Machine Learning Foundations
Shallow Models Training, Validation and Tuning

March 26, 2025

1 Description

Welcome to Assignment II! In this challenge, you will build upon our established EDA and feature engineering skills while utilizing the new modeling tools you have acquired to predict the number of daily bike rentals for a bike-sharing system. Our predictions will be based on weather, seasonal, and temporal features using three regression algorithms of increasing complexity. You will conduct feature engineering, tune hyperparameters, compare model performance, and select our best model with maximum predictive accuracy and minimal bias.

2 Objectives

This assignment allows us to apply supervised learning to a real-world regression problem. You learn to frame a regression problem using a real-life dataset and to understand the business value of predictive modeling in urban transportation systems. The main objective is to obtain the best possible prediction for the given problem. By completing this assignment, you will use three shallow models of increasing complexity to solve a real-life prediction problem with supervised learning. You will define and solve a regression problem, using real-world data and apply machine learning models to predict continuous outcomes (bike rental counts).

3 Dataset

The Bike Sharing Dataset can be downloaded from the UCI Machine Learning Repository. Inside the zipped file, you have two files. **you must use the** `hour.csv` dataset.

In the README, you read that bike-sharing rental highly correlates with environmental and seasonal settings. For example, among other variables, weather conditions, precipitation, day of week, season, and hour of the day can affect rental behaviors. The core data set is related to the two-year historical log corresponding to 2011 and 2012 from the Capital Bikeshare system, Washington D.C., USA. The authors aggregated the data hourly and daily and then extracted and added the corresponding weather and seasonal information.

The `hour.csv` dataset contains 17379 records with the variables listed in Table 1.

For this assignment, you solve the following ML problem: **Prediction of bike rental count hourly based on the environmental and seasonal settings.**

4 Instructions

Code the following tasks in Python using a Jupyter Notebook. Use the GitHub (GH) repository you created for this course and push your notebook to that repository as you did with Assignment I. Name your Jupyter Notebook using the following naming template: `assignment_2_<name_surname[_surname]>.ipynb`. You should not push your notebook only at the end of the assignment, right before submitting it. Instead, get used to pulling and going to your GH repository while working on the assignment. Add the link to your GH repository **at the beginning** of the notebook. Upload a PDF version of your **tested** notebook on Blackboard, which includes code, cell output, and text.

Focus on analytical instead of qualitative reasoning, and explain your approach, the reasoning for your choices, and the conceptual/formal tools you employed. While you must perform EDA and feature engineering, you do not have to compare different methodologies explicitly as you did in Assignment I. Instead, you must focus on

Table 1: Variables Table – UCI Bike Sharing Dataset

Variable Name	Role	Type	Description	Units	Missing Values
instant	ID	Integer	Record index	-	no
dteday	Feature	Date	Date	-	no
season	Feature	Categorical	1: winter, 2: spring, 3: summer, 4: fall	-	no
yr	Feature	Categorical	Year (0: 2011, 1: 2012)	-	no
mnth	Feature	Categorical	Month (1 to 12)	-	no
hr	Feature	Categorical	Hour (0 to 23)	-	no
holiday	Feature	Binary	Whether the day is a holiday (extracted from the official calendar)	-	no
weekday	Feature	Categorical	Day of the week	-	no
workingday	Feature	Binary	If the day is neither weekend nor holiday: 1, otherwise: 0	-	no
weathersit	Feature	Categorical	1: Clear, Few clouds, Partly cloudy, 2: Mist + cloudy, 3: Light snow/rain	-	no
temp	Feature	Continuous	Normalized temperature in Celsius, range: [–8, +39]	C	no
atemp	Feature	Continuous	Normalized feeling temperature in Celsius, range: [–16, +50]	C	no
hum	Feature	Continuous	Normalized humidity, values divided by 100	-	no
windspeed	Feature	Continuous	Normalized wind speed, values divided by 67	-	no
casual	Other	Integer	Count of casual users	-	no
registered	Other	Integer	Count of registered users	-	no
cnt	Target	Integer	Count of total rental bikes including both casual and registered users	-	no

those EDA and feature engineering methodologies that lead to the best-performing model. You will have to design, train, evaluate, and tune three shallow models: Linear Regression (baseline model), Random Forest, and Gradient Boosting (Note: **no neural networks in this assignment 8-)** You must explain the iterative process you adopted to evaluate and tune your models, and whether you refined your EDA/feature engineering to achieve better performance. Consistently, you must also explain why your EDA and feature engineering were the best for each of the three models. You must detail your training, evaluation, and tuning process, describing in detail your analytical/qualitative reasoning, the choice you made, the expected results, and the actual results. Finally, you have to compare the three models and explain why one performed better than the others.

The order of the tasks is indicative and supports an iterative development cycle. You are expected to order the tasks consistent with the formal requirements of your analysis and loop back to earlier tasks if evaluation or tuning reveals issues in feature engineering or model design. Each task lists the relevant lectures and slides to support your implementation.

Task 1: Exploratory Data Analysis (EDA)

Lecture material: Lecture 6 (slides 4–16), Lecture 7 (slides 3–9), Lecture 8 (slides 2–5)

- Load the `hour.csv` dataset.
- Examine the target variable (`cnt`) distribution and identify its skewness.
- Explore analytically the influence of temporal (`hr`, `weekday`, `mnth`, `season`), binary (`holiday`, `workingday`), and weather-related features (`temp`, `atemp`, `hum`, `windspeed`, `weathersit`) on `cnt`.
- Visualize relationships using, for example, scatter plots, box plots, and line plots grouped by hour, day, and season, or any other analysis/plot you deem necessary.
- Identify any suspicious patterns, outliers, or anomalies.
- Consider dropping the columns `instant`, `dteday`, `casual`, and `registered`.

