Review a peer's work(Group 2) and provide feedback

1.GitHub repo review

1.1 Strengths we noticed in Group 2 repo

Group 2's GitHub repo gives a clear sense of steady progress across the semester. The folder list at the top level already signals the main parts of the workflow, such as data, notebooks, model artifacts, documents, and outputs, which helps a reviewer quickly understand what to click first. The documents folder is especially helpful because it stores weekly reports as PDFs, so it is easy to track how the project evolved over time. We also liked that the model folder includes saved artifacts like "joblib" and "pkl" files and a final predictions csv, since it shows the team did not stop at experimentation but actually kept the key deliverables needed for reuse. In the notebook folder, the week based files make it easy to locate what was done in each stage, and the presence of exported html files can be convenient for quick viewing without running the notebooks. Overall, the repo feels complete and realistic for a class project, with evidence of documentation, modeling, and outputs rather than only raw code.

1.2 Suggestions to make it easier to read and reuse

One clear improvement is the README setup. Right now, README files appear inside individual folders, but the main repo page still shows the Add a README prompt, which makes the project harder to understand at first glance. A short but informative top level README would help a lot. It could explain the goal in two or three sentences, list what each folder is for, and provide a simple how to run section, including where to place raw data, what notebooks to run in what order, and what outputs to expect. Some folder READMEs are very short, like a placeholder line in the data folder, so expanding them would make the repo more self explanatory for someone outside the team.

We also noticed the code and notebooks are quite spread out. The week based organization is great for class submissions, but for a new reader it can be harder to see the full end to end pipeline. One idea is to add a single entry point notebook or a short run guide that links to the key notebooks in order, such as data prep, feature engineering, modeling, evaluation, and final predictions. Another small cleanup would be to reduce extra files that do not help reviewers, such as ipynb checkpoints folders or tool generated folders like catboost info, and add them to gitignore. Finally, it would help to include a requirements file or environment notes so others can

reproduce the run without guessing package versions. These are small changes, but they would make an already solid repo much easier to navigate and reuse.

2.Code Review

Below are our team's observations and suggestions regarding Group 2's code section. From an MDLC perspective, we believe Group 2's Week 13 project demonstrates a high level of overall completion, particularly excelling in its end-to-end workflow. Their code does not merely piece together previous weeks' content but clearly presents a complete pipeline—from data acquisition, EDA, feature engineering, model comparison, to final model training and result export. This approach aligns closely with the core principles of MDLC and brings the project significantly closer to a production-ready version.

Regarding the data layer, we found their approach particularly strong. First, they employed a station-level data splitting method instead of simple random splitting, which is crucial for demand forecasting problems as it effectively prevents data leakage. Second, their feature engineering incorporates station-specific temperature normalization instead of global standardization. This demonstrates an understanding of operational context: usage patterns and environmental conditions inherently vary across stations. Furthermore, they mitigate bias from differing station activity levels through sample weighting—a classic data center AI approach. This shows their modeling prioritizes fairness and stability alongside prediction accuracy.

The logical progression from EDA to subsequent modeling is also well-defined. For instance, they identified distinct commuting peaks during EDA and directly converted these into a rush hour indicator during feature engineering. Upon discovering a significant right skew in ride durations, they applied truncation to the target variable. The high proportion of membership users also provided a reasonable explanation for the overall stability of demand. This chain of analysis, feature processing, and modeling is coherent.

We also identify areas for improvement in this project. First, reviewing the Week 1 repository, we observe a noticeable shift in focus during later stages. By Week 13, the project leaned more toward engineering-driven demand forecasting, with diminished attention to user behavior variations and interpretive analysis. The final model primarily predicts highly aggregated ride_count metrics rather than returning to the original objective of “understanding demand

change mechanisms.” We recommend explaining this shift in objectives within the final report to prevent readers from perceiving inconsistency in research focus.

Regarding model selection and result interpretation, the rationale for choosing Decision Tree as the final model feels underdeveloped. Particularly concerning is the summary's direct assertion of “production-ready” status despite negative R^2 values on both validation and test sets, which feels logically disconnected. Clarifying that the model optimization objective was RMSE or MAE rather than variance would enhance overall persuasiveness.

