Supervised Learning Report
CS7641: Machine Learning
Fall 2025

1 Assignment Weight

The assignment is worth 12.5% of the total points.

Read everything below carefully as this assignment has changed term-over-term.

2 Objective

The purpose of this project is to explore techniques in supervised learning. It is important to realize that understanding an algorithm or technique requires understanding how it behaves empirically under a variety of circumstances. As such, rather than implement each of the algorithms, you will be asked to experiment with them and compare their performance. This is quite involved and also possibly quite different from what you are used to; however, it is central and in many ways the essence of supervised learning!

3 Procedure

You are given two vastly different datasets. You will design **exactly two** supervised learning tasks in total, **one per dataset**. Each task may be either classification or regression, your choice! Do what is more interesting to you as you explore the data. For this assignment, a *classification* problem predicts a discrete label (e.g., `is_canceled` $\in \{0, 1\}$ or `Severity` $\in \{1, 2, 3, 4\}$), whereas a *regression* problem predicts a continuous numeric target (e.g., `adr`, `lead_time`, or incident duration `End_Time` $-$ `Start_Time`). For **each** dataset, you must define a training set and a held-out test set and choose appropriate evaluation metrics (e.g., accuracy/F1 for classification; MAE/MSE for regression). You will need to explain *why* the datasets are interesting from an ML practitioner perspective and be able to discuss context with a deeper understanding of the datasets.

You will go through the process of exploring the data, develop hypotheses, tune algorithms, and write a thorough analysis of your findings. You need not implement any learning algorithm yourself, however, you must participate in the journey of exploring, tuning, and analyzing. Concretely, this means:

- You may program in any language you wish and are allowed to use any library, **as long as it was not written specifically to solve this assignment**. Code repositories, notebooks, or AutoML templates created for CS7641 assignments ("plug-and-play" solutions) are **not allowed**. With many ML-friendly languages available, I recommend Python for consistency. However, we have plenty of support for other languages.
- TAs must be able to recreate your experiments on a standard Linux machine if necessary.
- The analysis you provide in the report is paramount and the focus of the assignment.

You should experiment with these four learning algorithms on each dataset. They are:

- **Decision Trees**. Use some form of pruning (e.g., post-pruning via *ccp_α* or depth/leaf limits). You are not required to use information gain. Gini or entropy are fine. Please **state what you use and why**.
- **k-Nearest Neighbors**. Compare **meaningfully different** values of k (e.g., small/medium/large values) and justify your choice.

- **Support Vector Machines**. Evaluate $\geq$**2 kernels** (e.g., linear vs. RBF) and tune C and γ (where relevant).
- **Neural Networks**. Investigating capacity scaling (depth vs. width), while keeping SGD constant. Hold the total parameter count approximately constant and compare *deeper–narrower* vs. *shallower–wider* MLPs (e.g., depth $\in \{2, 3, 4, 6\}$ with adjusted widths). Optimize with **SGD only** (*no momentum, no Nesterov, no adaptive variants such as Adam/Adagrad/RMSprop*). You will want to report learning rate, batch size, and other metrics you feel will help your analysis or explanatrion. For classification, use a sigmoid/softmax head with cross-entropy, and for regression, use a linear head with MSE/MAE.

Each algorithm is described in detail in your textbook, the assigned readings on Canvas, and on the internet. Instead of implementing the algorithms yourself, you should use libraries that do this for you and make sure to provide proper attribution. Also, note that you'll need to do some tinkering to obtain good results and graphs, and this might require you to modify these libraries in various ways. **If you use any large language models (LLMs) at any stage (e.g., for code suggestions, debugging, or code for figure generation), you must disclose and cite them**, include the model/provider (e.g., ChatGPT/OpenAI, Claude/Anthropic), version or access date, and a brief description of how it was used; provide an in-text citation at the end of your paper. To be very clear, you need to have a citation in your references if an LLM was used. You should not use an LLM to generate your text. Please reach out if you have any questions.

Extra Credit Opportunity (up to 5 points; NN only; *SGD only*).

- **Activation function study (one dataset only).** On exactly *one* dataset of your choice (Hotel Booking *or* US Accidents), evaluate **four** activation functions (e.g., ReLU, GELU, SiLU/Swish, tanh) using the *same* architecture, preprocessing, regularization, and training protocol. Use **SGD only**. *Deliverables:* epoch-wise validation curves for each activation, a comparison table of final metrics, and a brief analysis of convergence speed/stability and any observed trade-offs under SGD.

4 What is required as you go through your analysis

Applies to both datasets, for each of the four algorithms (NN, SVM, kNN, DT).

Data & targets (per dataset)

- Declare targets (classification or regression) and justify metrics for that target.
- If imbalanced, report PR-AUC alongside ROC-AUC/F1 and give the **prevalence baseline** for PR-AUC; calibrate probabilities if you threshold.
- State leakage controls you applied (e.g., Hotel: drop `reservation_status`/`reservation_status_date` for cancellation; Accidents: drop post-outcome fields like `End Time`, post-dated `Weather_Timestamp` for severity-at-onset).

Methodology & splits

- One held-out test used once at the end; tuning via CV/hold-out on the train split only.
- Set all random seeds and record exact splits.

