Literature Review:

Signal or background? Our selected dataset, provided by the UC Irvine Machine Learning Repository, looks in-depth at both raw and manipulated data on Higgs Boson particles. We aim to categorize this data determining if it is either a real instance of the ‘God’ particle, known as a signal, or merely background effects. The large-picture goal of this binary classification is to lead to an understanding of what events lead to the presence of these particles. A signal response represents the events that led to this event. For example, this occurrence could come from just a single feature or possibly could be when multiple features are all present. Background events are more random. These could be any random features that represent any number of other events happening simultaneously during data collection. The dataset we have chosen to use has a selection of 21 traditional features that together could identify a Higgs boson particle. Each hypothetical particle is accompanied by 7 other features, which were custom-made to assist in the identification of our question of classification.

Logistic Regression:

Preparing the data:

Working with this specific dataset was intriguing because there were many layers to our classification problem. There are 28 features, 21 of which are traditional features, regarding a single aspect of the data, and 7 others that were specially designed to identify which group the piece of data is classified as. From this, all steps were completed twice to prepare for testing. An X train and test were created for the 21 features and another for the 7 special features. With the data split into two categories for testing, it was ready to proceed to the next step.

Initial Results:

Beginning the testing of the sklearn Logistic Regression model I worked with a base case of no special hyperparameters. The results were low, around a 50% match, which was very average given the binary classification at hand with a fairly balanced dataset. **FIGURE X** is a c infusion matrix reviewing the results of this initial model. Wanting to expand from just a basic Logistic Regression approach I also decided to use Stochastic Gradient Descent to help find the best hyperparameters from the best possible logistic regression results.

Hyperparameters:

The hyperparameters available for sklearn’s Logistic Regression are:

Penalty & Regularization type. 'l1' for L1 regularization, 'l2' for L2 regularization, or 'none' for no regularization. //

C & Inverse of regularization strength. Smaller values specify stronger regularization.

Solver & Algorithm to use for optimization. Options include 'liblinear', 'newton-cg', 'lbfgs', 'sag', and 'saga'. The choice of solver depends on the size of the dataset and whether you want to use L1 or L2 regularization. //

Multi_class & Determines how to handle multiple classes. Options are 'ovr' (one-vs-rest) or 'multinomial' (softmax/multiclass logistic regression). //

Max_iter & Maximum number of iterations for the solver to converge. //

Dual & Dual or primal formulation. Set to False unless there are more features than samples. //

Fit_intercept & Whether to include an intercept (bias) term in the model. //

random_state & Seed for random number generation for reproducibility. //

Tol & Tolerance for stopping criteria. It controls the convergence threshold. //

Warm_start & If set to True, the solution of the previous call to fit is used as an initialization for the new call. //

class_weight & Weights associated with classes. Useful for handling imbalanced datasets. //

Verbose & Set to a positive integer for verbosity during optimization. //

N_jobs & Number of CPU cores to use during training (-1 means using all available cores). //

When working with my models, for simplification I only worked with penalty and solver, as they have the greatest control over the results of the model.

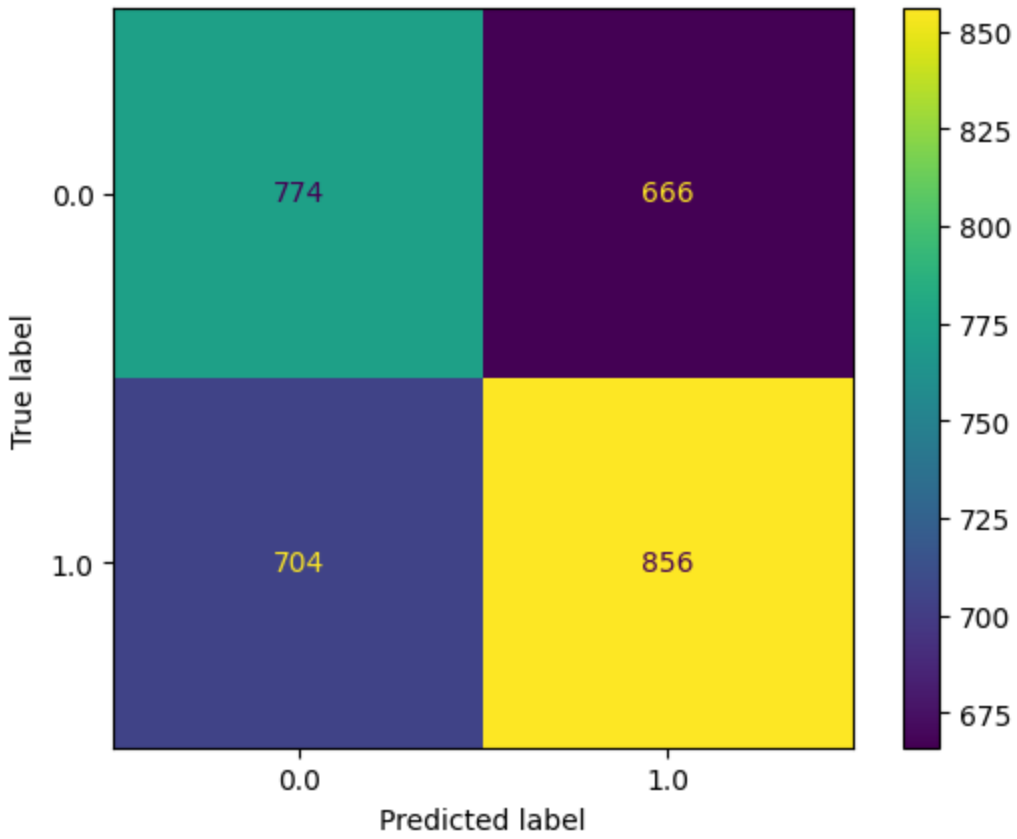
Finding the Best Hyperparameters:

