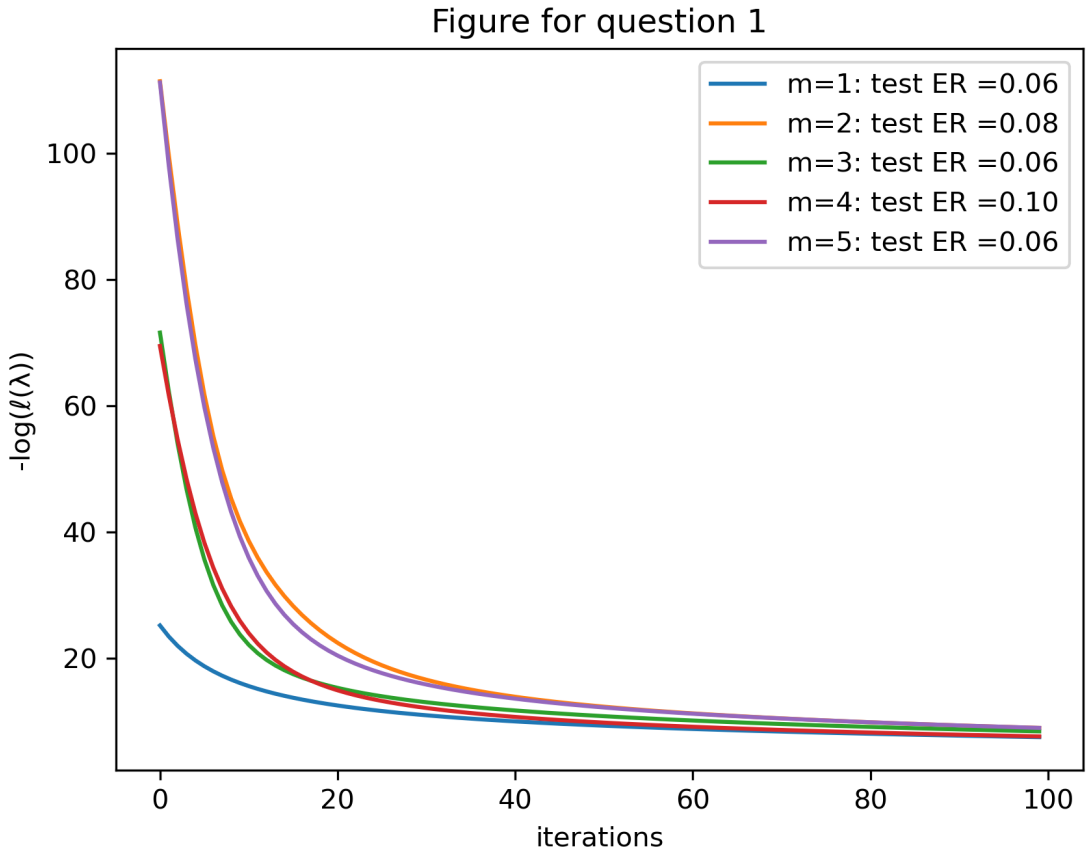
CS5100 CW Assignment 1

Section 1:

Q1 - Synthetic data



What conclusions can you draw from the obtained results?

The objective function shows a decreasing trend with the number of iterations for all five initialisations, which is expected and suggests that the algorithm is converging.

The Error Rate (ER) for all models are quite low, which means good generalisation performance.