Per-algorithm required figures/tables

- **Learning curves (LC):** training and validation metric vs. training size. Diagnose bias/variance.
- **Validation / model-complexity curves (MC):** validation metric vs. one hyperparameter (e.g., NN width or L2; SVM C/γ; kNN k; DT depth/`ccp_alpha`). Include the training curve too if space allows.
- **Runtime table:** fit and predict wall-clock times with a brief hardware note.
- **Classification add-ons:** ROC & PR curves; confusion matrix at a justified threshold; probability calibration check (reliability plot) if you present probabilities.
- **Regression add-ons:** residuals vs. prediction/feature; MAE/MedAE/MSE as appropriate.

Algorithm-specific essentials

- **NN:** show iteration-based learning curves (epoch vs. loss/metric) and discuss early stopping/regularization; report chosen activation(s).
- **SVM:** scale features, use $\geq$2 kernels per dataset, tune C and kernel parameters; calibrate if you report probabilities.
- **kNN:** scale features, justify distance/weighting; note sensitivity to irrelevant features and high dimensionality.
- **DT:** show a pruning path/validation curve and report final depth/leaves.

Cross-model comparison

- Compare across algorithms and datasets with evidence-backed trade-offs (metric vs. wall-clock; stability vs. capacity; sensitivity to hyperparameters/features). Tie conclusions to the course concepts.

Conclusion

- Explicitly discuss Type I/II in the context of your operating point (classification) or residuals (regression).
- Name one realistic next step.

5 Experiments and Analysis

External sources & literature (required)

- Include outside sources beyond course materials, with at least **two peer-reviewed references**.
- Use these to **justify choices** (metrics for imbalance, threshold selection, calibration, pruning, kernel/activation decisions, leakage controls, spatiotemporal validation) and to **interpret results** (e.g., why PR-AUC is preferred, why pruning reduces variance).
- Acceptable styles: **MLA, APA, or IEEE**, pick one and be consistent in-text and in the bibliography.

Interpret, don't summarize

- Every figure/table needs takeaway tied to algorithm behavior & data. Avoid repeating the legend; explain what the figure/table *means*.

Analysis writeup is limited to 8 pages.

The page limit includes your citations. Citations must be in IEEE, MLA, or APA format. Anything past 8 pages will not be read. Please keep your analysis concise while still covering the requirements of the assignment. As a final check during your submission process, download the submission to double-check everything looks correct on Canvas. Try not to wait until the last minute to submit as you will only be tempting Murphy's Law.

In addition, your report must be written in LaTeX on Overleaf.

You can create an account with your Georgia Tech email (e.g., `gburdell3@gatech.edu`). When submitting your report, you are required to include a **READ-ONLY** link to the Overleaf Project. If a link is not provided in the report or Canvas submission comment, 5 points will be deducted from your score. Do not share the project directly with the Instructor or TAs via email. For a starting template, please use the IEEE Conference template.

6 Updated Datasets for Fall 2025

The following datasets are required for the Fall 2025 cohort. Each semester these datasets may change to avoid overused "toy" data. These are small- to mid-sized, real-world datasets that support rigorous analysis. Treat each dataset *separately* so you can compare algorithmic behavior across distinct domains. If access to the original download is limited, copies are posted on Canvas. If these datasets are not used, you will receive a zero for the assignment.

- **Hotel Booking Demand (H1/H2).** Kaggle Repository: *Hotel booking demand*.
  Source/paper: N. C. António, A. Almeida, and L. Nunes (2019), "Hotel booking demand datasets," *Data in Brief* 22:41–49. doi:10.1016/j.dib.2018.11.126.
  Scope & size: Bookings from a city hotel (H2) and a resort hotel (H1) in Portugal; ∼119,000 rows and ∼32 features.
  Suggested targets & tasks:
  - **Classification**: `is_canceled` (cancellation prediction); `is_repeated_guest`; other classification via `assigned_room_type` vs. `reserved_room_type` (binary "upgraded?" or multiclass).
  - **Regression**: `adr` (average daily rate) and `lead_time`. Optional derived target: total nights = `stays_in_week_nights` + `stays_in_weekend_nights`.

  Design notes/pitfalls: Avoid leakage. Exclude `reservation_status` and `reservation_status_date` when predicting cancellation. Handle high-cardinality fields (`country`/`agent`/`company`) with target/frequency encoding *inside* CV. Class imbalance is common for `is_canceled`; use stratified CV and report calibrated probabilities.

- **US Accidents (since 2016).** Kaggle Repository: *US Accidents (since 2016)*.
  About: Large, countrywide traffic accident records from 49 U.S. states, enriched with weather/points-of-interest and temporal context. Licensed CC BY-NC-SA 4.0.
  Size/features: Millions of rows (versioned) with ∼47 attributes (timestamps, geolocation, distance, weather, road features, day/night, etc.).
  Suggested targets & tasks:
  - **Classification**: `Severity` (ordinal 1–4). You may treat it as multiclass classification; if you choose ordinal-aware methods, justify briefly.
  - **Regression**: incident duration (derive: `End_Time - Start_Time`) or `Distance(mi)` at onset.

  Design notes/pitfalls: For severity-at-onset models, exclude post-outcome fields (`End_Time`, post-dated `Weather_Timestamp`) to prevent leakage. Use stratified sampling for any downsampling. High-cardinality `City`/`Street`/`Zipcode` need careful encoding. Report PR/AUPRC in addition to accuracy/F1 due to imbalance.

