Department of Electronic & Telecommunication Engineering
University of Moratuwa, Sri Lanka.

EN3150 - Pattern Recognition

Assignment 02: Learning from data and related challenges and classification

210391J - Morawakgoda M.K.I.G.

29th September 2024

Contents

1 Logistic regression

2. The purpose of `y_encoded = le.fit_transform(df_filtered['species'])` is to encode the categorical variable `"species"` into numerical labels. Since machine learning models like logistic regression require numerical inputs, this step converts the `"Adelie"` and `"Chinstrap"` species into 0 and 1 (or other integers). The `fit_transform()` method fits the LabelEncoder to the `"species"` column and transforms the species names into encoded integers in a single step.

3. The purpose of `X = df_filtered.drop(['species', 'island', 'sex', 'class_encoded'], axis=1)` is to select the feature columns for the model, excluding those that aren't relevant for logistic regression. Specifically, `"species"` and `"class_encoded"` are dropped because they represent the target variable, and `"island"` and `"sex"` are dropped because they are categorical features that haven't been encoded.

4. We cannot use the `"island"` and `"sex"` features because they are categorical variables and have not been converted into numerical representations (e.g., using one-hot encoding or label encoding). Logistic regression requires all input features to be numerical, so the categorical `"island"` and `"sex"` columns need preprocessing before they can be included in the model.

6. The purpose of using `random_state=42` is to ensure the data is split into training and testing sets in a consistent, reproducible way. By specifying `random_state=42`, the same random sequence is used every time the code is run, leading to the same train-test split. This is important for debugging and comparison of results, as it guarantees that any changes in model performance are due to changes in the model, not due to different train-test splits.

7. - **Reason for low accuracy :** Excluding important categorical features like `"island"` and `"sex"` without encoding them results in a loss of valuable information. Also the limited amount of training data can prevent the model from capturing enough variance, reducing its ability to generalize well to unseen data.
   - **Reason for poor performance of the `saga` model :** The `saga` solver is designed for large datasets and regularization problems, making it less effective for small datasets like this one.

8. **Classification accuracy :** `1.0`

9. - `liblinear` typically converges faster on small datasets, providing better results without the need for excessive computational resources. `saga` is optimized for scalability and may struggle with small datasets, resulting in slower convergence and less efficient optimization.
   - For small datasets, `liblinear` can efficiently handle L2 regularization and binary classification, whereas `saga` require additional tuning of regularization parameters to achieve good performance.

10. - The follwoing table compares the performance of the `liblinear` and `saga` solvers when used with and without feature scaling. The accuracy values demonstrate that feature scaling has a significant impact on the `saga` solver's performance, whereas the `liblinear` solver shows only a small difference.

Solver	Without Feature Scaling	With Feature Scaling
liblinear	1.0	0.9767441860465116
saga	0.5813953488372093	0.9767441860465116

Table 1: Accuracy Comparison of liblinear and saga Solvers with and without Feature Scaling

- When feature scaling is applied, both solvers achieve nearly identical accuracy values of around 0.98. This indicates that feature scaling normalizes the feature values, ensuring that all features contribute equally to the model. For the `saga` solver, scaling stabilizes the optimization process and helps it converge faster to a more accurate solution.

11.
- **Problem :** The `X` matrix still includes string-based features and the logistic regression algorithm cannot process strings directly. These categorical features need to be encoded into numerical form before training the model.
- **Solution :** Above problem can be solved by converting the categorical features into numerical values using one-hot encoding. The `pd.get_dummies()` function from `pandas` is useful for this.

Listing 1: Corrected code

```
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the penguins dataset
df = sns.load_dataset("penguins")
df.dropna(inplace=True)

# Filter rows for 'Adelie' and 'Chinstrap' classes
selected_classes = ['Adelie', 'Chinstrap']
df_filtered = df[df['species'].isin(selected_classes)].copy()  # Make a
    copy to avoid the warning

# Initialize the LabelEncoder for the target variable
le = LabelEncoder()
y_encoded = le.fit_transform(df_filtered['species'])
df_filtered['class_encoded'] = y_encoded

# Drop the target variable columns from features (X)
X = df_filtered.drop(['species', 'class_encoded'], axis=1)

# One-hot encode the categorical features (island, sex, etc.)
X_encoded = pd.get_dummies(X, drop_first=True)  # drop_first avoids
    multicollinearity

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded,
    test_size=0.2, random_state=42)

# Train the logistic regression model
logreg = LogisticRegression(solver='saga', max_iter=5000)
logreg.fit(X_train, y_train)

# Predict on the testing data
y_pred = logreg.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Coefficients:", logreg.coef_)
print("Intercept:", logreg.intercept_)
```

