CPSC 5310 – Machine Learning
Winter Quarter 2026

**Review & Reflection**

# End-To-End Machine Learning Project

**Rizvan Ahmed Rafsan**
**Ben Tran**

## Project Overview

The project involves building a model using California census data to replace manual expert housing price estimates that are often inaccurate and costly. This is specifically a *multiple univariate regression problem* where several features are used to predict a single continuous value which is the median house price. The ultimate goal is to increase company revenue by accurately identifying profitable real estate investment opportunities. The price predictions from the model will act as a *signal*[1] for a downstream investment analysis system, which will determine whether a specific area is worth the financial investment.

## Reproducing the Workflow

First the problem was identified as a supervised multiple regression task to predict district housing prices. Data loading involved fetching census data, followed by exploration using histograms and correlation matrices to identify underlying patterns. Preprocessing included handling of missing values via imputation, one-hot encoding categorical features, and applying feature scaling using transformation pipelines. Models like Linear Regression and Random Forests were trained, using cross-validation to detect overfitting. Evaluation relied on RMSE. The best model is fine-tuned via random search cross-validation and assessed on the test set. Most of the steps were straightforward to me except for using the `ColumnTransformer` to transform the data and finally integrating it to the model pipeline.

## Learnings from Code Execution

While I found most of the core project concepts intuitive, mastering the implementation of machine learning pipelines required more effort. I learned that pipeline chains multiple data processing steps (like imputation and scaling) into a single object. This ensures the exact same transformations are applied to the training set, test set, and new data, preventing errors and data leakage. The `ColumnTransformer` is used to apply different pipelines to specific column types simultaneously and concatenates the results. Wrapping preprocessing steps in a pipeline helps in tuning the preprocessing parameters alongside the model's own hyperparameters. Ultimately, while the reading provided a conceptual foundation, executing the code myself was essential for solidifying my understanding and identifying potential pitfalls.

## Challenges and Debugging

While executing the code, I noticed a minor discrepancy between the outputs printed in the book and those generated by the author's own notebook. Interestingly, my personal execution perfectly matched the notebook's results, which suggests that the book's static figures may be slightly outdated. We attribute these differences to a likely update in the dataset after the book's publication. This discrepancy serves as a practical reminder of a common real-world challenge

[1] The "signal" is the underlying real-world pattern or insight we want the model to detect.

which is *data drift*[2]. Aside from this minor variation, the code execution was entirely seamless and there were no further technical obstacles.

## Team Experience

Throughout the project, we maintained an individual execution of the codebases while establishing a consistent peer-review process to sync our outputs and interpretations. This approach allowed us to deconstruct the author's logics and troubleshoot complex concept. I believe that for any machine learning project, a comprehensive understanding of the end-to-end workflow is vital. Therefore, frequent technical discussions regarding methodology, alternative approaches, and result validation are essential to a project's success.

## Reflection & Preparation for the Quarter-Long Project

Top learnings from this project:

1. **Automation via Pipelines:** Transformation Pipelines are important for output reproducibility. By wrapping steps like imputation, scaling, and encoding into a single `Pipeline` object, we can ensure the exact same transformations applied to training data are applied to new data.
2. **Data Snooping Bias:** We must set aside a test set immediately before exploring the data. If we look at the test data too early, we risk subconsciously tailoring our model to it.
3. **Model Rot:** Models degrade over time as data evolves, making monitoring and maintenance (MLOps) often more work than building the initial model.

The insights gained from this project provide a structured framework that I can apply to more complex, large-scale ML projects in the future. This experience has significantly increased my proficiency with Scikit-Learn's pipelines, as well as its transformer, estimator, and inspection classes. One notable challenge I encountered was the computational intensity of traditional `GridSearchCV` or `RandomSearchCV` when applied to models like Random Forest, where processing time can scale exponentially with dataset size. Consequently, I am eager to implement more efficient strategies, such as `HalvingRandomSearchCV` or `HalvingGridSearchCV`, to optimize hyperparameter tuning in the upcoming project.

[2] Data drift occurs when the statistical properties of the input data (features) or .the target variable change over time.