CS 363M Machine Learning Project - Kaggle Competition

Overview

For the project in this class, you will participate in a machine learning competition where you'll apply your ML skills to a real-world dataset. You will work in teams of 3 or 4 students.

The dataset for this competition contains records of 250,000 dairy cows, including details such as age, weight, feed type, vaccination status, weather, and other attributes. Each animal in the dataset has a recorded milk yield, which represents the total volume of milk produced by the cow during the recorded milking period.

Your goal in this competition is to build a machine learning model that predicts the milk yield of each animal, based on its other characteristics.

By accurately predicting milk yield, your model can help optimize feeding and grazing strategies, improve resource planning and management, forecast financial outcomes, monitor animal health status, and enable data-driven decisions for more sustainable farm operations.

Submission Requirements

1. Kaggle Submission: Submit your final model predictions on Kaggle before the closing deadline of **Sunday, 11/23, 11:55pm.**
2. Jupyter Notebook Submission: Submit a Jupyter Notebook or a GitHub repo, with all code used for the project, through Gradescope, by **Tuesday, 11/25, 11:59pm**.

Grading

Your project grade will be based on both your performance on the competition leaderboard and your submitted Jupyter Notebook.

1. **Leaderboard Performance (10% of the grade)**
    - There will be a linear mapping from your final rank on the leaderboard to a number of points between 10 and 0.
    - The top-ranking team will earn 10 leaderboard points.
    - The lowest-ranked team will earn 0 leaderboard points.
    - All other teams will earn a score that is linearly spaced between 10 and 0 based on their leaderboard rank.
    - Number of leaderboard points earned will be calculated using this formula, where N is the total number of teams:

$$\text{Leaderboard Points} = 10 \times \frac{N - \text{Rank}}{N - 1}$$

    - Example of points earned based on leaderboard rank, if N=100:

Rank	Points earned
1	10
5	9.18
10	8.16
25	5.10
40	2.04
50	

2. **Jupyter Notebook (90% of the grade)**
   You will not be graded solely on your model's performance. More important for this project is your methodology. What is the process you used, and what things did you do to try to improve the performance of your model(s), even if those things maybe did not ultimately move you up the leaderboard. The report should be clear, well-organized, and effectively communicate the "story" of your modeling - what you did, why you did it, whether or not it worked, and what you tried next. **It should display your entire process, not just the final model.** It should convey all of the effort you put into the project.

   It will be evaluated based on the following:

- **Data Cleaning (20 points)**
  - How did you handle missing values, various data types, outliers, and other data quality issues? Was there thoughtful justification of data cleaning choices based on exploratory data analysis, model performance improvements, or domain knowledge? Does the code correctly implement the described techniques?
- **Data Exploration (20 points)**
  - Did you thoroughly examine the dataset to understand its structure, distributions, and key relationships? Did you use data exploration to identify data quality issues? Was it used to gain a better understanding of the dataset before building predictive models? Did thoughtful data exploration help guide feature selection, engineering, and modeling decisions?
- **Feature Engineering (20 points)**
  - Did you experiment with engineering features in a way that could potentially improve model results? Did you apply feature transformation, feature creation, feature selection, or useful/logical dimensionality reduction that could potentially enhance predictive performance? Was there thoughtful justification of feature engineering choices based on exploratory data analysis, model performance improvements, or domain knowledge? Does the code correctly implement the described techniques?
- **Modeling Approach (20 points)**
  - Were correct machine learning and data modeling techniques used? Was effort put into making iterative improvements through model comparisons, experimentation, hyperparameter tuning/regularization, ensembling, or any other ML techniques. Are models compared in a correct and reasonable way? Does the code correctly implement the machine learning techniques?
- **Quality & Clarity of Code and Notebook (20 points)**
  - Is the Jupyter Notebook, including the code itself, well-organized and well-written? Does it provide a clear explanation of the team's process and efforts? Does it effectively communicate the team's project and process? Are Markdown cells used to explain reasoning and process? Is the code clean, commented, and easy to follow? Can someone else run the notebook from start to finish without errors? Are there repeated blocks of code that should be cleaned up/consolidated? Does code demonstrate thoughtful and maintainable coding practices? Can reviewers clearly understand your workflow by reading through your code and notebook?
- **Peer Evaluation (at instructor's discretion)**
  - Each team member will assess the contribution of the other members of their team. Peer evaluations will be used to adjust any individual's final project grade at the instructor's discretion. Any group member who does not participate will receive a 0 for their project grade.

Project/Competition Rules

- **Each team will be limited to a maximum of 3 submissions per day.** This means that you will want to evaluate your models on your own first, to make sure you want to use one of your daily submissions on it.
- Your final leaderboard entry will be your team's best scoring submission. So if you submit one that is worse scoring than a previous submission, don't worry, the leaderboard will hold on to the best scoring submission from your team. However, you'll want to remember which model generated each submission so that you can correspond your submission scores back to your code!
- You may not share code or strategies with other teams. Remember that your final grade partially depends on your position on the leaderboard. Sharing your techniques with other teams could allow them to surpass you, lowering your own grade.
- Any attempt to manipulate the leaderboard or the competition will result in a 0 (e.g., multiple accounts, manual adjustments to outputs/predictions, use of external data, working with anyone outside of this class, etc.)
- You may use generative AI for this project. Based on the code I received last semester, here is what I suggest:
    - Use AI as an efficiency tool, not a replacement for thinking. Think about how YOU would approach the problem, then ask AI to help you write that code. Rather than asking AI how it would approach the problem - it will give you back the most generic/average solution.
    - Always read, understand, and adapt AI-generated code. Blindly copy/pasting AI code can introduce errors and result in projects that make it appear as though you do not correctly understand the concepts.
    - Do not leave repetitive copy/pasted AI code blocks in your final submission. Clean up your code and use functions or Jupyter Notebook cells to eliminate repetitive code.
- **You are limited to only the ML techniques we have covered in this class**.
    - Use the Python libraries (i.e. scikit-learn, etc.). I am ok with you using any Python or ML libraries you want, so long as it is concepts we covered or discussed in this class.
    - I am ok with you using other neural net libraries if you want (such as TensorFlow, PyTorch, Keras, etc.), so long as you are only implementing a feed-forward multi-layer perceptron.
    - We covered boosting. I am ok with you using any/all types of boosting.
    - When in doubt, ask me first.

Join the Competition: https://www.kaggle.com/t/da65371d78f34c56a3bc51e7ca27d59c

Each individual must make a Kaggle account and join the Kaggle competition individually. Then, one member of the team can create the team (on the 'Team' tab of the competition), and can invite their other members to join the team. (Fun team names encouraged :)

Good luck and have fun!

Some resources that may be useful:

- Frequency encoding for categorical features
- ML Performance Improvement Cheat Sheet
- Weight regularization for neural nets: Weight regularization is a technique used to prevent overfitting by penalizing larger weights and preferring smaller weights.
    - Explanation of weight regularization
    - To implement L2 regularization with scikit-learn MLPRegressor, use the 'alpha' parameter. (scikit-learn MLPRegressor does not support L1 regularization, but it is supported in other neural net libraries like PyTorch, Keras, and Tensorflow)
- Dropout for neural nets: Dropout is another regularization technique for neural networks that randomly deactivates various neurons during training to prevent overfitting and potentially improve generalization.
    - Explanation of dropout
    - Explanation of dropout + how to implement dropout with scikit-learn (note that dropout is more directly integrated into PyTorch, Keras, and Tensorflow)
    - Andrew Ng's videos explaining dropout