Remember: Use `hour.csv` from the UCI Bike Sharing Dataset.

Task 2: Data Splitting

Lecture Material: Lecture 8 (slide 4), Lecture 12 (slides 4–5)

- Split the dataset into: Training set (60%), Validation set (20%), Test set (20%). If necessary, reevaluate these percentages when tuning the model.
- Use a random split while preserving temporal order if possible.
- Apply the split before performing any feature engineering or scaling to avoid leakage.

Task 3: Feature Engineering

Lecture material: Lecture 9 (slides 2–8), Lecture 12 (slides 4–9).

- Encode cyclical features (`hr`, `weekday`) using sine and cosine transforms.
- One-hot encode categorical variables: `season`, `weathersit`, and `mnth`.
- Apply scaling (e.g., `StandardScaler`) to continuous features: `temp`, `atemp`, `hum`, and `windspeed`.
- Fit all transformations using only the training set, and apply them to validation and test sets.
- Consider interaction terms such as `temp × humidity` if they are justified by EDA.
- Remove leaky or redundant features (e.g., `atemp` if highly collinear with temp).

Task 4: Baseline Model – Linear Regression

Lecture material: Lecture 9 (slides 4–7), Lecture 11 (slides 2–4).

- Train a Linear Regression model.
- Evaluate on the validation set using, at least, Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 Score.
- Plot at least the residuals and analyze their distribution.
- Reflect on bias and variance characteristics of this model.

Note: Make the training and evaluation of the models as uniform as possible to enable proper comparison. Use the same features to train all the models (before refinement and tuning) and use at least Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 Score.

Task 5: Random Forest Regressor - Model Specification and Training

Lecture material: Lecture 11 (slides 5–7), Lecture 12 (slides 4–5).

- Train a Random Forest Regressor.
- Use default or initial parameters (e.g., 100 trees, no depth limit) to establish a baseline.
- Evaluate using the same metrics as above.
- Compare with the baseline model and explain observed differences.
- Include at least a feature importance plot and comment on top predictors.

Task 6: Gradient Boosting Regressor - Model Specification and Training

Lecture material: Lecture 12 (slides 4–7), Class 10 Training Notebook.

- Train a Gradient Boosting Regressor (e.g., XGBoost or LightGBM).
- Use basic parameters to establish initial results.
- Plot at least residuals and compare performance with previous models.
- Note any early signs of overfitting or high variance.

Task 7: Hyperparameter Tuning

Lecture material: Lecture 12 (slides 6–9), Class 10 Training Notebook.

- Tune the Random Forest Regressor:
  - Use Randomized Search CV with 5-fold cross-validation.
  - Tune the following hyperparameters: `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`.
  - Report: best parameter combination, validation performance, updated feature importance.
- Tune the Gradient Boosting Regressor:
  - Use Bayesian Optimization (e.g., via `BayesSearchCV`).
  - Tune: `learning_rate`, `n_estimators`, `max_depth`, `subsample`.
  - Visualize convergence of the optimizer if possible.
  - Report: best parameters, cross-validated performance, impact of tuning on generalization.
- Explain whether tuning significantly improved performance or not, and hypothesize why (e.g., model variance, overfitting, flat loss surface, etc.).

Note: Compare pre- and post-tuning performance. Highlight overfitting, underfitting, or convergence issues. This task is part of a broader iterative loop — feel free to return to earlier tasks if the results are suboptimal.

Task 8: Iterative Evaluation and Refinement

Lecture material: Lecture 9 (slides 6–7), Lecture 11 (slides 3–4), Lecture 12 (slide 9).

- Based on model results, revisit EDA and feature engineering if needed. For example:
  - Do new interaction terms help?
  - Should you drop or transform certain features?
  - Are there outliers that are harming performance?
- Model tuning and evaluation (Tasks 4–8) are iterative. Based on performance, you may revisit Task 3 (feature engineering), add new transformations, and adjust model complexity. Document your iterations and reasoning thoroughly in the notebook.
- Retrain, re-evaluate and re-tune your models as needed.

Task 9: Final Model Selection and Testing

Lecture material: Lecture 12 (slides 8–9).

- Select the best-performing model based on validation performance and bias/variance tradeoffs.
- Retrain it on the combined training + validation set.
- Evaluate on the test set using MSE, MAE, and R^2.
- Report the final metrics and justify why the chosen model generalizes best.

5 Deliverables

A Jupyter Notebook containing code and a detailed explanation of the models you used and how they performed, how you engineered and selected features, how you tuned and evaluated models, and why your final model is the best. The explanation should equate to a standalone document of at least two pages.

6 Evaluation Criteria

- **Correct Implementation of Tasks** (40%): Ensure that all steps are properly executed.
- **Explanation and Rationale** (30%): Justify your specific choices and explain the methods, techniques, and parameter selections that informed those decisions.
- **Code Organization and Cleanliness** (10%): Your code should be modular, include meaningful and informative comments (for instance, the comment 'this is a variable definition' is not helpful), and properly utilize GitHub.
- **Depth of Analysis** (20%): Evaluate how thoroughly you explored and statistically interpreted results instead of merely applying techniques without understanding them and their results.

WARNING: Code that fails to compile will be deemed invalid and contribute 0 to the overall grade, regardless of how trivial the error may be. **You must test the entire notebook before submission**. To perform a complete test, execute 'Restart the kernel and run all cells' right before submitting the assignment and ensure that every cell in the notebook compiles successfully. One last thing. . . Did I mention that you must test the entire notebook before submission?!