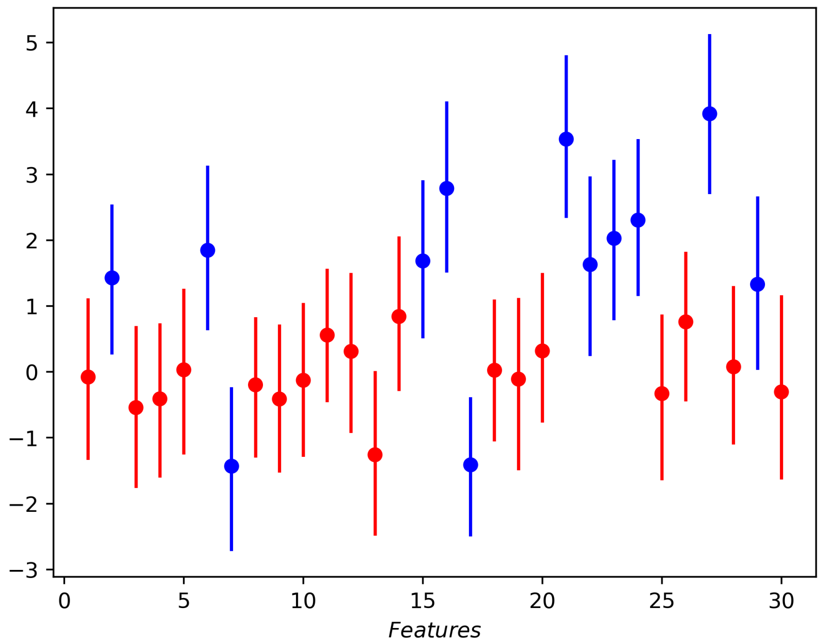
Question 1

a

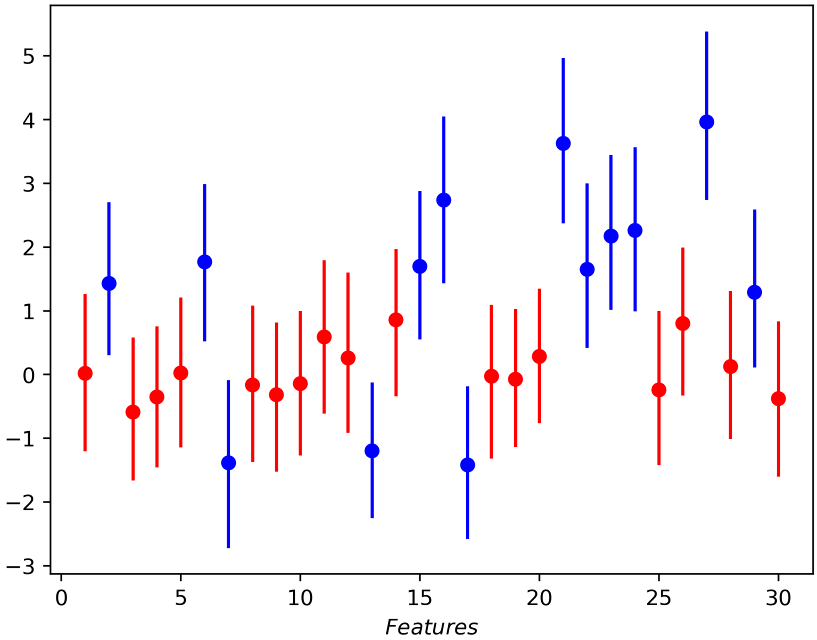


Screenshot of code here: 0.1.

i. C is a hyperparameter used for regularization to prevent overfitting of the Logistics Regression model. This is inversely proportional to the penalty constant λ and penalises models that have a lot of features and therefore the effect is that it reduces each feature importance.

ii. The effect that this has on the Bootstrap graph is that it increases the variance of each feature, therefore making the 90% confidence intervals larger and more reliable since their largeness means they are more likely to include the true value.
At $C = 0.1$, most of the features have their average at 0 with a very small confidence interval. Because I know how the data was generated and I know that the mean for each feature *should* be 0, I know that this is a reliable estimate. However in the real world when I don't know how the data was generated, a model with $C = 0.1$ would probably be overfit on the data and not contain a true mean.

b



Screenshot of code here: 0.2.

