Andy Ho

March 9, 2025

CS372: Machine Learning For Data Science

**Project 3 Report:**

1: In this project I worked with a housing dataset on Kaggle found [here](https://www.kaggle.com/datasets/dhirajnirne/california-housing-data/data). The data works with housing measurements in California, providing information on residencies by city block. This aggregation by city block is why two key features in the data are represented as median values, median income and median house value, serving as a better representation of a center without getting skewed by one outlier on a city block. The features are mainly numerical, with the two exceptions being an “ocean\_proximity variable” and latitude/longitude measurements for each city block.

2: For this project we we’re tasked with 5 different models to build and train, and in this project I accomplished both regression and classification tasks. I chose to use both linear regression and neural networks to predict median housing values in different city blocks. In addition to these regression models, I used SVM’s, decision trees, and random forests to attempt to classify the “ocean\_proximity” label. As a note here, latitude/longitude was excluded for both decision trees and random forests in this classification task as it made the model “too accurate” (Near 100% accuracy on test sets), as ocean proximity was primarily dependent on location (found in latitude/longitude).

A graph of a house value

AI-generated content may be incorrect.3: To pre-process this data there were three main things that had to be accomplished: Deal with missing values/outliers, encode the categorical variables, and standardize the numeric data. As far as missing values went, there were only 207 in them in the “total\_bedrooms” column and with this dataset of 20640 observations I made the decision to drop the rows from this data set considering how small this chunk of data is. Another problem in this data set is that the median house value seemed to have data that was encoded incorrectly, given that there was a massive spike in feature counts past 500,000 (See histogram). Upon further investigation I found that these observations we’re primarily close to the ocean, a quality that very commonly raises house prices in California. With this I concluded that these values we’re city blocks that had median values >500,000. With no way to obtain the true values, I removed them with the acknowledgement that this limits our model to median house values to be below 500,000 considering we’re removing the chunk above 500,000. To conclude our pre-processing, I standard scaled numeric features and used one hot encoder to encode our categorical features, leaving the longitude and latitude features untouched.

4: Starting with our linear regression models, I fitted three linear regression models onto this data set being ordinary multi-linear regression, ridge regression with a gaussian kernel, and ridge regression with a polynomial kernel of degree 3. An important note is that the full kernels we’re ridiculously computationally expensive, so when I say “kernel” in regards to this regression I mean an approximation of the kernel using the Nystroem module in scikit. Ranking these three methods I found that the model with the lowest error (measured in MSE) was the linear model with the gaussian kernel, followed by the polynomial kernel, with ordinary multi linear regression ranked last. Non-suprisingly this order remained the same for train sets, validation sets and test sets (See test error output below). One comment to make is that the polynomial kernel was subject to high variability where it would occasionally massively underpredict/overpredict one observation point by millions, remaining stable for other points. See figure below to compare predicted median value vs actual median value on our different models.

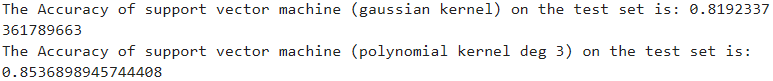
A close up of text

AI-generated content may be incorrect.

A yellow and blue dotted diagram

AI-generated content may be incorrect.

Continuing on to SVM’s, our goal with these models was to predict ocean proximity given features without longitude and latitude. For these models I used a gaussian kernel (No approximation needed) and a polynomial kernel of degree 3. These models we’re atrociously bad at predicting ocean proximity, predicting with roughly 40ish percent accuracy before I got curious and started feeding it longitude/latitude and even then it only managed to predict these labels at around an 80% - 85% accuracy. From here we can conclude that our model just wasn’t that linearly separable even in higher planes, with SVM’s being a poor choice to classify these models.



Luckily, decision trees and random forests we’re much better at predicting ocean proximity. With my curiosity fueled after the SVM’s, I started by feeding both the decision tree and the random forest longitude/latitude to find … about a 99% classification on train, validation, and test sets without any hyper-parameter tuning. With this restoring my faith I removed the longitude and latitude feature to have the decision tree predict with roughly 66% accuracy on the train, validation and test sets (see below). To tune these parameters I utilized the grid search module in Scikit to go through possible combination of hyper-parameters.

A close up of black text

AI-generated content may be incorrect.

For the random forests, there wasn’t anything different, with the results being roughly the same. While ever so slightly higher, I’m attributing the difference in accuracy between the random forests and decision trees to complete chance as opposed to model superiority.

A white background with black text

AI-generated content may be incorrect.

Finally moving onto the neural network, we’re shifting gears back to attempt to solve the first linear regression task of predicting median house values. With the neural network I also fed it the longitude/latitude features, where-in I didn’t for the linear regression models. With this model I ran into the problem of vanishing gradients, where the gradient was somehow 0, so model weights weren’t updating and the network was completely failing to converge. I fixed that with some online research and by changing the RELU function to [ELU](https://pytorch.org/docs/stable/generated/torch.nn.ELU.html), which essentially allowed gradients to be slightly negative in some cases preventing vanishing gradients. These changes made the network predict better values than our linear regression tasks by roughly 10% (RMSE going from 56000 -> 51000 on respective test sets.

A black text on a white background

AI-generated content may be incorrect.

5: Let’s start with the neural networks: Here are two plots of the training/validation/test sets RMSE. In these plots we can see that the model seems to converge around 3000 epochs. A graph of a number of numbers

AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.However while these do converge there, it’s important to note the scales of this graph. For our model we don’t gain significant improvements from further training, but small improvements in the range of 1000 are still attainable, which would have impacts on somebody looking to buy a home in these areas. Continuing to decision trees, it seems that they are the most prone to overfitting in comparison to any other model, where you would find accuracies on the training set of 100% without any sort of hyper parameter tuning, with the test set accuracy around 40% in these incidences. Random forests suffer from the same problem but that’s no surprise as random forests are literally a collection of bootstrapped decision trees. For our linear regression model I noticed that towards higher median values we tended to under-predict housing median value, and we could attempt to combat that by adding a sort of exponential bias term that increases in value as we increase in predicted housing values to offset this consistent under-prediction.