End-to-End Machine Learning Project Reflection

Hongru He

CPSC5310 26WQ

This project follows the end-to-end machine learning example in Chapter 2 of *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, which focuses on predicting median house values in California districts using census data. The task is a **supervised learning regression problem**, where the goal is to build a model that can generalize well to unseen districts and provide accurate price predictions based on multiple numerical and categorical features.

Although the project is completed as a team exercise, one of the first things I realized is that **each team member still benefits from personally going through the data loading and exploration steps**. Working directly with the dataset—examining distributions, correlations, missing values, and feature meanings—helped me develop an intuitive understanding of what the model was learning from. This individual familiarity with the data made later team discussions more productive and grounded.

Before any code was written, the **problem framing step** turned out to be especially important. Our team spent time aligning on what "success" meant for this model, what type of prediction we were making, and which evaluation metric made sense for a regression task. This step helped ensure that all subsequent decisions—feature engineering, preprocessing, and model selection—were aligned with a realistic and well-defined goal.

During data exploration, we made a conscious effort to **record questions and hypotheses** about the data. For example, we considered how location (longitude and latitude), population density, and median income might influence housing prices, and what kinds of new features could be derived from the raw variables. This stage encouraged creative thinking, but the project also made it clear that **feature engineering decisions should ultimately be guided by statistical methods**, such as correlation analysis and model performance, rather than intuition alone. The distinction between numerical features and categorical features (such as the ocean proximity variable) became especially clear here.

In the preprocessing stage, handling **missing values** emerged as a critical first step. Since regression models expect continuous inputs, missing data must be addressed carefully to avoid breaking the learning process. Applying techniques like median imputation helped maintain data continuity while preserving statistical robustness. Another key lesson was the importance of **splitting the dataset into training and test sets early**, so that model evaluation remains unbiased and truly reflects generalization performance.

Running the code made several abstract concepts much clearer. The use of **pipelines** stood out as particularly powerful, as they allow preprocessing, feature transformation, and model training to be treated as a single, organized workflow. This structure not only reduces errors but also makes experiments easier to reproduce and modify. Additionally, experimenting with **grid search,**

randomized search, and ensemble methods clarified when simple models are sufficient and when more complex approaches are worth the added cost.

We did encounter minor debugging issues, mainly related to data transformations and feature alignment, but resolving them reinforced the reality that real ML code rarely runs perfectly on the first try. As a team, we are going to divide responsibilities between code execution, conceptual understanding, and documentation in the upcoming quarter-long project, which we believe would help us move efficiently while still learning collaboratively.

Overall, my main takeaways are the importance of careful problem framing, disciplined data preprocessing, and structured workflows. This guided project gave me greater confidence in navigating the full ML pipeline and highlighted areas—such as feature engineering and hyperparameter tuning—that I am eager to explore more deeply in the upcoming quarter-long project.

[**Word count: 570**]