Appendix

0.1 q1a

```
def q1a():        You, 2 hours ago • initial
    # Import data
    data = import_data(Q1_DATA_DIR)

    Xtrain = data.iloc[:, :30].to_numpy()
    Ytrain = data.Y.to_numpy()

    # Set random seed
    np.random.seed(12)

    # Set B and C and p
    B = 500
    C = 1000
    p = Xtrain.shape[1]

    # Find confidence
    coef_mat = np.zeros(shape=(B,p))
    for b in range(B):
        b_sample = np.random.choice(np.arange(Xtrain.shape[0]), size=Xtrain.shape[0])
        Xtrain_b = Xtrain[b_sample]
        Ytrain_b = Ytrain[b_sample]
        mod = LogisticRegression(penalty="l1", solver="liblinear", C=C).fit(Xtrain_b, Ytrain_b)
        coef_mat[b,:] = mod.coef_

    means = np.mean(coef_mat, axis=0)
    lower = np.quantile(coef_mat, 0.10, axis=0)
    upper = np.quantile(coef_mat, 0.90, axis=0)

    colors = ["red" if lower[i] <= 0 and upper[i] >= 0 else "blue" for i in range(p)]

    plt.vlines(x=np.arange(1,p+1), ymin=lower, ymax=upper, colors=colors)
    plt.scatter(x=np.arange(1,p+1), y=means, color=colors)
    plt.xlabel("$Features$")
    plt.savefig("./outputs/NPBootstrap.png", dpi=400)
```

0.2 q1b

```python
def q1b():
    # Import data
    data = import_data(Q1_DATA_DIR)

    Xtrain = data.iloc[:, :30].to_numpy()
    Ytrain = data.Y.to_numpy()

    # Set random seed
    np.random.seed(20)

    # Set B and C and p
    B = 500
    C = 1000
    p = Xtrain.shape[1]

    # Train initial model
    mod = LogisticRegression(penalty="l1", solver="liblinear", C=C).fit(Xtrain, Ytrain)
    B0 = mod.intercept_[0]
    coefs = mod.coef_[0]

    # Find confidence
    coef_mat = np.zeros(shape=(B,p))
    for b in range(B):
        # Get X values randomly as before. Here, n = Xtrain.shape[0].
        b_sample = np.random.choice(np.arange(Xtrain.shape[0]), size=Xtrain.shape[0])
        Xtrain_b = Xtrain[b_sample]

        Ytrain_b = np.zeros(shape=(1,Xtrain.shape[0]))
        idx = 0
        for i in Xtrain_b:
            # Calculate "p" for bernoulli for this instance
            prob = math.exp(B0 + coefs.T @ i) / (1 + math.exp(B0 + coefs.T @ i))
```


```python
            prob = math.exp(B0 + coefs.T @ i) / (1 + math.exp(B0 + coefs.T @ i))
```


```python

            # The Ytrain_b must come from a bernoulli distribution
            response = np.random.binomial(n=1, p=prob, size=1)

            # Add to ytrain data
            Ytrain_b[0, idx] = response[0]
            idx +=1

        # Train new model fit to the b_set
        mod_b = LogisticRegression(penalty="l1", solver="liblinear", C=C).fit(Xtrain_b, Ytrain_b[0])

        # store coefficients
        coef_mat[b,:] = mod_b.coef_

    means = np.mean(coef_mat, axis=0)
    lower = np.quantile(coef_mat, 0.10, axis=0)
    upper = np.quantile(coef_mat, 0.90, axis=0)

    colors = ["red" if lower[i] <= 0 and upper[i] >= 0 else "blue" for i in range(p)]

    plt.vlines(x=np.arange(1,p+1), ymin=lower, ymax=upper, colors=colors)
    plt.scatter(x=np.arange(1,p+1), y=means, color=colors)
    plt.xlabel("$Features$")
    plt.savefig("./outputs/NPParameterisedBootstrap.png", dpi=400)
```