12.
- **Issues with Label Encoding and Scaling :** This approach is not correct because using label encoding on a categorical feature can create issues when applying feature scaling methods like Standard Scaling or Min-Max Scaling.
  - Label encoding assigns integer values to each category (e.g., red = 0, blue = 1, green = 2). This implies an ordinal relationship among the categories, suggesting that green is greater

than blue, which is not true for nominal categories. This can lead the model to infer incorrect relationships and affect the training process.

- After label encoding, numerical values are treated as continuous data. When feature scaling methods are applied, the relative distances between the encoded values may not represent meaningful differences among the categories. For example, scaling the values [0, 1, 2] will yield values that imply an artificial distance between the categories.

- **Proposed approach :** To address the issues associated with label encoding, we can use one-hot encoding for nominal categorical features. One-hot encoding creates binary columns for each category, eliminating misleading ordinal relationships. After one-hot encoding, applying feature scaling methods is appropriate, as the binary values can be scaled without distortion, thereby preserving the integrity of the data.

red	blue	green
1	0	0
0	1	0
0	0	1

- **Question 2**

1. - Calculate the logit:

$$\begin{aligned}\text{logit}(P(y=1)) &= w_0 + w_1x_1 + w_2x_2 \\ &= -5.9 + 0.06(50) + 1.5(3.6) \\ &= -5.9 + 3 + 5.4 \\ &= 2.5\end{aligned}$$

   - Convert logit to probability:

$$\begin{aligned}P(y=1) &= \frac{e^{\text{logit}(P(y=1))}}{1 + e^{\text{logit}(P(y=1))}} \\ &= \frac{e^{2.5}}{1 + e^{2.5}} \\ &= \frac{12.1825}{1 + 12.1825} \\ &\approx \frac{12.1825}{13.1825} \\ &\approx 0.923\end{aligned}$$

   - The estimated probability that the student will receive an A+ is approximately **0.923** or **92.3**%.

2. - Set the probability:

$$P(y=1) = 0.6$$

   - Calculate logit:

$$\begin{aligned}\text{logit}(P(y=1)) &= \ln\left(\frac{0.6}{1-0.6}\right) \\ &= \ln(1.5) \approx 0.4055\end{aligned}$$

– Set up the equation:

$$\begin{aligned} 0.4055 &= -5.9 + 0.06x_1 + 1.5(3.6) \\ 0.4055 &= -5.9 + 0.06x_1 + 5.4 \\ 0.4055 &= -0.5 + 0.06x_1 \\ 0.9055 &= 0.06x_1 \\ x_1 &= \frac{0.9055}{0.06} \approx 15.09 \end{aligned}$$

– To achieve a 60% chance of receiving an A+, the student needs to study approximately **15.09** hours.

2 Logistic regression on real world data

2. • The correlation matrix shows weak to moderate positive relationships between most word frequencies, indicating that these features are largely independent. The highest correlations with the target variable are seen in `word_freq_free` (0.263), `word_freq_order` (0.232), and `word_freq_receive` (0.235).

	word_freq_internet	word_freq_order	word_freq_credit	word_freq_free	word_freq_receive	target
word_freq_internet	1.000000	0.105302	0.109163	0.051115	0.128495	0.206808
word_freq_order	0.105302	1.000000	0.123217	0.008269	0.137760	0.231551
word_freq_credit	0.109163	0.123217	1.000000	0.027665	0.155769	0.189761
word_freq_free	0.051115	0.008269	0.027665	1.000000	0.098711	0.263215
word_freq_receive	0.128495	0.137760	0.155769	0.098711	1.000000	0.234529
target	0.206808	0.231551	0.189761	0.263215	0.234529	1.000000

Table 2: Correlation matrix of selected word frequencies and target variable

• All features are skewed toward low values and the pair plot indicates that all features can effectively differentiate between target classes, with higher frequencies linked to target 1.

