Homework 2: Logistic Regression **CSCE 633**
Due: 11:59pm on Jun 18, 2024

Instructions for homework submission
a) Please write your code in a Jupyter Notebook. Please explain your thought process, results, and observations in markdown cells. Please do not just include your code without justification.
b) Your submission should be a Jupyter Notebook that can be run seamlessly and performs all the required steps one after another. Any submission with a runtime error would result in lost points.
c) Allowed libraries include NumPy, Pandas, Matplotlib, and scikit-learn (in part B).
d) Please start early :)
e) Total: 100 points

Part A - Logistic Regression (50 points)

We will first do data pre-processing using Hitters Dataset, downloaded from the link.

Since there's a column *NewLeague* which only includes *A* and *N*, we can use this dataset for a binary classification problem. Then we will apply simple Logistic Regression to our pre-processed data.

Regression models can predict probabilities, such as the probability that the new league of the player is 'N'. Thus the model can be used as a classifier by imposing a decision rule - if the probability is more than 50%, we can say the new league is 'N'.

1. Read Hitters Dataset into a *pandas* dataframe.
2. Pre-process the data. How does the data look like? What's the shape of the data? Drop all the rows with any missing values. Extract the features and the label from the data, assuming the label is *NewLeague* and the others are considered features. Then do one-hot encoding for categorical features: (1) separate numerical columns from nonnumerical columns; (2) use *get_dummies* or *replace* for transforming to categorical; (3) concatenate both parts. Transform the output into numerical format. Transform 'A' in label to 0 and 'N' to 1.
3. Split the data into train and test sets. Here we use 70% of the data as a training set, and 30% as a testing set. You should have the following four splits: X_train, X_test and y_train, y_test.
4. Train a logistic regression model. You may only use library NumPy and here you are supposed to implement everything from scratch.
5. Please provide the coefficients for each feature for Logistic model.
6. Please plot the ROC curve for Logistic model. What is the area under the curve?
7. For Logistic model, what is the optimal decision threshold to maximize the f1 score?

Part B - Regularization (50 points)

Our dataset is still Hitters Dataset. Since labels in column *NewLeague* only include *A* and *N*, you can try to use your regression models as binary classifier.

Note that in Python, we refer to regularization strength as *alpha*.

1. Split the data into 3 sets - train, validation, and test sets. This time we use 60% of the data as a training set, 20% as a validation set and 20% as a testing set. You should have the following six splits: X_train, X_val, X_test and y_train, y_val, y_test.

2. Write a function *train_ridge* to implement a *Ridge* model for Hitters dataset that returns a dictionary with *alpha values* as keys and corresponding *mean aucs* as value pairs. In the function, apply bootstrapping and repeat the training process for *n_bootstraps* = 1000 times, and for each time train the model for *max_iter* iterations with all *alpha*s from *alpha_vals*. Compute the AUC values with the validation set and append the values each time to list *aucs_ridge*. Then calculate *mean auc* over time for each *alpha*. Please describe your hyperparameter tuning procedures and optimal *alpha* in *alpha_vals* = [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3] that gives the best model.

3. Write a function *train_lasso* to implement a *Lasso* model for Hitters dataset that returns a dictionary with *alpha values* as keys and corresponding *mean aucs* as value pairs. In the function, apply bootstrapping and repeat the training process for *n_bootstraps* = 1000 times, and for each time train the model for *max_iter* iterations with all *alpha*s from *alpha_vals*. Compute the AUC values with the validation set and append the values each time to list *aucs_lasso*. Then calculate *mean auc* over time for each *alpha*. Please describe your hyperparameter tuning procedures and optimal *alpha* in *alpha_vals* = [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3] that gives the best model.

4. Write a function *ridge_coefficients* that returns a tuple of trained ridge model with *max_iter* iterations and optimal *alpha* as well as the model's coefficients. Next, write a function *lasso_coefficients* that returns a tuple of trained lasso model with *max_iter* iterations and optimal *alpha* as well as the model's coefficients. Compare the coefficients for each feature for Lasso and Ridge models. Are they the same? If they are different, why?

5. Write a function *ridge_area_under_curve* that returns area under curve measurements, and a function *lasso_area_under_curve* that returns area under curve measurements. Plot the ROC curve of the Ridge Model and the Lasso Model. Include axes labels, legend, and title in the plots.