Scale requirement (read this). The **US Accidents** dataset is intentionally large. *Prototype small, submit large.* It is fine to start with a stratified subset for EDA and coarse tuning (e.g., 5–20% of rows), but your **final experiments, figures, and cross-model comparisons should be run on at least 80% of the available rows after cleaning and leakage controls**, with the clear goal of using 100%. You must **report exact row counts and percentages** at each stage (raw → cleaned → train/test). Any subsampling must be *reproducible* (fixed seed) and *stratified* on the target (and, for US Accidents, respectful of time/geography to avoid spatiotemporal leakage). ~~If you use less than 80% in any final experiment, you must include a brief Compute Justification describing your hardware (CPU/GPU/RAM), observed runtime/memory limits, and mitigation steps attempted (e.g., efficient dtypes, chunked/streaming pipelines, sparse/ordinal encodings, linear/scalable baselines, or use of cluster/cloud resources). If we find you used fewer than 1,000,000 rows (i.e., <15% of a typical current release), we will treat that as cherry-picking and deduct -50% from the report score.~~

Update as of 09/08/25: There has been a lot of discussion on hardware capabilities over the last several days with concern to the datasets. I did preliminary data analysis before the term started and chose these datasets to help throughput learning in the SL, OL, and UL reports since we will build on methods to better understand how to handle data analysis at different scales. Both datasets are still smaller in size than you'd expect in industry, however I do understand how some of the paradigms like SVM and kNN might explode computationally on local hardware.

Since the goal is for you to explore the algorithms in a practical way rather than play hardware lottery, I have modified the requirements for data use. Please see the recommendations below. If you deviate from these requirements, you will incur a 50% deduction to the report.

Global rules (apply to all models)

- **Encoding:** Do *not* one-hot high-cardinality categoricals (e.g., City/Street/Zip). Use target/frequency encoding inside CV so SVM/kNN/NN see comparable inputs.
- **Scaling:** Put `StandardScaler` in your pipeline for all numeric features.
- **Dtypes:** Use `float32` features/weights; labels `int32`/`int64`. No `float64`.
- **CV & splits:** On the large dataset, prefer `StratifiedShuffleSplit` with a fixed number of evaluations (don't run 5×5 nested CV on millions of rows).
- **Runtime/RAM reporting:** Log wall-clock and peak RAM for each final model. If a single run exceeds $\sim$ 15 minutes or $\sim$ 12 GB RAM, reduce sample size (halve until it fits) and show the trade-off with a learning curve.
- **Learning curves:** For every algorithm on the large dataset, include score vs. training size (at least 4–5 points).
- **Repro:** Fix random seeds; report hardware.

Dataset-specific minimums

- **Smaller dataset (Hotel; $\sim 100k \times 31$):** Run all required variants (DT, kNN exact, SVM linear + kernel, NN). This is still a small enough size for all algorithms to run in a reasonable time.
- **Large dataset (US Accidents; $\sim 7M \times 46$):**
  - **DT:** use $\geq$ 1,000,000 rows with regularization (depth/leaf caps). If runtime exceeds the budget, halve the sample and justify with a learning curve.
  - **SVM (linear):** use $\geq$ 1,000,000 rows (stratified).
  - **SVM (kernel):** cap at $\leq$ 100,000 rows (stratified) and compare to linear SVM trained on 1M.
  - **kNN (exact):** cap training at $\leq$ 250,000 rows and evaluate on $\leq$ 25,000 stratified hold-out.
  - **NN:** you may use the full dataset (80%+ after cleaning) if you follow the NN compute budget below.

Decision Trees (DT; single CART)

Single Decision Tree

- **When:** Interpretable nonlinear baseline; handles mixed feature types; insensitive to feature scaling.
- **Model:** `DecisionTreeClassifier(random_state=...)` (`criterion="gini"` or `"entropy"`).
- **Pipeline:** Encoder $\rightarrow$ (optional `StandardScaler`) $\rightarrow$ `DecisionTreeClassifier`. *Use the same encoded features you use for SVM/kNN/NN for fair comparison.*
- **Tune (regularize to control depth/size):**
  - `max_depth` $\in \{6, 10, 14, 18\}$ (avoid `None` on large data).
  - `min_samples_leaf` $\in \{50, 100, 200\}$.
  - `min_samples_split` $\in \{100, 200, 400\}$.
  - `max_features` $\in$ {"sqrt", "log2", 0.5}.
  - `ccp_alpha` $\in \{0.0, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$ (cost-complexity post-pruning).
  - `class_weight="balanced"` if imbalance.
- **Large dataset hints:** Cap depth (≤ 18) and set `min_samples_leaf` ≥ 100 to keep training fast at $\geq$ 1M rows. Prefer a coarse grid and early stop if fit time exceeds budget.
- **Report:** Final tree *depth*, *#leaves*, node count; top-10 Gini importances (and optionally permutation importances on a held-out split); confusion matrix. Include a *validation curve* vs. `max_depth` and a *learning curve* vs. training size.