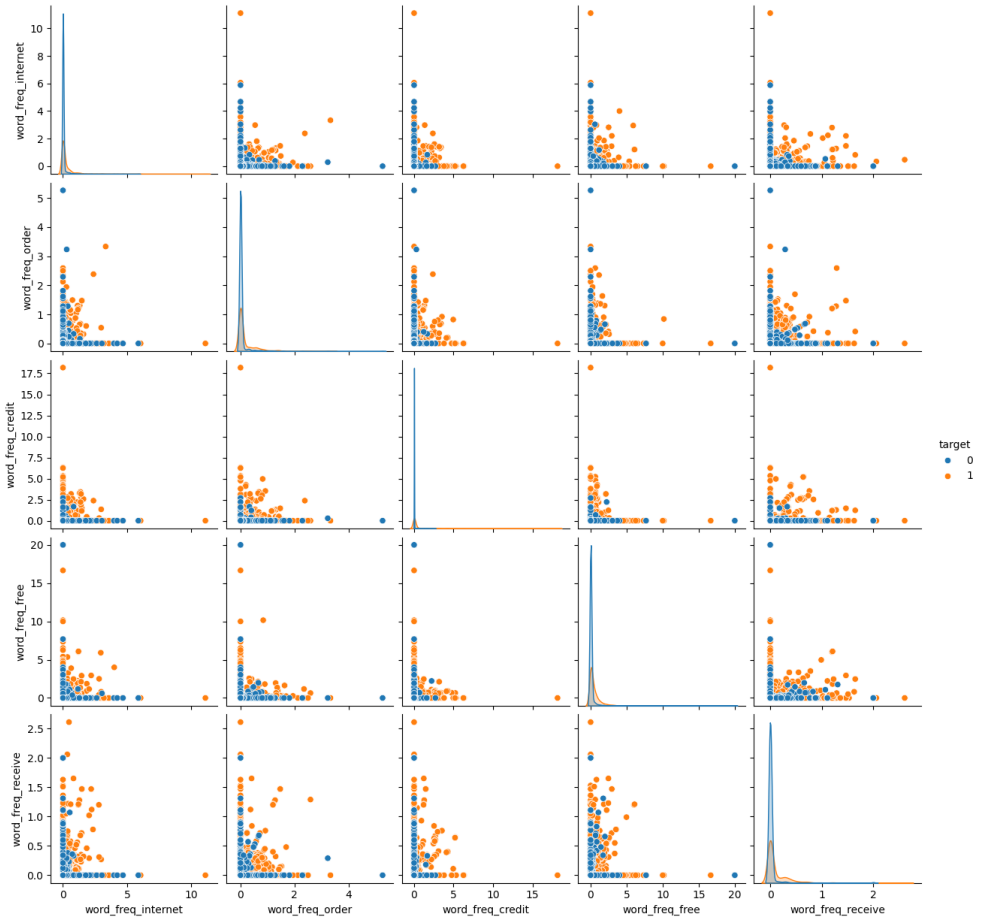
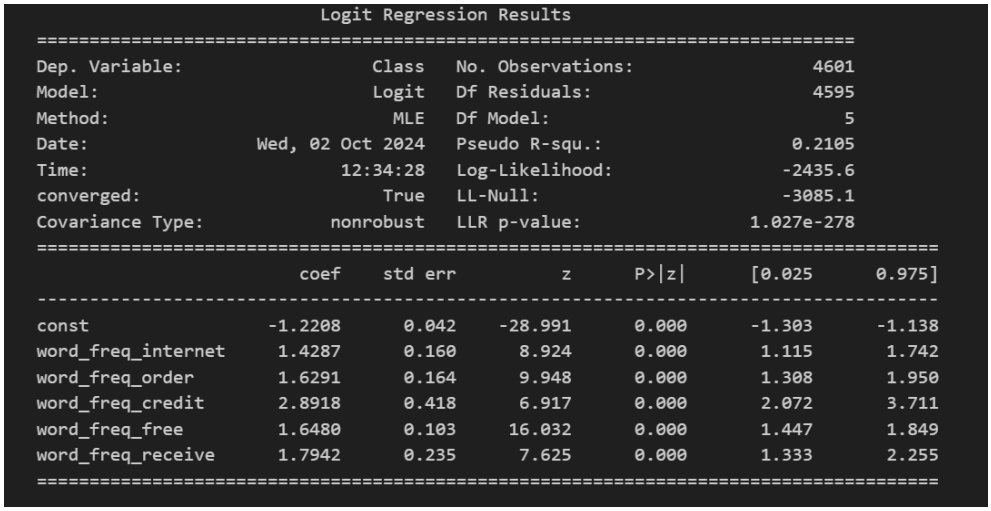


Figure 1: Pair Plots

3. **Accuracy** = 0.75

4. All p-values were found to be almost zero, indicating a strong statistical significance for each feature in predicting the dependent variable. Hence any feature can not be discarded.



