Group Project 1: Regression Analysis for Kaggle Competition

Course: Spring 2025 DS310: Machine Learning for Data Analytics

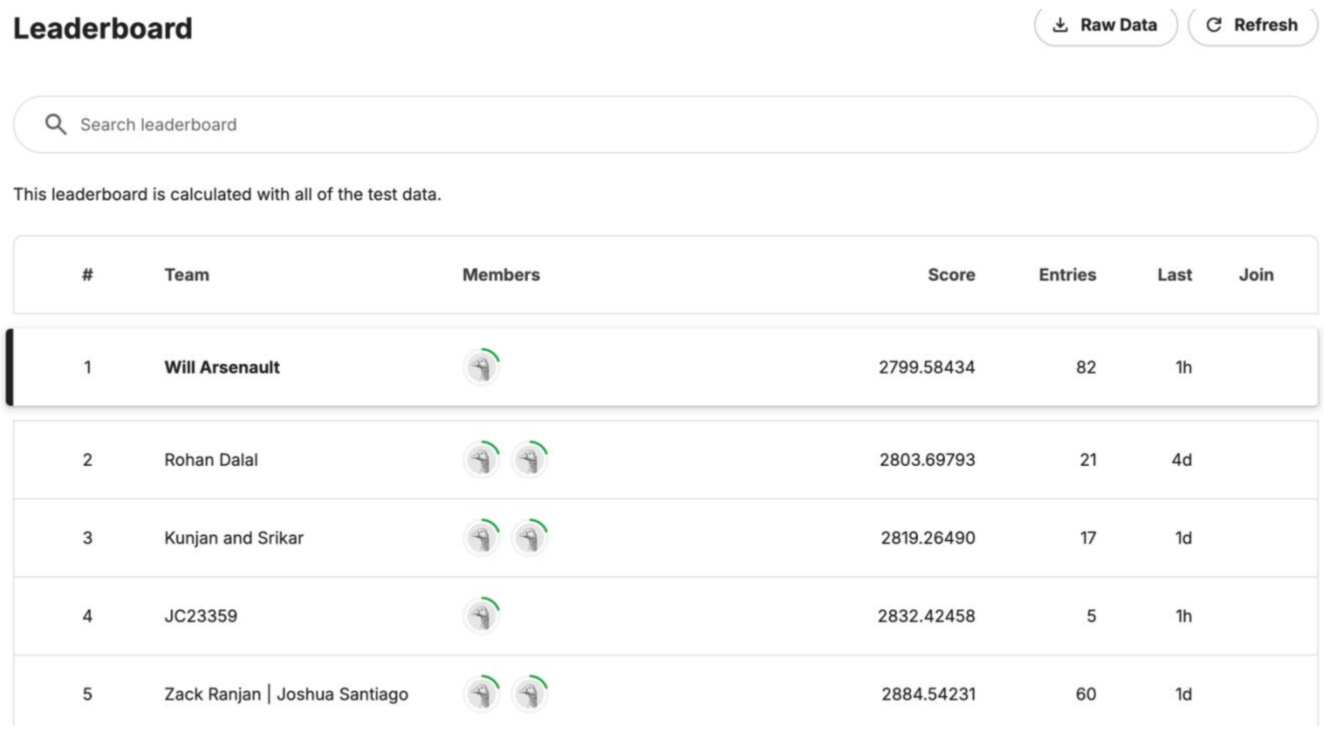
Instructors: Lu Lin, Hangfan Zhang

Team Members: Will Arsenault

Kaggle Account Name: willarsenault1

PSU Email: wca5061@psu.edu

Leaderboard Performance (also on pg. 8):



Leaderboard

This leaderboard is calculated with all of the test data.