SVM (Support Vector Machines)

Linear SVM (large & small datasets)

- **When:** Baseline for high-n, mid-d tabular; scales well.
- **Model:** `LinearSVC` (liblinear) for small; or `SGDClassifier(loss="hinge")` for large (faster, supports class imbalance).
- **Pipeline:** `StandardScaler` → `LinearSVC`/`SGDClassifier`.
- **Tune (keep tight):**
  - `C` $\in \{0.1, 1, 10\}$ (`LinearSVC`) **or** `alpha` $\in \{10^{-5}, 10^{-4}, 10^{-3}\}$ (`SGD`).
  - `class_weight="balanced"` if imbalance.
  - `max_iter`: 5k–20k; stop when tolerance reached.
- **Diagnostics to include:** accuracy/PR-AUC (per class if imbalanced), support-vector fraction (LinearSVC proxy: margin distribution), calibration curve (optional).

Kernel SVM (RBF) (small: full; large: subsample $\leq 100k$)

- **When:** Nonlinear boundary check; do *not* run on millions of rows.
- **Model:** `SVC(kernel="rbf")`.
- **Pipeline:** `StandardScaler` → `SVC`.
- **Tune (coarse grid):** $C \in \{0.5, 2, 8\}$, $\gamma \in \{$`"scale"`$, 1/d, 2/d\}$. Stop early if runtime explodes.
- **Deliverable:** Compare to linear SVM; explain if/when nonlinearity helps.

k-NN (k-Nearest Neighbors)

Exact kNN

- **When:** Local smoothness baseline; expect runtime to grow with sample size.
- **Model:** `KNeighborsClassifier(algorithm="brute", metric="euclidean", n_jobs=-1)`.
- **Pipeline:** `StandardScaler` → `KNN`.
- **Large dataset limits:** $\leq 250k$ train, $\leq 25k$ test. Batch predictions in chunks (e.g., 2k–10k) if RAM tight.
- **Tune:** $k \in \{3, 5, 11, 21\}$, optionally *metric* $\in \{$`euclidean`, `manhattan`$\}$. Pick one after validation and stick to it across the size sweep.
- **Report:** Learning curve (score vs. train size), sensitivity to k on a fixed subset, wall-clock for `predict()`.

Neural Networks (compact MLPs with SGD)

Compute budget (must follow for both datasets)

- **Param cap:** Keep total trainable params in $0.2M - 1.0M$ range.
- **Epoch cap:** ≤ 15 epochs with early stopping (patience 2–3). Plot best-epoch marker.
- **Batch size:** 512–2048 (pick once per dataset; hold fixed across width/depth sweeps).
- **Precision:** `float32` everywhere.
- **Regularization:** L2 (10^{-4}–10^{-3}). Optional dropout (≤ 0.2) but keep consistent across runs.

Architectures (pick two to compare capacity distribution at a similar param count)

- **Shallow–wide:** e.g., $[512, 512]$.
- **Deeper–narrower:** e.g., $[256, 256, 128, 128]$.
- Keep parameter counts comparable across the two shapes; that's the point of the comparison.

Encodings & pipelines

- **Categoricals:** target/frequency encoding inside CV; no wide one-hots.
- **Numerics:** `StandardScaler` before the MLP (or normalize inside your NN input pipeline).
- **Outputs:** Use appropriate loss (logistic vs. MSE); show calibration curve if classification with skew.

Required plots

- Epoch curve with early-stopping marker (train vs. val).
- Learning curve vs. training size.
- Model-complexity curve varying one dimension (e.g., width) while staying inside the param cap.

What to hand in for every algorithm

1. Exact config (preprocessing, hyperparams, seed).
2. Data size used (train/val/test counts) and why (especially for large dataset caps).
3. Wall-clock & peak RAM for the final model.

Fast start (do this in your first work session)

1. Load & clean; cast to `float32`; build a single preprocessing pipeline shared across SVM/kNN/NN.
2. Run Linear SVM and kNN ($k = 5$) on 10k rows to sanity-check the pipeline.
3. Scale up: Linear SVM @ 1M, kNN @ 100k, RBF SVM @ 50–100k, NN with the compute budget.
4. Start logging times and RAM; begin the learning curves early so you can justify any sample caps later.

Please reach out if you have any questions!

7 Acceptable Libraries

Here are a few **examples** of acceptable libraries. You can use other libraries as long as they fulfill the conditions mentioned above.

Core ML (Python)

- **scikit-learn** — pipelines, CV, metrics, calibration.
- **imbalanced-learn** — resampling and class-weight tools for imbalanced targets.
- **scikit-learn-extra** — extra algorithms (e.g., k-medoids) when relevant.

Deep Learning (for NNs)

- **PyTorch** (*Lightning optional for cleaner loops*).
- **Keras / TensorFlow 2**.

Optimizer rule for this assignment: use SGD only (no momentum/Nesterov/Adam/Adagrad/RMSprop).

Data & Visualization

- **pandas / polars** — tabular wrangling.
- **NumPy** — arrays and linear algebra.
- **matplotlib / plotly / altair** — plotting; use plotly/altair for interactive PR/ROC/residuals.

8 Submission Details

All scored assignments are due by the time and date indicated. Here "time and date" means Eastern Time (ET). **Canvas displays times in your local time zone if your profile is set correctly; grading deadlines are enforced in ET.** Please double check your settings and assignments for the exact due dates to mark your calendars appropriately.

All assignments will be due at 11:59:00 PM ET on the final Sunday of the unit. Since we will not be looking at the assignments until morning, submissions received by **7:59:00 AM ET** the next morning are accepted **without penalty**.