Logit Regression Results

Dep. Variable:	Class	No. Observations:	4601
Model:	Logit	Df Residuals:	4595
Method:	MLE	Df Model:	5
Date:	Wed, 02 Oct 2024	Pseudo R-squ.:	0.2105
Time:	12:34:28	Log-Likelihood:	-2435.6
converged:	True	LL-Null:	-3085.1
Covariance Type:	nonrobust	LLR p-value:	1.027e-278

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.2208	0.042	-28.991	0.000	-1.303	-1.138
word_freq_internet	1.4287	0.160	8.924	0.000	1.115	1.742
word_freq_order	1.6291	0.164	9.948	0.000	1.308	1.950
word_freq_credit	2.8918	0.418	6.917	0.000	2.072	3.711
word_freq_free	1.6480	0.103	16.032	0.000	1.447	1.849
word_freq_receive	1.7942	0.235	7.625	0.000	1.333	2.255

Figure 2: Logit Results

3 Logistic regression First/Second-Order Methods

2. • **Method:** Random initialization using np.random.randn.
   • **Reason:** This adds randomness to avoid symmetric updates during optimization.
3. • **Loss Function :** Binary Cross Entropy.
   • **Reason :** This is appropriate for classification problems, as it penalizes incorrect predictions more when the prediction is very wrong.
4. • Batch gradient descent

