MSIS 522

Analytics and Machine Learning

Foster School of Business | University of Washington
Instructor: Prof. Léonard Boussioux

Homework 1: The Complete Data Science Workflow

Individual Assignment	**Total Points: 100**

Deliverables: *A GitHub repository containing your code, saved models, and a deployed Streamlit app*

Overview

In this homework, you will carry out the **entire data science workflow** — from exploratory data analysis through predictive modeling, model explainability, and deployment — on a **tabular dataset of your choice**. Your final deliverables include a **GitHub repository** containing your code and **saved models**, as well as a **Streamlit web application**, deployed online, that showcases every stage of your analysis.

This assignment is designed to mirror the kind of end-to-end work you would do as a data scientist or analytics professional: pick a problem, explore the data, build and compare models, explain the results, and ship something stakeholders can interact with.

Choosing Your Dataset

You must select a **tabular dataset** suitable for a **classification or regression** task. The dataset should have a clear target variable you are trying to predict or forecast.

Options for finding a dataset:

- **Reuse a dataset from a previous course** that you already understand well.
- **Find a new dataset** on a topic you're interested in. Suggestions:
  - Kaggle Datasets
  - Hugging Face Datasets
  - Google Dataset Search
  - UCI Machine Learning Repository
- **Use the instructor-curated COVID-19 mortality dataset**: If you want a dataset known to be suitable, check the notebook and dataset here: Google Colab.

Requirements for the dataset:

- It must be **tabular** (rows and columns, not images or raw text) so that you can practice with all the models below.
- It should have **enough rows** (at least 100–200) to train meaningful models.

- The prediction task can be **classification** (e.g., will a patient die or survive, will a customer churn) or **regression** (e.g., predicting a price, a score, a count).
- If the dataset is imbalanced, you should address this (e.g., through sampling, class weights, or appropriate metrics).

Assignment Structure

Your work is organized into three parts, plus deployment.

Part 1: Descriptive Analytics (25 points)

The goal here is to deeply understand your dataset before modeling. You should tell a compelling visual story about the data.

1.1 Dataset Introduction (5 points)

Provide a clear description of your dataset:

- What does it contain? Where did it come from?
- What is the **prediction target** (the dependent variable)?
- Why is this prediction task interesting or impactful?
- Basic statistics: number of rows, number of features, feature types (numerical vs. categorical).

1.2 Target Distribution (5 points)

Plot the distribution of your target variable.

- For classification: a count plot or bar chart showing class frequencies.
- For regression: a histogram or KDE plot of the target values.

Comment on what you see. Is the target balanced? Skewed? Are there outliers? If the target is imbalanced, explain how you plan to handle this.

1.3 Feature Distributions and Relationships (10 points)

Create **at least four** insightful visualizations. These should go beyond basic bar charts — aim to reveal patterns that would not be obvious from the raw data. Examples include:

- Histograms or boxplots of key features, possibly broken down by the target variable.
- Violin plots or swarm plots showing distributions across categories.
- Pair plots for a subset of the most interesting numerical features.
- Bar charts of categorical features vs. the target (e.g., mortality rate by comorbidity).
- Any other visualization you find revealing.

For **each** visualization, write a brief interpretation (2–3 sentences) explaining what insight it provides.

1.4 Correlation Heatmap (5 points)

- Compute and plot the correlation matrix as a heatmap (e.g., using `sns.heatmap`).

- Briefly comment on the strongest correlations you observe and what they might mean for your modeling.

Part 2: Predictive Analytics (45 points)

In this part, you will build, tune, and compare multiple models. Use hold-out validation or cross-validation for hyperparameter tuning and keep a held-out **test set** (e.g., 30%) for final evaluation.

Use `random_state=42` wherever randomness is involved.

2.1 Data Preparation (not graded, but required)

- Define your features `X` and target `y`.
- Perform a train/test split (e.g., 70/30).
- Apply any necessary preprocessing (scaling, encoding, handling missing values). Document what you did and why.

2.2 Linear or Logistic Regression Baseline (5 points)

If your task is **regression**, fit a **Linear Regression** model. If your task is **classification**, fit a **Logistic Regression** model.

Report relevant metrics on the test set:

- **Classification:** Accuracy, Precision, Recall, F1, AUC-ROC.
- **Regression:** MAE, RMSE, R^2.

This serves as your **baseline**. All subsequent models should aim to beat it. You can also include Lasso, Ridge, and/or Elastic-Net variants if you wish.

2.3 Decision Tree / CART (5 points)

Implement **5-fold cross-validation** with `GridSearchCV` over:

- `max_depth`: e.g., [3, 5, 7, 10]
- `min_samples_leaf`: e.g., [5, 10, 20, 50]

Use the appropriate scoring metric (F1 for classification, neg_mean_squared_error for regression). Report and save the best hyperparameters and test-set metrics. Visualize the best tree (if it is not too deep to display).