#	Team	Members	Score	Entries	Last	Join
1	**Will Arsenault**		2799.58434	82	1h	
2	Rohan Dalal		2803.69793	21	4d	
3	Kunjan and Srikar		2819.26490	17	1d	
4	JC23359		2832.42458	5	1h	
5	Zack Ranjan | Joshua Santiago		2884.54231	60	1d	

Abstract

In this project, I explored the prediction of diabetic levels using 64 different patient physical measurements from 160 patients through a regression-based approach. I did this by implementing a custom linear regression model that used mini-batch gradient descent with momentum and L2 regularization. The goal of this project was to construct a model that balances fitting the training data with generalizability to new, unseen data or cases. In addition to standardizing features using the StandardScaler, I applied a feature selection process with VarianceThreshold in order to remove low-variance features. After that I used PCA to reduce the dimensionality to the top 30 components. Furthermore, I integrated 5-fold cross-validation and performed extensive hyperparameter tuning with GridSearchCV in order to adjust the learning rate, number of iterations, batch size, momentum, and L2 regularization strength to make sure that there was stable convergence and prevent overfitting. What made this project particularly enjoyable for me was the opportunity to directly apply key concepts from class, such as gradient descent, proper feature normalization with some of the machine learning concepts that I knew before this class such as momentum and PCA. In the end, my findings showed that the combination of these machine learning techniques can yield a very robust predictive model. This was evidenced by a competitive mean squared error on the Kaggle validation sets. This project has deepened my understanding of data handling, model selection, and iterative optimization to achieve strong predictive performance.

Introduction

Diabetes is one of the most significant health issues not only in the United States but globally. Early prediction of diabetic levels is crucial for effective intervention by doctors and for treatment planning. The goal of this project is to use patient measurement data to predict diabetic levels, which in turn could aid in early detection and intervention with real-world implications.

My approach centered around a linear regression model implemented from the start. Later I enhanced this model by mini-batch gradient descent with momentum and L2 regularization. In class, we discussed gradient descent which is a method that iteratively minimizes the loss function by adjusting model parameters. In this project, I explored how changes in the learning rate, iteration count, and the inclusion of momentum can influence the convergence of the model. Given the limited training data incorporating L2 regularization was also essential to penalize large weights and help prevent overfitting.

Another important aspect of my strategy was data preprocessing. I firstly used StandardScaler for feature scaling to normalize the features to zero mean and unit variance. I also applied a VarianceThreshold to remove low variance features and used PCA to reduce the dimensionality to the top 30 components. Furthermore, I utilized 5-fold cross validation via GridSearchCV to reliably assess the model's performance and fine tune the hyperparameters learning rate, batch size, momentum, number of iterations, and L2 regularization strength. This process provided a clearer picture of the model's performance in terms of mean squared error (MSE) and how it was going to perform on the Kaggle validation data. By systematically combining feature scaling, gradient descent optimization techniques, regularization, and cross validation, this project shows how these methods can be integrated to build an effective predictive model.

Data Preprocessing and Exploration

The first step in my project was to get familiar with the datasets that I was given. I believe it's essential to understand the data before going into the actual modeling. I started by looking at the the raw inputs from the CSV files. The training data consisted of measurements for 160 patients, with 64 feature columns representing various physical attributes, and a target column indicating the diabetic level. My initial look at the data also involved generating summary statistics and simple visualizations to assess the data's quality and distribution. Although the raw features appeared relatively standardized and simple I still applied the StandardScaler later in the preprocessing pipeline because this was important to make sure that any minor differences between the training and test data distributions were handled consistently. This was particularly important to my project because differences in feature scale can affect the convergence speed during gradient descent optimization.

Feature Engineering and Scaling

Recognizing the need for uniform feature contributions, I applied the StandardScaler from scikit-learn in order to transform the features so that each has a mean of zero and a standard deviation of one. This preprocessing step ensures that no single measurement disproportionately impacts the learning process. This is important when using gradient descent. Before the scaling, I also filtered out features with very low variance using a VarianceThreshold and then applied PCA to reduce the dimensionality to the top 30 principal components. These steps helped in achieving smoother convergence and preventing numerical instability during training. I complemented these steps with exploratory data analysis (EDA), after have using histograms and box plots to visualize feature distributions and check for outliers.

Overall, this thorough preprocessing phase set a strong foundation for building a balanced and robust predictive model.

