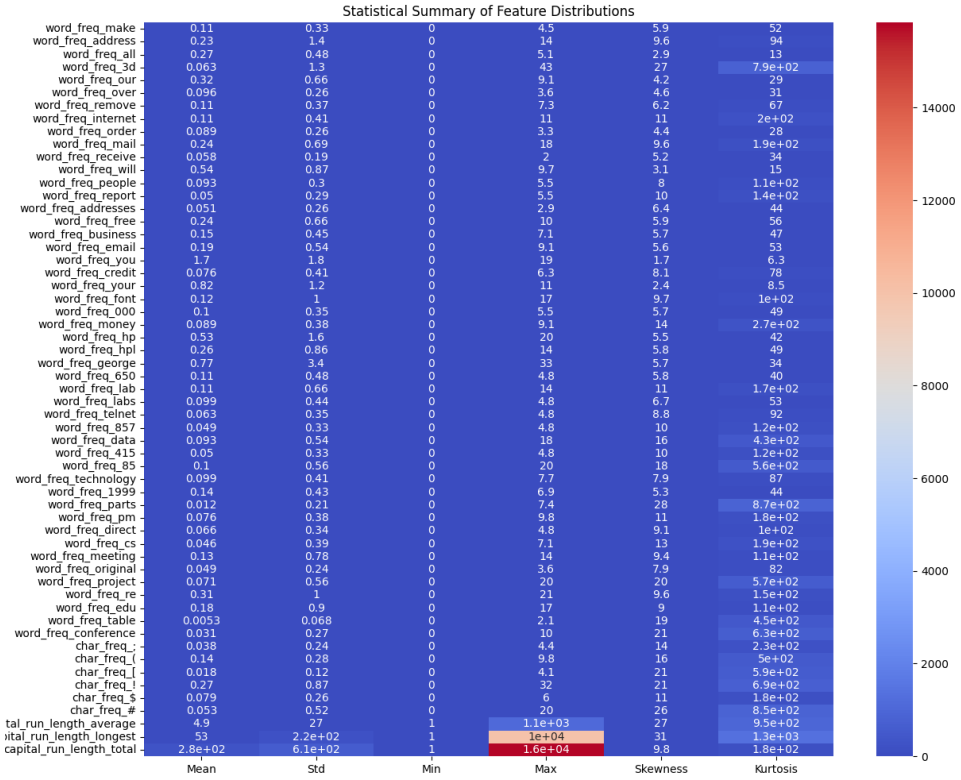

Figure 7: Statistical Summary of Feature Distributions

B. Prediction: Confusion Matrix

The confusion matrix resulting after classifying the test set with GD and SGD models in Figure 8 shows nearly identical performance.

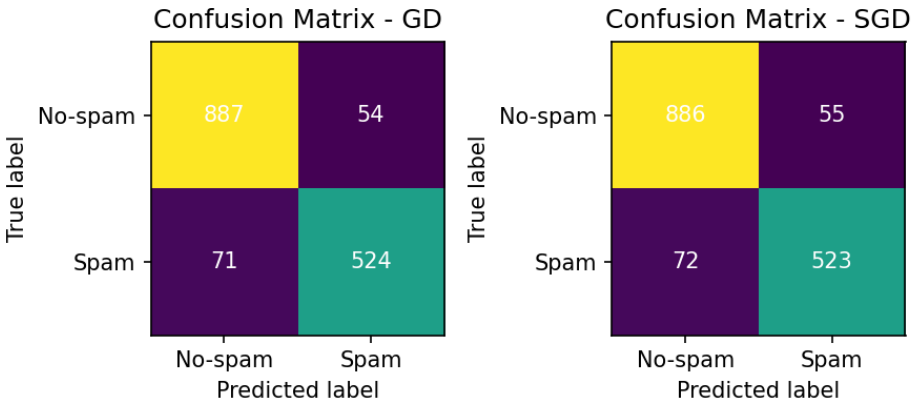

Figure 8: Confusion matrix of GD and SGD on test data

C. Weight Vector Composition: Largest weights for GD and SGD

The Table 1 shows 10 features with largest weights for GD and SGD. 10 features are the same for both models, but the ranking varies in some cases. Features that are assigned different ranks in GD and SGD are marked in italics in the corresponding columns of the table.