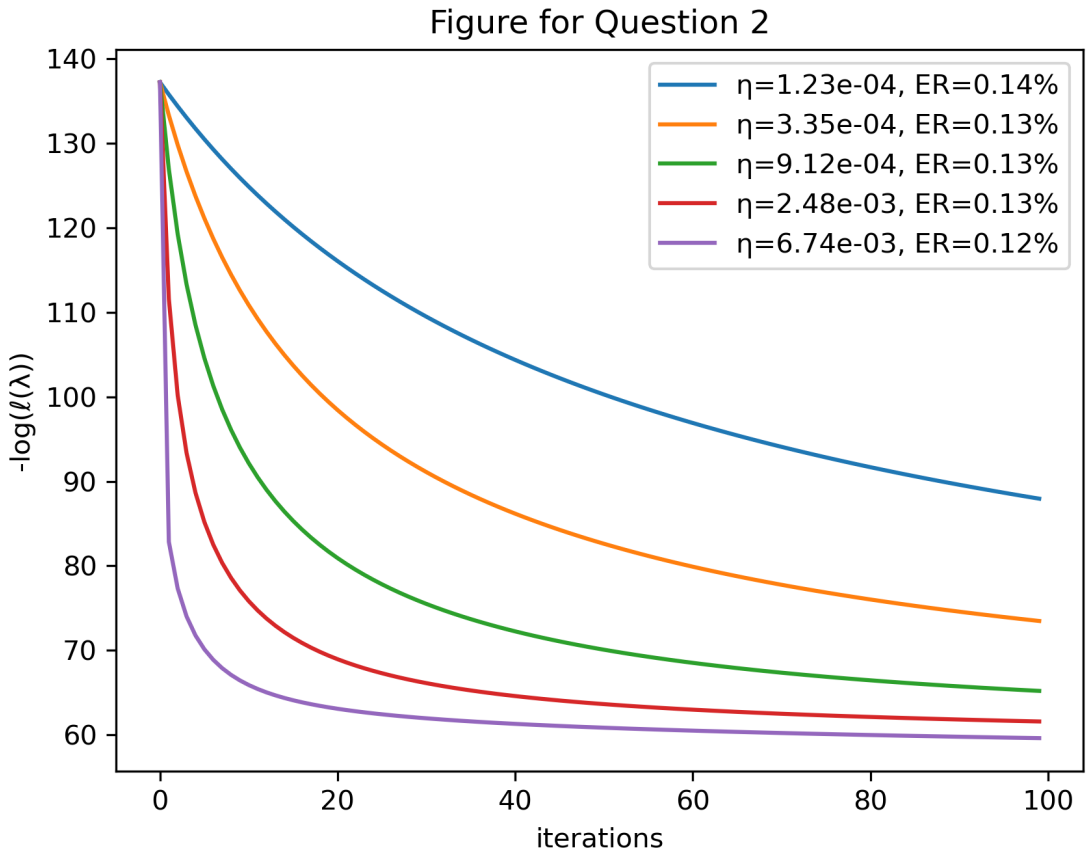
Does the model performance depend on the initialisation?

Yes, the model performance in this case depends on the initialisation. Different initialisations lead to different rates of convergence and different final error rates.

Can you say anything about the global or local optimality of your estimates? Since all lines converge and their error rates are really close, we can conclude that, that is a global optimum instead of local optima.

Section 2:

Q2 - Cars data set



What is the best learning rate for this task?

The best learning rate is η=6.74e-03 as it has the lowest ER.

What happens if you consider higher values of η, e.g. η = 1?

When η = 1, the error rate is 0.46%.

A higher η can cause:

- the optimisation process to take big steps and possibly overshoot the minimum. This means that instead of converging to the best possible solution, the optimiser may bounce around and not settle.
- the loss function to swing around the minimum, which prevents the model from having a stable convergence.

Briefly explain why, in general, you expect ℓ(λ(t+1)) ≤ ℓ(λ(t)) for all t = 1, . . . , Tepochs if η is small enough and λ(t+1) = λ(t) − η∇ℓ(λ (t)).

λ(t+1) = λ(t) − η∇ℓ(λ (t)) is for updating the gradient decent:

- $\lambda(t)$: is the current parameter value at iteration
- η is the learning rate.
- $\nabla \ell(\lambda(t))$ is the gradient of the loss function ℓ with respect to the parameter λ at iteration t.

The function above aims to decrease the loss ℓ by taking a step in the direction opposite to the gradient(the negative sign).