Model Building and Methodology

Linear Regression

In developing the predictive model, I started with a linear regression frame which I implemented from scratch using mini-batch gradient descent with momentum and L2 regularization. Even though linear regression is relatively simple, it gave me a good and robust starting point for handling a continuous numerical response variable like diabetic levels. In class, we also talked about how a linear combination of input variables can effectively capture relationships in data, so this allowed me to apply that concept

Gradient Descent

A key component of my approach was using a custom mini-batch gradient descent algorithm with momentum and L2 regularization to optimize the regression model. In this process, the Mean Squared Error (MSE) is minimized by iteratively updating the weights and bias based on the negative gradient of the loss function. Using the momentum accelerates the convergence by incorporating information from previous gradient updates. This all happens while the L2 regularization penalizes large weights to help prevent overfitting. Balancing the learning rate and number of iterations (with other hyperparameters like momentum and batch size) was very important. Having a high learning rate risks overshooting the optimal parameters, whereas a low learning rate can slow down convergence, so you have to find a balance.

L2 Regularization

After implementing gradient descent, I enhanced the model’s robustness more by incorporating L2 regularization. Using L2 regularization adds an additional term to the loss function that penalizes the magnitude of the coefficients which prevents overfitting to the training data. This is particularly relevant when dealing with 64 features for only 160 observations, which is objectively a small amount of data. By tuning the regularization parameter (λ), I was able to get a balance between model complexity and generalizability. Intertwining mini-batch gradient descent (with momentum) and L2 regularization was one of the main reasons my model could effectively generalize to new unseen data and got such a low validation MSE.

Hyperparameter Tuning

Hyperparameter choices such as the learning rate, number of iterations, batch size, momentum, and regularization strength (λ) were critical to the model’s ability to generalize to unseen data. The learning rate had to be small enough for stable convergence but large enough to avoid overly long and computationally expensive training times. I tested iteration counts of 120, 150, and 180 through a grid search. Then I selected the best iteration count based on cross-validation results. I also explored different batch sizes (16, 32, 64) and momentum values (0.8, 0.9, 0.99). This thorough search allowed me to find a relationship between these hyperparameters that minimized the mean squared error (MSE) while staying computationally efficient.

Cross Validation

Another important aspect of my project methodology was using cross-validation to evaluate the model’s performance because we were not given the labeled validation data. I used a 5-Fold Cross-Validation strategy, implemented through GridSearchCV which we talked about in class this week. This divides the dataset into five subsets, trains the model on four of them, and uses the remaining subset for validation. This process is repeated five times which makes sure every observation is used for both training and validation. The average Mean Squared Error (MSE) across these folds provided a reliable estimate of how good the model was likely to perform on unseen data (ie. the Kaggle validation data).

Using MSE metric was also crucial for the hyperparameter tuning. By tracking the average MSE across the cross-validation folds this allowed me to look at the learning progress and make decisions regarding further tuning or adjustments. This continuous evaluation helped me understand the model's performance and guided the selection of the best combination of hyperparameters.

I think that my overall model building process and methodology in this project reflect several key lessons I learned from the course. This starts at linear regression and gradient descent to the applications of L2 regularization and 5-Fold Cross-Validation. Each of these components played a crucial role in the success of my final model. This process showed me how careful model evaluation and hyperparameter tuning can give a more robust and reliable predictive model.

Model Training and Evaluation

Model Training Process

After completing the data preprocessing and outlining my methodology, the next step was to actually train and evaluate the regression model. I implemented mini-batch gradient descent with momentum and L2 regularization to minimize the Mean Squared Error (MSE) on the training data. The final model was trained using **150 iterations,** with a **learning rate of 0.002**, and an **L2 regularization parameter of 0.01** which was determined by the grid search. I also looked at the loss value at regular intervals to ensure that the model was converging and to observe how the parameters were updated iteratively. This gave me a reinforcement of how important the balance between learning rate and convergence speed is.

Mini Batch Gradient Descent

In this project, I used mini batch gradient descent as the main optimization strategy in my custom regression model. Mini-batch gradient descent finds a balance by processing small subsets of the data at each iteration. For every training iteration, the dataset is first shuffled to ensure randomness and then divided into mini batches based on a specified batch size. For each mini batch, the model computes predictions, evaluates the gradient of the Mean Squared Error (MSE) with respect to the weights and bias, and then updates these parameters using the momentum. Using momentum was important since it helps accelerate convergence by taking into account previous gradient directions. While the L2 regularization is applied to control model complexity and prevent overfitting, the batch size itself was treated as a hyperparameter, optimized alongside others like the learning rate, iteration count, momentum, and the L2 regularization strength.