Additionally, Week 13's EDA utilized raw trip-level data, while modeling employed highly aggregated demand data. Although this process is documented elsewhere in the code, the data transformation logic between these stages is not fully presented within the same pipeline, making it less accessible for subsequent code reviewers.

Overall, we believe Group 2's project significantly exceeds the course average in engineering completeness, data processing rigor, and reproducibility, representing a case study remarkably close to real-world industrial scenarios. Incorporating explanations in the areas mentioned above would make this project even clearer and more accessible.

3.Code output review

This section focuses specifically on the outputs generated by Group 2’s code and how effectively those outputs support interpretation, validation, and reuse of the modeling results.

One clear strength of Group 2’s code outputs is that they are structured to support end to end verification of the modeling process. The pipeline produces concrete, inspectable artifacts at multiple stages, including intermediate evaluation results, final predictions, and summary statistics. The inclusion of exported CSV files for model comparison and final test predictions allows reviewers to independently inspect model behavior without rerunning the full pipeline, which is particularly valuable given the large dataset size. This design choice reflects an awareness that code outputs should serve not only as internal checkpoints but also as communicable results.

The model comparison output is especially helpful in understanding the team’s decision making process. By evaluating multiple models under the same sampled validation setup and exporting

the comparison table, the team makes their selection process transparent. The comparison clearly highlights trade offs in training time and predictive performance across linear regression, random forest, and decision tree models. From a peer perspective, this output makes it easier to understand why simpler tree based methods were ultimately favored and demonstrates that the final model choice was based on empirical evidence rather than convenience.

The final evaluation outputs further reinforce this transparency. Reporting RMSE, MAE, and R squared on validation and test sets provides a multi dimensional view of performance. While some metrics indicate limitations in variance explanation, the consistency of RMSE and MAE across splits helps establish that the model behaves in a stable and predictable manner. Framing the error in terms of ride counts per station per time interval also helps contextualize these numbers, making the outputs more interpretable from an operational standpoint rather than purely a statistical one.

Another positive aspect of the code output is its alignment with data centric modeling goals. The pipeline explicitly documents steps such as leakage prevention, station level splits, and per station normalization, and these choices are reflected in the resulting metrics. The improvement from earlier iterations to the final evaluation outputs suggests that data refinement, rather than increased model complexity, drove performance gains. This makes the outputs more credible, as improvements can be traced back to concrete changes in preprocessing rather than opaque tuning.

The fairness related outputs are also a valuable addition. By incorporating sample weighting and evaluating performance across station activity levels, the team produces results that go beyond aggregate accuracy. Even though the fairness metrics themselves are not deeply visualized in the final outputs, the presence of weighting logic and activity based evaluation signals that the outputs were generated with distributional performance in mind. This adds an important dimension to the results and makes them more relevant for real world deployment scenarios.

There are, however, several areas where the code outputs could be made clearer or more informative. First, while the pipeline exports final predictions, it is not immediately obvious how these predictions should be interpreted or used downstream without referring back to the code. Adding a short README or metadata file describing the schema of the output files, the time

granularity, and the intended interpretation of each column would significantly improve usability for external reviewers.

Second, although the model comparison output is useful, it relies on a sampled validation set for efficiency. While this choice is reasonable, it may leave readers uncertain about how representative these comparisons are relative to the full dataset. Including a brief note in the output logs or accompanying documentation explaining the rationale for sampling, and how it compares to full scale evaluation, would make the results easier to contextualize.

Third, the outputs mix information derived from raw trip level data and highly aggregated demand data. While both are valid, the transition between these representations is not fully visible in the outputs themselves. As a result, a reviewer who only inspects the final artifacts may struggle to trace how raw observations ultimately map to predicted demand values. Providing an intermediate summary output that documents aggregation logic, time bins, and station filtering would help bridge this gap.

Finally, the labeling of the final pipeline as “production ready” in the output summary could benefit from additional qualification. From an engineering perspective, the pipeline is robust and reproducible, but some evaluation metrics suggest room for improvement in predictive performance. Clarifying that the model is production ready in terms of structure, reproducibility, and monitoring readiness, rather than optimal accuracy, would make the outputs more precise and defensible.