D. Effects of Extreme Values of λ

Negative log-likelihood and step size over the epochs for the values of λ equal to 500 and 1000 are plotted in Figure 9

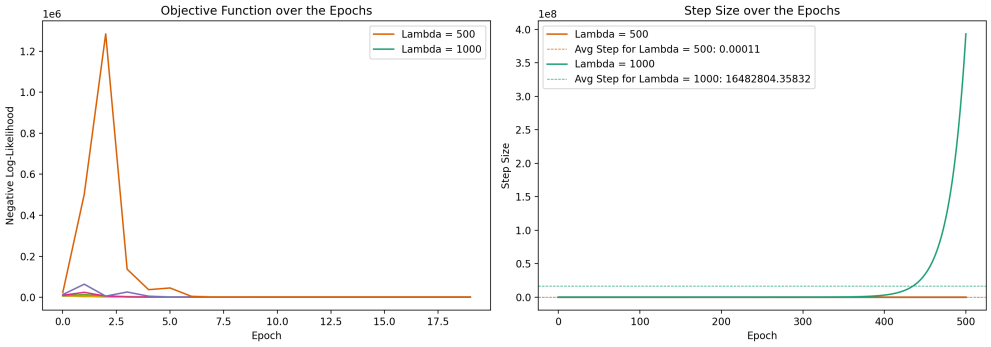

Figure 9: Objective Function and Step Size over Epochs for Extreme λ Values (500, 1000)

Rank	GD		SGD	
	Feature	Weight	Feature	Weight
1	word_freq_3d	8.08626	word_freq_3d	3.95717
2	*capital_run_length_longest*	2.31366	*char_freq_$*	2.25393
3	*char_freq_$*	2.18927	*capital_run_length_longest*	2.20270
4	word_freq_hp	-2.07768	word_freq_hp	-2.00621
5	word_freq_000	1.46175	word_freq_000	1.60874
6	*word_freq_george*	-1.41615	*word_freq_857*	1.23966
7	*word_freq_857*	1.28950	*word_freq_remove*	1.19136
8	word_freq_415	-1.22487	word_freq_415	-1.15847
9	*char_freq_#*	1.15454	*word_freq_george*	-1.14657
10	*word_freq_remove*	1.15334	*char_freq_#*	1.12545

Table 1: Top 10 features by importance (absolute weight) for GD and SGD

E. Composition of weight vector with different values of λ

Figure 10 allows to compare weights assigned to features with varying values of lambda

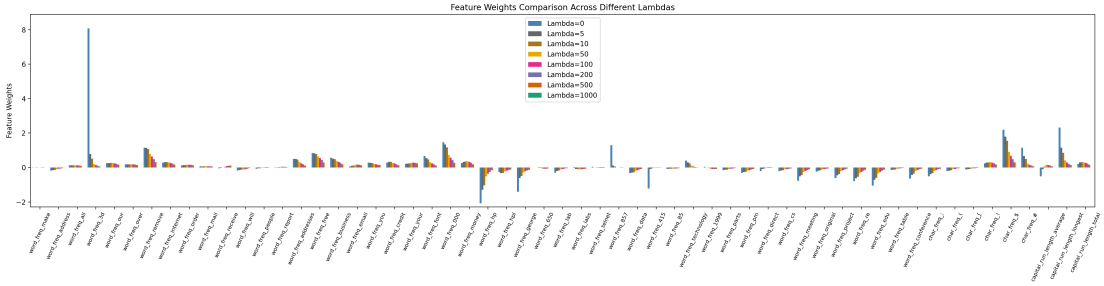

Figure 10: Feature Weight Comparison Across Different λ Values

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
DeepL	Style Edits	Throughout	++
GPT-4	Code debugging	Throughout	+-

Unterschrift 

Mannheim, den 2. November 2024