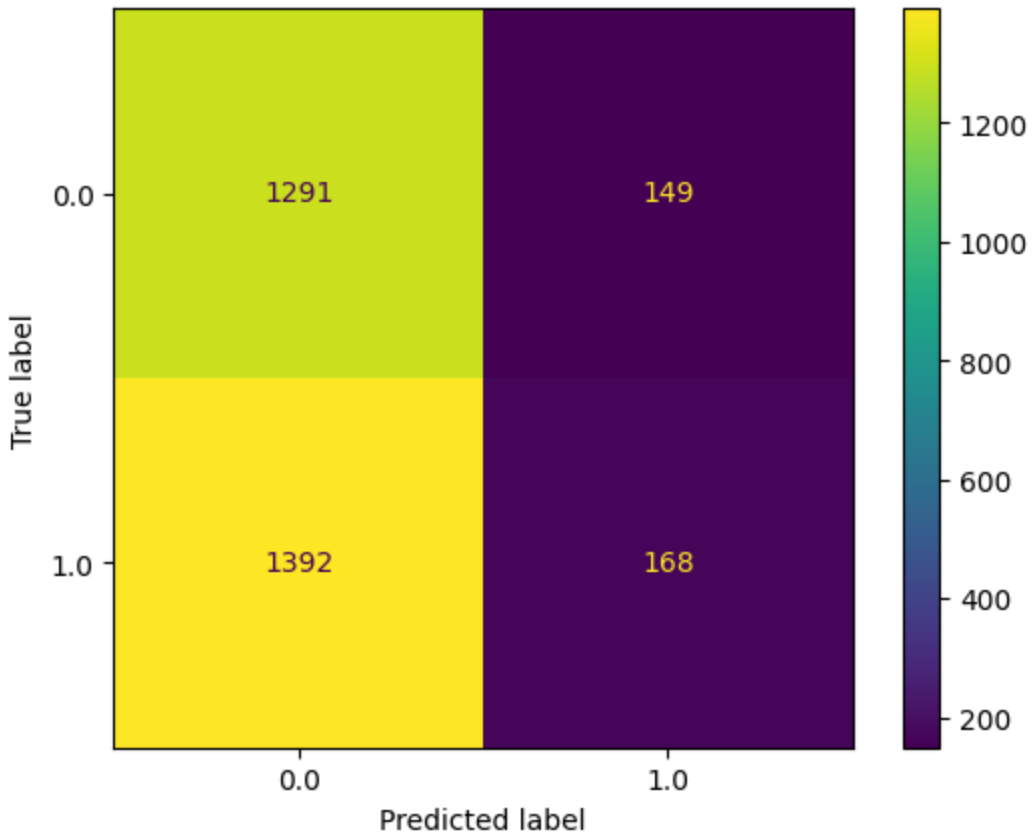
To test many models I created a python dictionary with all the possible combos of penalty and solvers for logistic regression. Running many models I found that the penalty L1 regularization had the worst results, they were often just as low as the default parameters or slightly above. Moving forward with L2 regularization was better in almost every case. When it came to the many solvers some had stronger values than others. The worst results came from liblinear, this solver's L2 compatibility issues held it back when running tests. Through testing, it was also discovered the difference between the regular 21 features and the special 7 features. When those 7 features were used to train and test the models, everything else could be the same and the results would be about 10% more accurate.

Final Model and Results:

The final model with the highest results came when working with the L2 regularization penalty and either the newton-cg, lbfgs, or sag solver. This was the case in both scenarios of 21 regular features and 7 specially developed features. See **FIGURE X + 1** to see the higher results of an L2 and sag model. This model used the special 7 features to reach 63% accuracy. This was the highest accuracy reached using Logistic Regression and was guided by the results of the Stochastic Gradient Descent.

Graphs:


774
666
704
856
0.0
1.0
True label
0.0
1.0
Predicted label
850
825
800
775
750
725
700
675


1291
149
1392
168
0.0
1.0
True label
0.0
1.0
Predicted label
1200
1000
800
600
400
200