Late penalty. After the grace window, the score is reduced **20 points per calendar day**. (Example: submitted Monday 9:00 AM ET $\rightarrow -20$; Tuesday 9:00 AM ET $\rightarrow -40$.) Treat **11:59 PM ET** as your real deadline; allow time for upload and verification on Canvas.

You will submit **two PDFs**:

1. **SL_Report_{GTusername}.pdf** — your report. Your document must be written in LaTeX using Overleaf.
2. **DOCSTRING_{GTusername}.pdf** — includes:
   (a) A **READ-ONLY** link to your Overleaf project.
   (b) A **GitHub commit hash** (single SHA) from the final push of your report/code.
   (c) **Exact run instructions** to reproduce results on a standard Linux machine (environment, commands, data paths/URLs, and seeds).

- Include the READ-ONLY Overleaf link in the report or Canvas submission comment. **Do not send email invitations.**
- Use the **GT Enterprise GitHub** for all course-related code. This must be the actual commit hash, not a general link.
- Provide sufficient instructions and pointers to retrieve code and data (URLs where appropriate). You need not vendor entire libraries; ensure we can reproduce your results on a standard Linux machine.

For a starting template, we recommend using the IEEE Conference template[1].

Only your **latest submission** will be graded. Please double-check that **both PDFs** are submitted.

Formal writing requirement (bullets). Your *report* is a formal technical paper. Bullet writing is not formal writing and will be treated as a draft. Use paragraphs and integrated prose. In your final report, any list environment (`itemize`, `enumerate`, or `description`) with more than **two** items in any section will incur a **50% deduction on the overall report score**. Keep enumerations to ≤ 2 items or convert them to narrative prose.

[1] `https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn`

9 Feedback Requests

When your assignment is scored, you will receive feedback explaining your errors and successes in some level of detail. This feedback is for your benefit, both on this assignment and for future assignments. It is considered a part of your learning goal to internalize this feedback. We strive to give meaningful feedback with a human interaction at scale. We have a multitude of mechanisms behind the scenes to ensure grading consistency with meaningful feedback. This can be difficult; however, sometimes feedback isn't as clear as you need. If you are confused by a piece of feedback, please start a private thread on Ed and we will help clarify. When you make a private request, please use the 'Feedback Request - SL' tag on Ed for this cycle.

Change for Fall 2025. Reviewer Response. In an effort to learn and grow assignment-to-assignment, we will provide a mechanism to edit and respond to your feedback. We will call this the *Reviewer Response.* You will have one week from the assignment grade being posted to edit and provide a two-page maximum response with both edits made and reviewer feedback. You will need to reasonably respond and edit your initial paper submission to improve your paper in good faith. This requires you to incorporate all the missed items from your feedback, rather than just a written rebuttal or disagreement. If all items are not met, the reviewer response will not be complete and no additional points will be awarded. Both the initial submission, revised submission, and two-page response will be needed for a proper Reviewer Response. If satisfied, you will receive half of the missed points back for the assignment. For example, if the initial grade was a 70/100, if everything is satisfied for the Reviewer Response, there will be 15 points added resulting in an 85/100. Further examples will be provided when the assignment grades are posted.

10 Plagiarism and Proper Citation

The easiest way to fail this class is to plagiarize. **Using the analysis, code, or graphs of others in this class is considered plagiarism**. The assignments are designed to force you to immerse yourself in the empirical and engineering side of ML that one must master to be a viable practitioner and researcher. It is important that you understand why your algorithms work and how they are affected by your choices in data and hyperparameters. The phrase "as long as you participate in this journey of exploring, tuning, and analyzing" is key. We take this very seriously and you should too.

What is plagiarism?

If you copy any amount of text from other students, websites, or any other source without proper attribution, that is plagiarism. The most common form of plagiarism is copying definitions or explanations from Wikipedia or similar websites. We use an anti-cheat tool to find out which parts of the assignments are your own and there is a near 100 percent chance we will find out if you copy or paraphrase text or plots from online articles, assignments of other students (even across sections and previous courses), or website repositories.

What does it mean to be original?

In this course, we care very much about your analysis. It must be original. Original here means two things: 1) the text of the written report must be your own and 2) the exploration that leads to your analysis must be your own.

It is well known that for this course we do not care about code. We are not interested in your working out the edge cases in k-NN, or proving your skills with Python. While there is some value in implementing algorithms yourselves in general, here we are interested in your grokking the practice of ML itself. That practice is about the interaction of algorithms with data. As such, the vast majority of what you're going to learn in order to master the empirical practice of ML flows from doing your own analysis of the data, hyperparameters, and so on; hence, you are allowed to use ML code from libraries but are not allowed to use code written explicitly for this course, particularly those parts of code that automate exploration. You will be tempted to just run said code that has already been overfit to the specific datasets used by that code and will therefore learn very little.

Information on citations:

If you refer to information from a third-party source or paraphrase another author, you need to cite them right where you do so and provide a reference at the end of the document [1]. Furthermore, "if you use an author's specific word or words, you must place those words within quotation marks and you must credit the source." [2]. It is good style to use quotations sparingly. Obviously, you cannot quote other people's assignments and assume that is acceptable. Citing is not a get-out-of-jail-free card: you cannot directly copy text flippantly, cite it all, and then claim it's not plagiarism just because you cited it. Too many quotes of more than, say, two sentences will be considered plagiarism and a terminal lack of academic originality.

All citations need to be in IEEE, MLA, or APA format.