Overall, Group 2’s code outputs demonstrate a strong emphasis on transparency, reproducibility, and practical usability. The generated artifacts make it possible to audit model performance, understand key trade offs, and reuse results without rerunning the full workflow. With modest improvements in documentation and clearer contextualization of outputs, this project’s results would be even more accessible to future reviewers and users. As it stands, the outputs reflect a mature and thoughtful approach to delivering machine learning results in an applied setting.

4. project report review

For their project report, the team takes on a practical and well defined problem. NYC’s bike share system still lacks proactive, station level demand forecasting, which means rebalancing

decisions often end up reactive and uneven. From the opening, the team framed the operational gap clearly. Dispatch and rebalancing can be inconsistent, and the system frequently runs into empty docks or full stations. They tied this gap to real consequences in a grounded way, including missed trips, frustrated riders, and service quality that can vary across neighborhoods. This framing made the motivation easy to follow and kept the work anchored in day to day operations rather than abstract model building.

Another strength was that the team was explicit about who the tool is for. Instead of treating the project as a leaderboard style modeling exercise, they consistently returned to the actual users, including riders, operations staff, and city planners. Their explanation of how hourly forecasts could support morning deployment, evening rebalancing, and longer term planning made it clear that the system is intended to function as decision support. This user centered focus helped the project feel relevant and made the potential impact easier to imagine.

In terms of the model development lifecycle, the workflow was structured and sensible. The team spent appropriate time on the data foundation, explaining how they combined multi year trip data with station metadata and hourly weather information. Importantly, they did not gloss over the messier parts of the process. They acknowledged schema inconsistencies, station ID complications, and the realities of working with large scale data. Building a canonical station hour table, including periods with zero rides, was a strong design choice and shows an understanding that demand modeling cannot rely only on observed trips without introducing bias.

Feature engineering was another highlight. The temporal and weather driven features made intuitive sense, and station identity and flow based features were linked back to patterns observed during EDA rather than added mechanically. Using feels like temperature instead of stacking multiple correlated weather variables was a practical decision that likely improved stability and reduced multicollinearity. Their treatment of lag features also reflected good judgment. They explored them initially and later removed them once leakage concerns became clearer. This willingness to revise earlier choices made the overall process feel careful and credible.

Their model selection narrative was straightforward and convincing. The team tested multiple model families and ultimately chose a regularized decision tree, arguing that interpretability and stability were more valuable than marginal accuracy gains from complex ensemble methods.

This choice matched the use case, as operations teams often need models that behave consistently and can be explained. The results also supported the conclusion that bike demand follows strong structure and patterns but is not highly nonlinear. The comparison between the Week 9 model and the final version showed meaningful progress driven by better data design and feature decisions rather than increasing model complexity.

On evaluation, the team communicated metrics clearly and kept them interpretable. Performance consistency across training, validation, and test sets suggests reasonable generalization. More importantly, framing error in terms of rides per station per hour aligns well with how operations teams think about demand. This translation from model metrics to operational meaning made it easier to judge whether the performance level is usable in practice.

The discussion of bias and fairness was one of the strongest aspects of the project. By breaking down errors across different station activity levels, the team showed awareness that overall averages can hide uneven performance. Their weighting approach to address imbalance was easy to follow, and they clearly explained the trade off between headline accuracy and more equitable system wide performance. The ethics discussion felt integrated into the workflow rather than added at the end, which is appropriate for a public facing system.

There were still a few areas where the presentation could be tightened. At times, particularly during data processing and feature construction, the level of technical detail slowed the narrative. Emphasizing major design decisions and their impact, rather than every pipeline step, could improve flow. In addition, while deployment and monitoring ideas were conceptually sound, the section would benefit from a more concrete example, such as a simple dashboard sketch or an example workflow showing how operations teams would use the forecasts.

Overall, this was a strong and thoughtfully executed project that aligns well with the course goals. The team demonstrated solid technical skills, effective collaboration, and a consistent ability to connect modeling decisions to operational needs. With slightly tighter editing and a more tangible deployment narrative, the work could be even more persuasive. As it stands, the project reflects a mature understanding of the model development lifecycle and a practical, responsible approach to building a machine learning system intended to support real decisions.