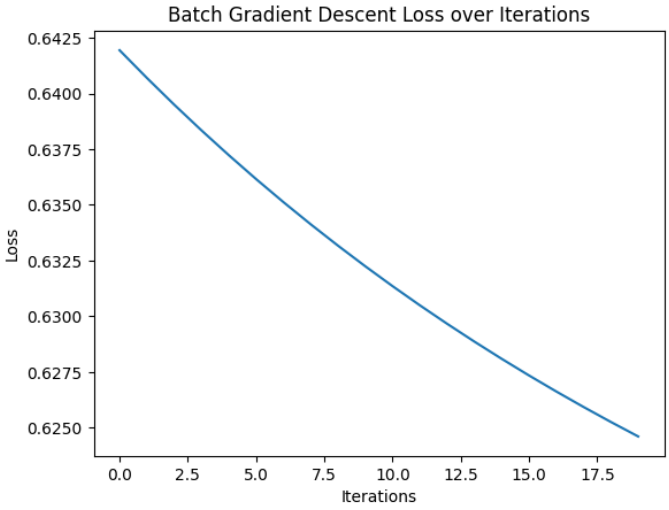


Figure 3: Loss over iterations for batch gradient descent

5. • Stochastic Gradient Descent

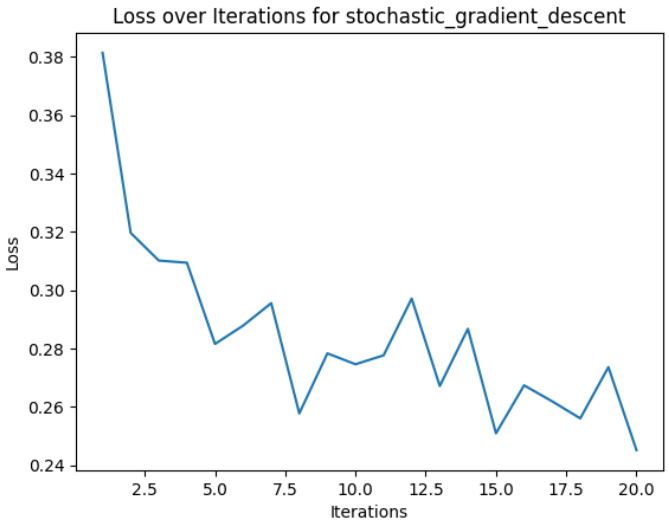


Figure 4: Loss over iterations for stochastic gradient descent

7. • Newton's Method

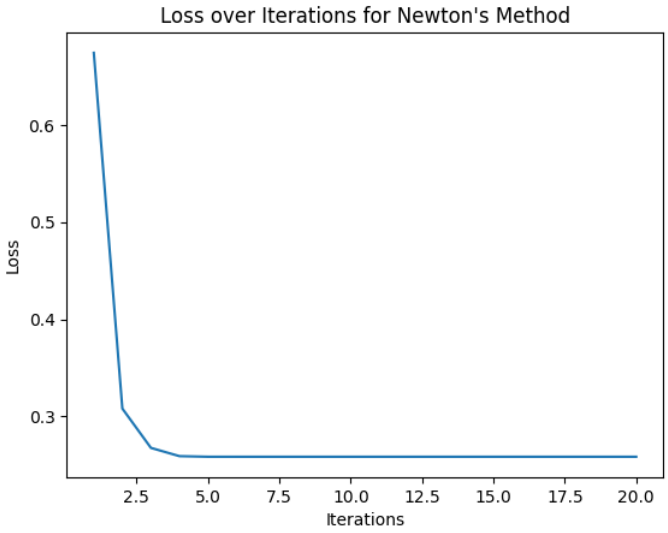


Figure 5: Loss over iterations for batch Newton's Method

8. • Batch Gradient Descent shows steady progress but is slower and Stochastic Gradient Descent fluctuates more.
   • Newton's Method converges faster than other two methods.

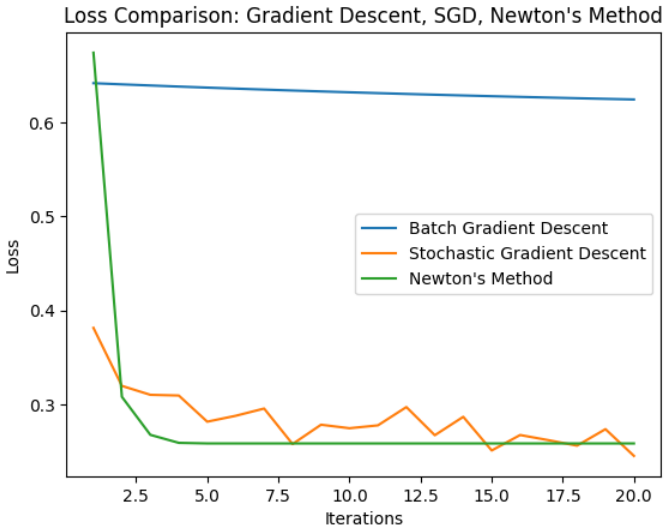


Figure 6: Batch Gradient Descent vs Stochastic Gradient Descent vs Newton's Method

9. • **Early Stopping :** Stop iterating when the change in loss between iterations falls below a threshold.
   • **Cross-validation :** Use cross-validation to evaluate the model performance over different iteration counts and choose the iteration count that minimizes the validation error.

10. • The convergence change due to the different separability of the classes in the new dataset. Since the classes are closer together, gradient descent converge more slowly

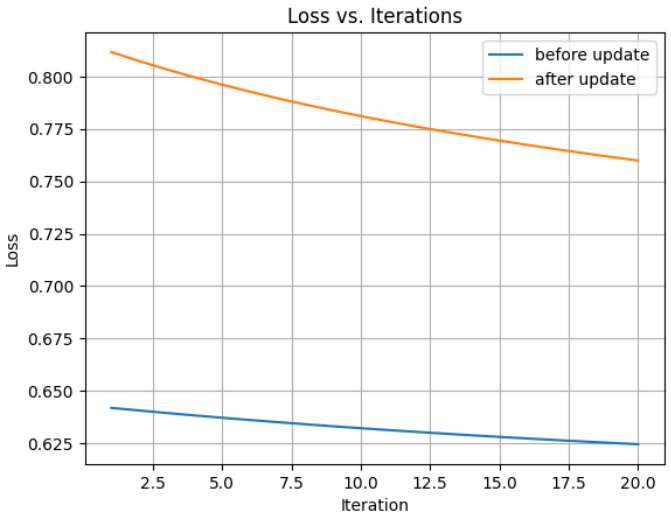


Figure 7: Batch Gradient Descent on New Data (Changed Centers)