Your README file will include pointers to any code and libraries you used.

If we catch you. . .

We report all suspected cases of plagiarism to the Office of Student Integrity. Students who are under investigation are not allowed to drop from the course in question, and the consequences can be severe, ranging from a lowered grade to expulsion from the program.

Update for Fall 2025: Use of LLMs and Generative AI

We treat AI based assistance the same way we treat collaboration with people. You may discuss ideas and seek help from classmates, colleagues, and AI tools, but all submitted work must be your own. For this course a large language model is any model with more than one billion parameters. These tools can increase productivity, but they are aids, not substitutes for your skills. The goal of reports is synthesis of analysis, not merely getting an algorithm to run. Even if you locate code elsewhere or generate code with an AI tool, you must apply it thoughtfully to your data and write the analysis yourself.

Every submission must include an AI Use Statement. List the tools used and what they assisted with and confirm that you reviewed and understood all assisted content. If an editor or platform surfaces suggestions, you may keep them only with disclosure and verification. You are not required to disable assistants. Be aware that coding editors often include assistants by default. Google Colab and Visual Studio Code commonly surface them. Also check any writing or grammar platforms you use. Overleaf through Georgia Tech is acceptable.

Allowed with disclosure. Brainstorming, outlining, grammar and clarity edits, code generation, code refactoring, and debugging.

Not allowed. Submitting AI written analysis, conclusions, or figures as your own. Fabricating results or citations. Paraphrasing AI or prior work to evade checks. Uploading course materials or private data to public tools.

Example Statement at the very end of the report before References. "AI Use Statement. I used ChatGPT and Visual Studio Code Copilot to brainstorm and outline sections of the report, generate and refactor small code snippets, debug a pandas indexing issue, and edit grammar and clarity throughout. I reviewed, verified, and understand all assisted content."

Verification will rely on provenance and reproducibility. We will not ask for live coding. We will review platform histories, including Overleaf revision timelines and copy and paste patterns, and Git commit and branch history. We may request your repository with full history, Overleaf project history, notebooks, logs, and intermediate outputs, and we may run reproducibility checks. Provide run instructions, environment details, random seeds, and the scripts used to generate figures and tables. Similarity and AI indicators, including Turnitin, are corroborating signals only and are not used in isolation.

If we suspect a violation, we will document evidence, notify you, and request artifacts. Cases may be referred to the Office of Student Integrity (OSI). Undisclosed or prohibited use is academic misconduct. This policy may be updated during the term. When in doubt, always use proper citations, and ask clarifying questions on Ed.

How to attribute & cite

Hotel Booking Demand: Data: Hotel booking demand (H1/H2), António, Almeida & Nunes (2019), *Data in Brief* 22:41–49, doi:10.1016/j.dib.2018.11.126.
US Accidents: Data: US Accidents (since 2016), Sobhan Moosavi et al., CC BY-NC-SA 4.0, retrieved from Kaggle US Accidents.

How to cite peer-reviewed sources in-text.

You may use **MLA**, **APA**, or **IEEE**; pick one style and stay consistent across the paper (in-text and references). Examples using the hotel paper:

- **APA**: "... (António, Almeida, & Nunes, 2019)." or "António et al. (2019) ..."
- **MLA**: "... (António, Almeida, and Nunes 2019)." or "António, Almeida, and Nunes argue ..."
- **IEEE**: "... [3]." (numbered, bracketed citations like [1]; reference list ordered by first appearance)

Include the full reference entry in your bibliography. In LaTeX, use BibTeX/BibLaTeX with an appropriate style (e.g., `style=apa` or `style=ieee`). *Tip:* In Google Scholar, click "Cite" → "BibTeX" to copy a starter entry, then verify authors, capitalization, and DOI.

11 Frequently Asked Questions (FAQ)

The assignment is open-ended, but expectations are concrete. Use this FAQ to avoid common pitfalls.

Scope & setup

Q: Exactly what must I run?

Run **four** algorithms on **both** datasets: **Neural Networks (NN)**, **Support Vector Machines (SVM)**, **k-Nearest Neighbors (kNN)**, and **Decision Trees** with **pruning**. Show required figures/metrics for each.

Q: Do I need outside sources?

Yes. Include at least **two peer-reviewed references** beyond course materials. This includes the datasets, the references must be helping to support your intial claims, methods choice, discussion rationale, etc. Use the references to justify methodological choices (metrics for imbalance, thresholding, calibration, pruning, kernels/activations, leakage controls) and to interpret results. Any one of **MLA, APA, IEEE** is acceptable, pick one style and be consistent.

Q: How do I split data?

If no split is provided: split once into **train/test**, tune on the **train** portion (CV or hold-out), and evaluate the finalized model **once** on the test set. Fix seeds and record exact splits.

Q: Any dataset-specific split guidance?

Hotel: if your target is tied to arrival dates (e.g., monthly ADR/cancellation rate), use a **time-aware** validation scheme. **US Accidents:** when predicting severity at onset, avoid using any post-outcome fields; consider time blocks and/or geography grouping to reduce spatiotemporal leakage.

Q: Can I add external features or extra data?

No. Restrict modeling to the provided datasets. You may do *minimal* feature engineering from existing columns (e.g., standardization, one-hot encoding, simple log transforms) but **avoid heavy feature engineering**, it can bias comparisons and confound which algorithm actually performs better. Do not augment with external data. If you create any features, (1) fit transformations on the training folds only (to prevent leakage), (2) apply the same pipeline consistently across all algorithms and both datasets, and (3) document exactly what you did and why.