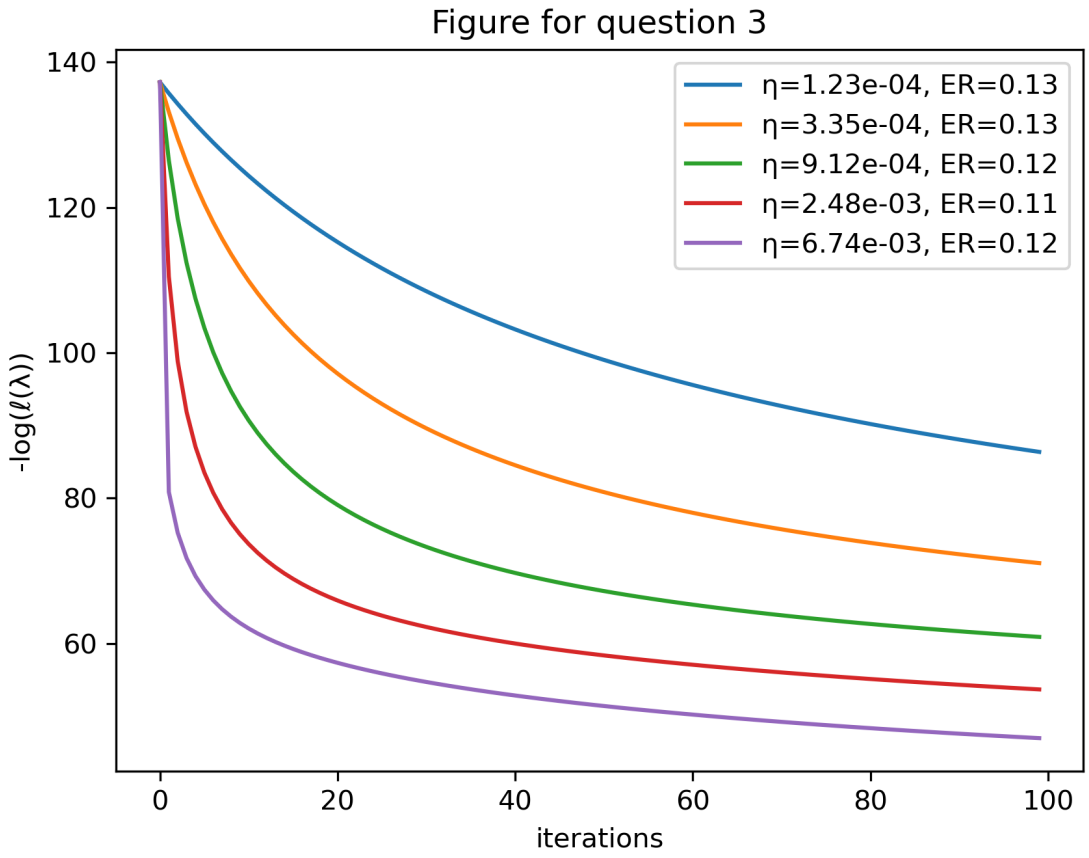
$\ell(\lambda(t+1)) \leq \ell(\lambda(t))$ for all t is based on the idea that if we take a small enough step(small η) in the direction of the gradient, we should move closer to the minimum of the loss function. For the following two reasons:

- Direction of Steepest Descent
- Step Size Matters

In conclusion, for a small enough learning rate η, the gradient descent update makes sure of a decrease in the loss function with each iteration, leading to $\ell(\lambda(t+1)) \leq \ell(\lambda(t))$ for all t.

Section 3:

2.3 Q3 - Nominal variables



Does including Znorigin improve the model performance?

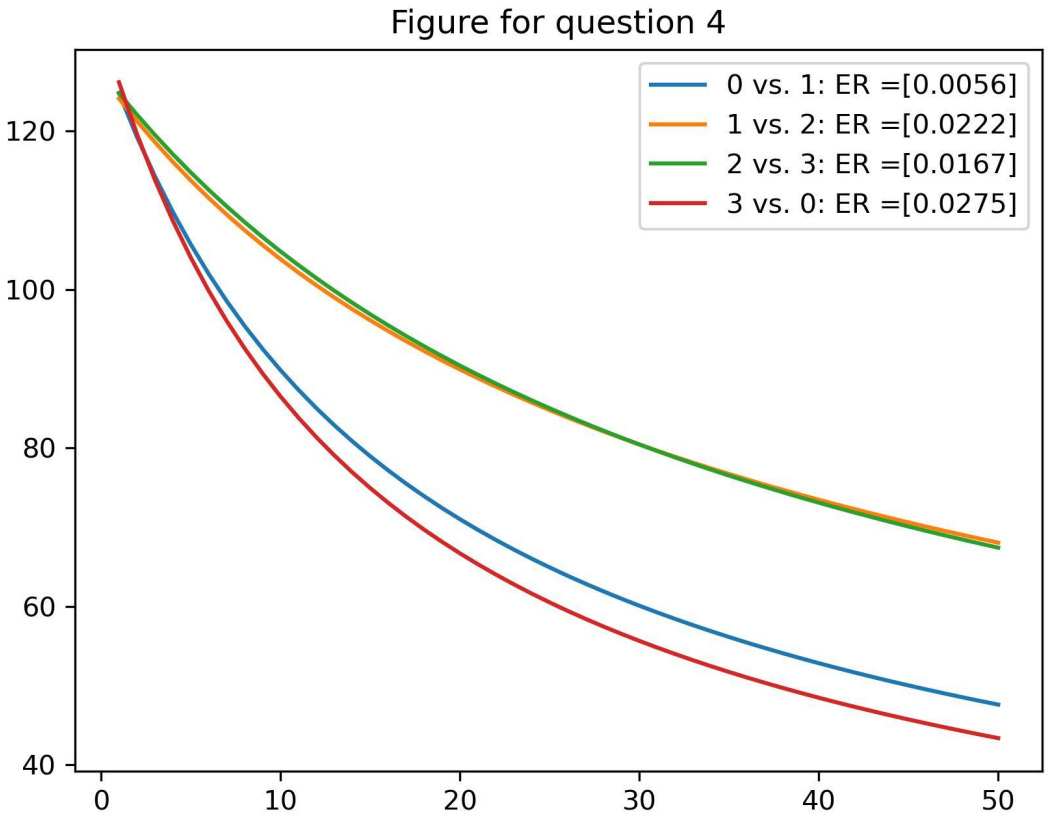
Since the Error rates are slightly lower in Question 3 this implies that including Znorigin has improved the model's performance.