> *Note: the values above are provided as guidance. Another set of parameters may be more relevant for you. For instance, if your dataset is very large, you may prefer 3-fold cross-validation.*

2.4 Random Forest (10 points)

Implement **5-fold cross-validation** with `GridSearchCV` over:

- `n_estimators`: e.g., [50, 100, 200]
- `max_depth`: e.g., [3, 5, 8]

Report and save the best hyperparameters and test-set metrics. Plot the ROC curve (classification) or a predicted-vs-actual scatter (regression).

2.5 Boosted Trees — XGBoost or LightGBM (10 points)

Choose **at least one** gradient-boosted tree model (e.g., XGBoost, LightGBM).

Implement **5-fold cross-validation** with `GridSearchCV` over at least three hyperparameters. Suggested grid:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [3, 4, 5, 6]
- `learning_rate`: [0.01, 0.05, 0.1]

Report and save the best hyperparameters and test-set metrics. Plot the ROC curve (classification) or predicted-vs-actual scatter (regression).

2.6 Neural Network — MLP (10 points)

Build a **Multi-Layer Perceptron (MLP)** using Keras, PyTorch, or TensorFlow.

Minimum architecture:

- Input layer matching your feature dimensions.
- At least **two hidden layers** (e.g., 128 units each) with ReLU activation.
- Output layer appropriate to your task (sigmoid for binary classification, linear for regression, softmax for multi-class).
- Appropriate loss function (binary cross-entropy, MSE, etc.) and the Adam optimizer.

Tasks:

1. Train the model and plot the training history (loss curve, and accuracy or relevant metric curve).
2. Report test-set metrics (same metrics as above).
3. *(Optional for bonus — see Section 4)*: Tune hyperparameters.

2.7 Model Comparison Summary (5 points)

- Create a summary **table** showing all models and their test-set metrics side by side.
- Create a **bar chart** comparing the key metric (F1 or RMSE) across all models.
- Write a brief paragraph: Which model performed best? Were you surprised? What trade-offs exist between the models (accuracy vs. interpretability, training time, etc.)?
- Include all of this into your Streamlit app.

Part 3: Explainability (10 points)

3.1 SHAP Analysis (10 points)

Using your **best-performing tree-based model** (Random Forest or Boosted Tree), perform a SHAP analysis.

Required plots:

1. **Summary plot** (beeswarm) — shows feature importance and direction of impact.
2. **Bar plot** of mean absolute SHAP values — ranks features by importance.
3. **Waterfall plot** for one specific prediction (e.g., a high-risk individual or an interesting edge case).

Required interpretation:

- Which features have the strongest impact on predictions?
- How do those features influence the prediction (positively or negatively)?
- How could these insights be useful to a decision-maker in your domain?

> ***Tip:*** *State-of-the-art LLMs (ChatGPT, Claude, Gemini) know SHAP well and can help you with the code. However, it is best to use your own judgment for the interpretation as they still make some subtle mistakes. SHAP documentation:* shap.readthedocs.io

---

Part 4: Streamlit Deployment (20 points)

Note: Your Streamlit app is the final deliverable for this project. All findings, visualizations, and models from Parts 1, 2, and 3 must be presented within the app.

You will build and **deploy** a Streamlit app that presents your entire analysis. The app must be accessible via a public URL (e.g., deployed on Streamlit Community Cloud, Hugging Face Spaces, or any other hosting service).

Required Tabs

Your Streamlit app must have the following tabs (use `st.tabs`):

Tab 1 — Executive Summary (4 points)

- A clear description of your dataset and the prediction task (at least one paragraph). Someone who doesn't know your dataset should understand exactly what your data is about, what the target is, and what features are available.
- Why this problem matters — the "so what." (at least one paragraph).
- A brief overview of your approach and key findings (1-2 paragraphs).
- This is the tab a non-technical stakeholder would read. Just placing key metrics is not enough.
- Be careful: If you heavily use AI to create your app, it will tend to create a very low-quality executive summary that won't meet the requirements for a full score unless you specifically prompt what you want in the executive summary.

Tab 2 — Descriptive Analytics (4 points)

- Display the key visualizations from Part 1 (e.g., target distribution, feature distributions, correlation heatmap).
- Include brief captions or commentary for each plot (brief captions = minimum 2 sentences that would help a reader understand your plot). Be careful: AI tends to forget to write those if you one-shot your app!