Figures, metrics, and runtime

Q: Which figures are required for each model?

- **Learning curves (LC):** training & validation metric vs. training size.
- **Validation / model-complexity curves (MC):** validation metric vs. one hyperparameter (e.g., NN width/L2; SVM C/γ; kNN k; DT depth/`ccp_alpha`). Include training curve if space allows.
- **NNs also:** iteration/epoch curve (metric/loss vs. epoch) for early stopping/optimization behavior.

Q: What metrics should I report?

Classification: PR-AUC (with the **prevalence baseline**), ROC-AUC, and F1 at a **justified** threshold. If you present probabilities, include a calibration note/plot (reliability). **Regression:** MAE, MedAE, MSE (MAPE if scale matters); include residual diagnostics.

Q: How do I choose a classification threshold?

Pick a rule and state it (e.g., maximize F1 on validation; cost-sensitive threshold given FP/FN costs; target recall with acceptable precision). Use calibrated probabilities (Platt or isotonic) when thresholding.

Q: Do I have to scale features?

Yes for **SVM/kNN**. NN and DT are less sensitive, but standardization often helps NN training stability.

Q: Why report wall-clock times?

Complexity shows up in fit/predict time. Report wall-clock for both, plus a brief hardware note. This makes fair trade-off comparisons possible when metrics are similar.

Algorithm-specific guidance

Q: Special guidance for Decision Trees?

- **Preprocessing.** Trees are invariant to monotone scaling but still require clean inputs: impute missing values; encode categoricals (avoid exploding one-hot for very high-cardinality features and consider target/frequency encoding with nested CV and document it).
- **Complexity control (required).** Show a **pruning path/validation curve**. Use cost–complexity pruning (`ccp_alpha`) or pre-pruning controls (`max_depth`, `min_samples_leaf`, `max_leaf_nodes`). Report the final tree's **depth**, **#leaves**, and chosen complexity parameter.
- **What to plot.**
  - Learning curve (train/val metric vs. training size) to diagnose high variance (deep trees) vs. high bias (over-pruned trees).
  - Model-complexity curve: validation metric vs. one of {`max_depth`, `min_samples_leaf`, `ccp_alpha`}.
  - Optional: **pruning path** figure (alpha vs. #leaves and validation score).
- **Metrics & imbalance.** For classification, consider `class_weight='balanced'`; report PR-AUC (with prevalence baseline) and F1 at a justified threshold. For regression, report MAE/MedAE/MSE; consider the `absolute_error` criterion if outliers dominate.
- **Interpretation.** Prefer **permutation importances** to impurity-based importances; if space allows, a brief partial-dependence or ICE plot can clarify a key split.
- **Pitfalls.** Very deep unpruned trees; tuning on the test set; leakage via target-encoded features fit outside CV; over-interpreting a single learned tree on noisy data.

Q: Special guidance for SVM?

- **Preprocessing. Standardize features** (pipeline with scaler → SVM). Handle class imbalance with `class_weight='balanced'` where appropriate.
- **Kernels (required).** Use at least **two kernels per dataset** (e.g., RBF + linear or polynomial). Start with a **coarse log grid** for C and kernel params; refine around promising regions.
  - RBF: search $C \in [10^{-3}, 10^{3}]$, $\gamma \in [10^{-4}, 10^{1}]$ on a log scale.
  - Linear: vary C; use a linear SVM baseline on wide/medium-high dimensional data.
  - Polynomial: degrees 2–3 with regularization; beware explosive feature growth.

- **What to plot.**
  - Learning curve (train/val vs. training size).
  - Model-complexity curve: validation metric vs. C (and γ or degree), one at a time.
- **Probabilities & thresholds.** If you report probabilities, **explain calibration** (e.g., Platt/isotonic via a calibration wrapper fit on training folds). Justify your decision threshold.
- **Diagnostics.** Report the fraction of **support vectors**; unusually high fractions can indicate overfitting (too large C or γ).
- **Runtime tips.** Kernel SVMs scale poorly with n; if data are very large, consider subsampling for exploratory grids, then confirm on a larger subset. Use **LinearSVC/SVR** as scalable baselines when linear separation is plausible.
- **Pitfalls.** Unscaled features; reporting probabilities without calibration; sweeping multiple hyperparameters simultaneously without isolating effects; using accuracy alone on imbalanced data.

Q: Special guidance for kNN?

- **Preprocessing. Standardize** numeric features; impute missing values before distance computation. Document any feature drops.
- **Distance & weighting.** Justify k and weighting (`uniform` vs. `distance`). Consider Manhattan or cosine distance if feature scales/semantics suggest it.
- **High dimensionality.** Discuss the **curse of dimensionality**. However, you **SHOULD NOT** reduce dimensionality (e.g., PCA or ICA). That'll be for another assignment.
- **What to plot.**
  - Learning curve (train/val vs. training size).
  - Model-complexity curve: validation metric vs. k (and optionally weighting).
- **Metrics & imbalance.** For classification, consider distance weighting to help rare classes; report PR-AUC, ROC-AUC, and F1 at a justified threshold. For regression, report MAE/MSE and consider **median**-based aggregation if outliers dominate.
- **Runtime.** Training is cheap; prediction is expensive. Note the neighbor search algorithm (kd-tree/ball-tree/auto) and report wall-clock for predict.
- **Pitfalls.** Unscaled inputs; relying on a single k; ignoring tie-breaking; distance metric mismatch; heavy feature engineering that bakes in task-specific knowledge and biases comparisons.