5-Fold Cross Validation

I implemented 5-Fold Cross-Validation to make sure that the model's performance wasn’t too optimistic or tied to a specific data split. This method as I described before splits the dataset into five subsets and for each fold, the model trains on four subsets and validates on the remaining one which gives five independent MSE scores. Averaging these scores gave me a good and robust estimate of the model’s performance and its generalizability to unseen data, such as the Kaggle validation set. Overall, this cross-validation process not only strengthened my grasp of model evaluation but also highlighted the importance of effectively using and exploiting all the given data.

Final Training on Full Dataset

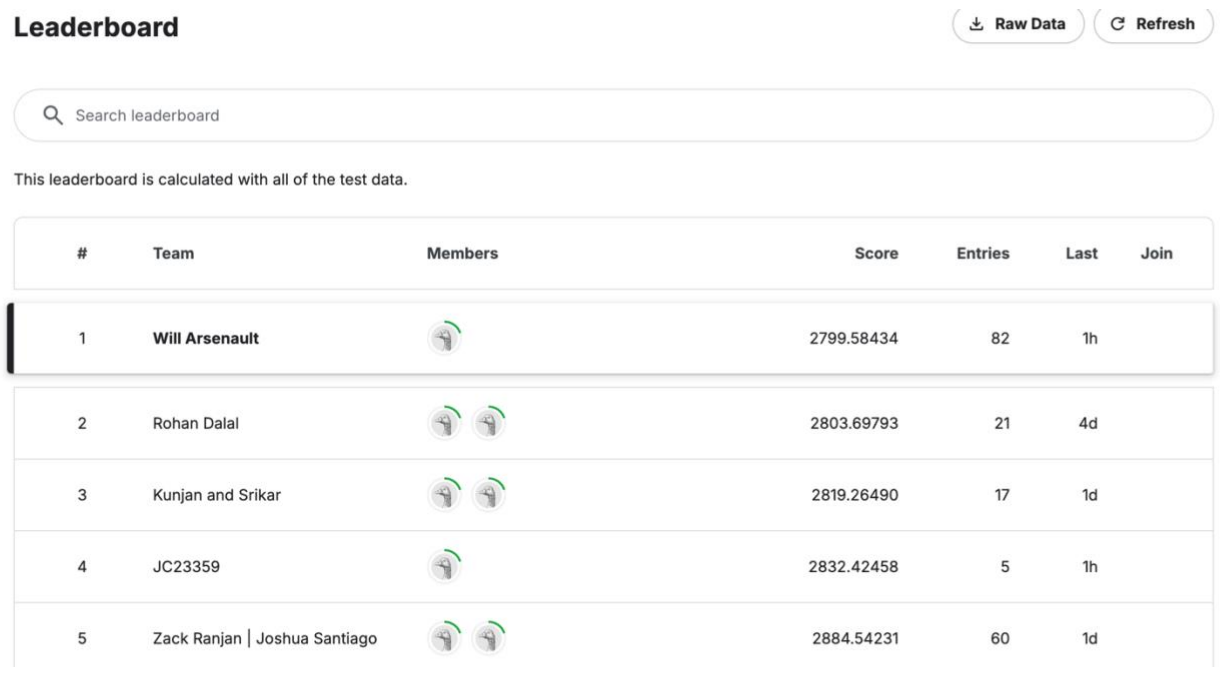
After I was satisfied with the model’s average cross-validation performance, I retrained it using the entire training dataset with the best hyperparameters that were found through the grid search. This step was essential because it maximized the data that I was given for learning underlying patterns ensuring the model was as well-tuned as possible before making predictions. I also maintained consistency by applying the same scaling (and other preprocessing steps) to both the training and test data. This prevents discrepancies in feature scales and helps ensure reliable performance on unseen data.

Generating Predictions and Submission Preparation

Once the final training was complete and the best model is selected via GridSearchCV, I applied the parameters to the transformed test data. The test data underwent the same exact preprocessing process that I followed before. This included the removing of low-variance features via VarianceThreshold, dimensionality reduction with PCA (the top 30 components), and feature scaling using StandardScaler. I did this to make sure there was consistency with the training set. Using the final model, predictions for the diabetic levels were generated on the scaled test dataset.

