CS 6375 Assignment – 1:

Suchith Natraj Javali (SXJ200024)

Harichandana Neralla (HXN210036)

PART -1:

ComputING Train and Test Errors for different number of iterations and learning rates.

```
for iter in [10, 100, 1000, 10000]:
        for a in [0.01, 0.1, 0.33]:
            # INSERT CODE HERE
            lr.fit(Xtrn, ytrn, lr=a, iters=iter)
            weights = lr.w
            train_pred = lr.predict_example(weights, Xtrn)
            test_pred = lr.predict_example(weights, Xtst)
            train_error = lr.compute_error(ytrn, train_pred)
            test_error = lr.compute_error(ytst, test_pred)
            results.append((iter, a, train_error, test_error))

    # Reporting best Parameters found
    min_test_error = min(results, key=lambda x: x[3])[3]
    best_iter, best_lr = min(results, key=lambda x: x[3])[0],
      min(results, key=lambda x: x[3])[1]
    print(f"Minimum test error: {min_test_error}")
    print(f"Best Parameters: \n Best iterations:{best_iter} \n
         Best Learning     Rate: {best_lr}" )
    print("_____________________________________________________________________")
```

Output:

```
_____________________________________________________________________
Minimum test error: 0.2361111111111111
_____________________________________________________________________
Best Parameters:
 Best iterations:100
 Best Learning Rate: 0.33
_____________________________________________________________________
```

PART -2:

Retrain Logistic Regression on the best parameters and store the model as a pickle file.

```
    # retrain the model with best parameters
    lr_best_model = SimpleLogisiticRegression()
    lr_best_model.fit(Xtrn, ytrn, lr=best_lr, iters=best_iter)

    # Code to store as pickle file
    netid = 'hxn210036'
    with open(f'{netid}_model_1.obj', 'wb') as file_pi:
        pickle.dump(lr_best_model,file_pi)
```

Output:

Netid_model_1.obj file in the folder.

Part-3

Compare your model's performance to scikit-learn's LR model's default parameters.

```
X_train, y_train = MTrain[:, 1:], MTrain[:, 0]
    X_test, y_test = MTest[:, 1:], MTest[:, 0]

    scikit_results = []

    scikit_lr = LogisticRegression()
    scikit_lr.fit(X_train, y_train)

    y_train_pred = scikit_lr.predict(X_train)
    y_test_pred = scikit_lr.predict(X_test)

    scikit_trn_error = lr.compute_error(y_train, y_train_pred)
    scikit_tst_error = lr.compute_error(y_test, y_test_pred)

    # compare the scikit train error and test error with part-1 best parameters
errors

    print("scikit error rates:")
    print(f' scikit train error: {scikit_trn_error} \n scikit test
error:{scikit_tst_error}')

    y_best_train_pred = lr_best_model.predict_example(lr_best_model.w,X_train)
    y_best_test_pred = lr_best_model.predict_example(lr_best_model.w, X_test)
```

```
    best_trn_error = lr_best_model.compute_error(y_train, y_best_train_pred)
    best_tst_error = lr_best_model.compute_error(y_test, y_best_test_pred)

    print("_____________________________________________________________")
    print("Error rates of our Model with best parameter:")
    print(f' our train error: {best_trn_error} \n our test
error:{best_tst_error}')
```

Output:

```
 _____________________________________________________________
 scikit error rates:
  scikit train error: 0.16393442622950818
  scikit test error:0.17824074074074073
 _____________________________________________________________
 Error rates of our Model with best parameter:
  our train error: 0.20491803278688525
  our test error:0.2361111111111111
```

PART-4

Plot curves on train and test loss for different learning rates. Using recompute=False might help.

```
for x, a in enumerate([0.01, 0.1, 0.33]):
        lr.fit(Xtrn, ytrn, lr=a, iters=1)
        # INSERT CODE HERE
        train_loss = []
        test_loss = []
        for i in range(11):
            lr.fit(Xtrn, ytrn, lr=a, iters=100, recompute=False)
            weights = lr.w
            # INSERT CODE HERE
            train_pred = lr.predict_example(weights, Xtrn)
            test_pred = lr.predict_example(weights, Xtst)
            train_loss.append(lr.compute_loss(Xtrn, ytrn))
            test_loss.append(lr.compute_loss(Xtst, ytst))
        print("_____________________________________________________________
")
        print(f"For learning rate {a}")
        print(f"Training Loss:{train_loss}")
        print(f"Test Loss:{test_loss}")

        epochs = np.arange(0, 1001, 100)
```

```
        plt.subplot(1, 3, x+1)
        plt.plot(range(0, 1100, 100), train_loss, 'r', label='Training Loss')
        plt.plot(range(0, 1100, 100), test_loss, 'b', label='Test Loss')
        plt.title('Learning Rate: {}'.format(a))
        plt.xlabel('Epochs')
        plt.ylabel('Loss')
        plt.legend()

plt.show()
```

Output:

```
_____________________________________________________________
For learning rate 0.01
Training Loss:[0.639412344222109, 0.6071630327550127, 0.5840692991136244, 0.5668977262197139, 0.5537137440422288, 0.5433245784708609, 0.5349660086715308, 0.52812
87498433576, 0.5224607058704589, 0.5177103577920692, 0.5136928713696771]
Test Loss:[0.6501474796374516, 0.6196406592457427, 0.5977015381359029, 0.5813358873339344, 0.5687381306724804, 0.5587931508477201, 0.5507847616760667, 0.54423418
5881238, 0.5388091539276902, 0.5342712741315733, 0.5304445285547484]
_____________________________________________________________
For learning rate 0.1
Training Loss:[0.517109545261879, 0.4949372358195487, 0.4886878347435536, 0.486520275213296, 0.4856907459352538, 0.48535572592038745, 0.48521604197695717, 0.4851
5663557913113, 0.4851310454518217 3, 0.4851199281540286, 0.4851150702445922]
Test Loss:[0.5337051796692784, 0.5130128867656069, 0.5078465895085429, 0.506475493139457, 0.5062077512209412, 0.5062616299596931, 0.5063892204476542, 0.506513062
6072858, 0.5066130672008764, 0.5066880734434104, 0.5067423226311818]
_____________________________________________________________
For learning rate 0.33
Training Loss:[0.48767313659879896, 0.48524986577348556, 0.4851200431559241, 0.48511183977174155, 0.4851112974145166, 0.48511126077519906, 0.4851112582522757, 0.
48511125807463595, 0.48511125806178634, 0.4851112580608271, 0.4851112580607530 6]
Test Loss:[0.507133208781503, 0.5063431728846418, 0.5066864296479261, 0.5068139335747123, 0.5068527248292622, 0.5068641922647806, 0.5068675791300398, 0.506868583
7712619, 0.5068688832608811, 0.5068689729543341, 0.5068689999266436]
(mlenv) suchithnj@SUCHITHs-Air ML-Programming-Assignments %
```