Tab 3 — Model Performance (4 points)

- Show the model comparison table and bar chart from Section 2.7.
- Include the ROC curves (classification) or predicted-vs-actual plots (regression) you generated for each model in Part 2.
- Report the best hyperparameters for each model.
- All metrics, plots, and graphs produced in Part 2 should be surfaced here so a reader can evaluate model performance without opening your code.

Tab 4 — Explainability & Interactive Prediction (8 points)

This is the interactive tab.

- Display the SHAP summary and bar plots.

- **Interactive prediction feature:** provide sliders, dropdowns, or input fields (using `st.slider`, `st.selectbox`, etc.) that let the user set feature values and see what a given model predicts in real time (if there are numerous input features, select a handful that would be relevant to manually set, and use the average value for the others).
  - The user should be able to select which model to use for the prediction.
  - Show the predicted outcome (class and probability for classification, or predicted value for regression).
  - Show a SHAP waterfall plot for the user's custom input.

Deployment Requirements

- Your models must be **pre-trained and saved** (e.g., using `joblib`, `pickle`, or framework-specific saving). The Streamlit app loads saved models — it does **not** retrain them on the fly.
- The app must be **deployed and accessible** via a public link at the time of submission.
- ⚠️ **Do NOT submit a `localhost` link.** A localhost URL (e.g., `http://localhost:8501`) only works on your own computer. No one else can access it. You must deploy your app to a cloud service so that anyone can open it in their browser. Test this by opening your link in an incognito window or on your phone. The simplest is probably to deploy with Streamlit Community Cloud.

Deliverables

Submit the following on Canvas:

1. **A link to your GitHub repository** containing:
   - Your Jupyter notebook or Python scripts with all analysis code.
   - The Streamlit app code (e.g., `app.py` or `streamlit_app.py`).
   - Saved model files.
   - A `requirements.txt` for reproducibility.
   - A short `README.md` explaining how to run everything.
2. **A link to your deployed Streamlit app.**

Grading Rubric

Component	Points
Part 1: Descriptive Analytics	**25**
1.1 Dataset Introduction	5
1.2 Target Distribution	5
1.3 Feature Distributions (≥4 visualizations + interpretation)	10
1.4 Correlation Heatmap	5
Part 2: Predictive Analytics	**45**
2.2 Linear/Logistic Regression Baseline	5
2.3 Decision Tree with CV	5

2.4 Random Forest with CV	10
2.5 Boosted Trees with CV	10
2.6 Neural Network (MLP)	10
2.7 Model Comparison Summary	5
Part 3: Explainability (SHAP)	**10**
Part 4: Streamlit Deployment	**20**
Tab 1 — Executive Summary	4
Tab 2 — Descriptive Analytics	4
Tab 3 — Model Performance	4
Tab 4 — Explainability & Interactive Prediction	8
Total	**100**

Bonus (1 point)

Bonus	Points
Neural Network hyperparameter tuning (grid search over hidden layer sizes, learning rates, dropout rates — with visualization of results)	1

Important Notes

- **Random seed:** Use `random_state=42` for all operations involving randomness (train/test split, cross-validation, model initialization).
- **LLM usage:** You are welcome to use LLMs (ChatGPT/Codex, Claude, Gemini, VS Code, Antigravity, etc.) as much as you want to help with **coding**. However, all **interpretations, commentary, and the executive summary** must be thoroughly checked by you or written in your own words. One way to get started could literally be to paste this entire document to your favorite model and ask it to "do the work" for your specific dataset. If you use this method, make sure everything is correct and that the model did not forget anything! It is very probable that it will oversimplify some details, forget a model, not hypertune properly, or have other potential bugs. We'll be on the lookout for these ;)

Optional Dataset: COVID-19 Patient Outcomes & Risk Stratification

If you prefer to use a dataset provided by the instructor, you can use this one.

Predicting patient mortality is one of the most critical challenges in modern healthcare informatics. This dataset provides a high-stakes environment for predictive modeling, containing ~1M anonymized patient records with features such as age, sex, and pre-existing comorbidities (diabetes, hypertension, obesity, COPD, etc.). The binary target, **DEATH**, allows you to build a classifier that distinguishes between survivors and high-risk cases.

Why This Matters

In a clinical setting, the ability to accurately stratify risk is not just a technical exercise — it is a tool for **resource allocation and triage**. By identifying which comorbidities most significantly drive mortality risk, healthcare providers can prioritize care for the most vulnerable populations. Furthermore, this dataset is ideal for **SHAP analysis (Part 3)**, as understanding the "why" behind a prediction is essential for building trust with medical professionals and ensuring ethical AI deployment.

Starter Resources:

- Google Colab Starter Notebook & Data Link

Good luck, and have fun building your end-to-end data science pipeline!

Questions? Email leobix@uw.edu