Q: Special guidance for NN?

- **Preprocessing.** Standardize numeric inputs; encode categoricals consistently across models. Keep feature engineering **minimal** to avoid biasing algorithm comparisons.
- **Architecture & activations.** State hidden-layer sizes and **activation(s)**. As a baseline, a small MLP (e.g., two hidden layers) is sufficient on small tabular data. Match initialization to activation when possible (He for ReLU-family; Xavier for tanh).
- **Optimization & regularization.** Describe the optimizer, learning-rate schedule, early stopping, and any regularization (L2, dropout, batch norm). Keep the **output head and loss matched to the task**: sigmoid/BCE (binary), softmax/CE (multiclass), linear/MAE+MSE (regression).
- **What to plot.**
  - Iteration-based curve: validation metric (and/or loss) vs. epoch; indicate early-stopping point.
  - Learning curve (train/val vs. training size).
  - Model-complexity curve: validation metric vs. a single hyperparameter (e.g., width, L2, dropout rate, learning rate).

- **Calibration & thresholds.** If you will choose a classification threshold, include a calibration plot and justify the threshold rule. For imbalance, report PR-AUC with the prevalence baseline.
- **Diagnostics.** Check for under/overfitting via curves; inspect residuals for regression. If training is unstable (divergent loss, dead ReLUs), briefly note the cause and the minimal fix you applied.
- **Pitfalls.** Mismatched heads/losses; no standardization; comparing different training budgets across models; changing multiple factors at once so ablations are not interpretable.

Data issues, leakage, and encoding

Q: What are the leakage rules I must follow?

Hotel: drop `reservation_status` and `reservation_status_date` when predicting cancellation. **US Accidents:** exclude post-outcome fields for severity-at-onset (e.g., `End Time`, post-dated `Weather_Timestamp`). Document all leakage controls in your Methods.

Q: How do I handle high-cardinality categoricals?

Prefer target/frequency encoding with **nested CV**. One-hot only when cardinality is small. Explain your choice and show that encoding was fit on train folds only.

Q: Ordinal labels like severity 1–4: treat as multiclass or ordinal?

Default to **multiclass**. Ordinal methods are allowed if you justify them; report macro metrics and explain any coding/thresholding used.

Reproducibility, style, and submission

Q: What counts as reproducible?

Include a `DOCSTRING_{GTusername}` with env/deps and exact run commands, a single Git commit hash, and fixed seeds. Your results should regenerate from clean checkout.

Q: Style & submission?

≤ 8 pages including references; Overleaf + DOCSTRING with Overleaf link, single Git hash, exact run instructions. MLA/APA/IEEE allowed, pick one and be consistent.

What counts as "interesting"?

An experiment that **reveals differences** among algorithms or across datasets (e.g., metric vs. wall-clock, stability vs. capacity), not just domain appeal.

Good questions to ask yourself while working

- **Hypothesis:** Did I make a grounded claim based on theory and past evidence? Can my claims be tested with a reasonable experiment? Did I follow up on these claims in my discussion? Do my results support my discussion or not?
- **Leakage:** What features could leak the answer for this task? Did I remove them and document it?
- **Splits:** Are my validation choices appropriate (time-aware when needed)? Are seeds and splits recorded?
- **LC/MC:** Do my learning and model-complexity curves diagnose bias/variance, and did I act on those diagnoses?
- **Encoding & scaling:** Did I scale where required and encode high-cardinality features without leaking (nested CV)?

- **Comparisons:** Did I compare algorithms on both datasets using metrics *and* runtime? What is my evidence-backed takeaway?
- **Sources:** Which peer-reviewed papers justify my choices? Are my citations consistent (MLA/APA/IEEE)?
- **Conclusion:** Did I explicitly discuss Type I/II (or residuals for regression) and name one realistic next step?

12 Version Control

- v6.0 – 09/03/2025 – TJL updated dataset requirements with lighter limits.
- v5.0 – 09/03/2025 – TJL updated dataset usage and bullet point warnings.
- v4.0 – 08/17/2025 – TJL updated with edits from Head TA team.
- v3.0 – 08/15/2025 – Updated for Fall 2025 term; clarified libraries, late policy, citation styles; added figure/reproducibility checklists.
- v2.0 – 05/23/2025 – TJL cleaned up extraneous comments. Updated IEEE template link.
- v1.0 – 05/16/2025 – TJL finalized SL Report for Summer 2025 term.

References

[1] W. College. "Citing your sources: Citing basics." (), [Online]. Available: `https://libguides.williams.edu/citing`.

[2] U. of Wisconsin - Madison. "Quoting and paraphrasing." (), [Online]. Available: `https://writing.wisc.edu/handbook/assignments/quotingsources`.

[3] N. C. António, A. Almeida, and L. Nunes, "Hotel booking demand datasets," *Data in Brief*, vol. 22, pp. 41–49, 2019. DOI: `10.1016/j.dib.2018.11.126`.

Assignment description refactored and written by Theodore LaGrow. Updated for Fall 2025 by Theodore LaGrow.