Do better performances on the training set imply better performances on the test set?

No, While a model might perform exceptionally well on the training set, it doesn't necessarily mean it will do similarly well on the test set. If a model is too tailored to the training data and fails to generalise to new, unseen data, this is called overfitting.

Section 4:

Q4 - Digits data set



Can you draw any conclusion from this experiment?

The model is converging for all the binary classification tasks. The value of objective value decreases as the number of epochs increases, which is to say of a successful training process.

What are the hardest and the simplest discrimination tasks?

Hardest Task: Based on the error rates, the 3 vs. 0 task has the highest error rate of 0.0275. Thus, differentiating between 3 and 0 is the hardest among the tasks.

Simplest Task: The 0 vs. 1 task has the lowest error rate of 0.0056, making it the simplest discrimination task among the pairs.

Are you surprised a logistic regression model obtains such good results?

Yes, especially since logistic regression is a linear regression model.

Do you think the models are overfitting the training data?

```
Results for 0 vs. 1:
Training Error Rate: 0.0056
Test Error Rate: 0.0000

Results for 1 vs. 2:
Training Error Rate: 0.0279
Test Error Rate: 0.0278

Results for 2 vs. 3:
Training Error Rate: 0.0167
Test Error Rate: 0.0278

Results for 3 vs. 0:
Training Error Rate: 0.0000
Test Error Rate: 0.0166
```

From the results above we can conclude that overfitting is happening in the "3 vs. 0" task, as indicated by the perfect training ER but non-zero test ER. The "2 vs. 3" task also is overfitting, since the test ER is higher compared to the training ER. On the other hand, the "0 vs. 1" and "1 vs. 2" tasks do not show signs of overfitting.

Appendix:

Q1)

```
def objective(par, data):
    ell = 0
    X, Y = data
    for n in range(len(X)):
        x, y = X[n], Y[n]
        f = model(x, par)
        s = -np.log(f) if y == 1 else -np.log(1-f)
        ell = ell + s
    return ell
```

```
def gradient(par, data):
    grad = np.zeros(len(par))
    X, Y = data
    for n in range(len(X)):
        x, y = X[n], Y[n]
        f = model(x, par)
        s = -(y - f)/(y * f + (1 - y) * (1 - f))
        grad = grad + s * dModel(x, par)
    return grad
```

```
def train(par0, eta, T, data):
    par = par0
    obj = []
    for t in range(T):
        ell = objective(par, data)
        obj.append(ell)
        grad = gradient(par, data)
        par = par - eta * grad
    return par, obj
```

Q3)

```
Znorigin = list(data['origin'].to_numpy())

unique_origins = list(set(Znorigin))

dummy_vars_list = []

for record_origin in Znorigin:
    dummy = [1 if record_origin == origin else 0 for origin in
unique_origins]
    dummy_vars_list.append(dummy)

dummy_vars = np.array(dummy_vars_list)
```

Q4)

```
def create_dataset(X, y, classes):
    X_new = []
    y_new = []
    for i in range(len(y)):
        if y[i] in classes:
            X_new.append(X[i])
            if y[i] == classes[1]:
                y_new.append(1)
            else:
                y_new.append(0)
    return np.array(X_new), np.array(y_new)

D01_X, D01_y = create_dataset(X, y, [0, 1])
D12_X, D12_y = create_dataset(X, y, [1, 2])
D23_X, D23_y = create_dataset(X, y, [2, 3])
D30_X, D30_y = create_dataset(X, y, [3, 0])
```