These predictions were then compiled into a CSV file (named "y_test.csv") formatted according to Kaggle’s submission requirements, containing both the patient IDs and the predicted diabetic levels. This final stage brought together all of the main concepts from data preprocessing, feature selection, model optimization, and evaluation. The entire process, from training with mini-batch gradient descent (with momentum and L2 regularization) to generating final predictions, helped me learn how model evaluation and hyperparameter tuning can lead to robust, real-world accurate predictions

Leaderboard Performance



This leaderboard is calculated with all of the test data.

#	Team	Members	Score	Entries	Last	Join
1	**Will Arsenault**		2799.58434	82	1h	
2	Rohan Dalal		2803.69793	21	4d	
3	Kunjan and Srikar		2819.26490	17	1d	
4	JC23359		2832.42458	5	1h	
5	Zack Ranjan | Joshua Santiago		2884.54231	60	1d	

Discussion and Lessons Learned

Performance Evaluation

My model demonstrated very strong performance, finishing **1st on the leaderboard** with a Kaggle Mean Squared Error (MSE) of 2799.58434. This closely aligned with the expectations set by 5-Fold Cross-Validation. The training MSE was around 4313.4749, which shows how despite a higher error on the training set, the model generalizes very well to new, unseen data. Looking at the loss reduction during mini-batch gradient descent (enhanced by momentum and L2 regularization) further reinforced the effectiveness of my approach before each submission to Kaggle. Overall, the MSE scores together with cross-validation gave me a solid indicator of the model's predictive capability and robustness.

Challenges and Limitations

Throughout the project I faced several challenges that gave me very valuable learning opportunities. One of the primary challenges was tuning the hyperparameters. The learning rate and L2 regularization strength (λ) but also the batch size, momentum, and number of iterations were all parameters that I was faced with tuning. As we discussed in class, setting these parameters too high or too low can lead to overshooting the optimal solution or excessively slow convergence. Despite thorough doing the hyperparameter tuning via grid search there still was some variability in the MSE across different folds indicated that the model might benefit from further fine-tuning. Because of this fact I implemented a loop to randomly permute and shuffle the data to reduce the variance. Also while the features were largely normalized, there was still differences between the training and test distributions could still affect the predictions.

Classroom Insights and Lessons Learned

Having this hands-on experience of actually implementing mini-batch gradient descent with momentum and L2 regularization allowed me to put into practice some of the things I learned in class along with the things that I knew previously. For example, the iterative update and the impact of learning rate adjustments became much more concrete during the training process. Also using L2 regularization gave me a demonstration of how to control the model complexity. Lastly, using the 5-fold cross-validation was important to ensuring that my model's performance was not a fluke but a reflection of its true predictive power.

Future Considerations

Even though my current model provides a robust starting point, I feel like there are some areas for future improvement. One thing I could do is to explore alternative regression techniques, such as Lasso regression, which might offer better performance under different conditions. Also, while I used GridSearchCV for the hyperparameter tuning, there are some more advanced methods that I could have used like Bayesian estimation. Another area to consider is the feature engineering. I could have transformed or combined certain features in order to better capture underlying patterns in the data and further boost performance on the validation set.

Conclusion

Summary of Findings

I was able to successfully build a linear regression model using mini-batch gradient descent with momentum and L2 regularization to predict diabetic levels. The overall process, removing low-variance features with VarianceThreshold, applying PCA, standardizing the data with StandardScaler, and using 5-Fold Cross-Validation showed me the key techniques we learned in class, such as iterative optimization and robust model evaluation along with the techniques that I already knew like using VarianceThreshold and applying PCA.

Reflections on Learning

Implementing these methods deepened my understanding of critical concepts like the bias-variance tradeoff and the importance of proper feature scaling. I was able to apply classroom topics like as hyperparameter tuning and regularization. This helped preventing overfitting the model to the training data and improved the model performance.

Final Thoughts:

This project resulted in a competitive model finishing **1st** on the leaderboard in terms of the lowest validation MSE obtained in the class. It also gave me valuable hands-on experience. Looking forward, I think that exploring advanced regression techniques and machine learning models could help enhance model performance while building on foundations that I have used already.