Machine Learning Applications in Real Estate: Predicting Housing Prices and Loan Approval

Andrew Ebert
Department of Mathematical and Statistical Sciences
University of Colorado Denver
Denver, CO, USA
andrew.ebert@ucdenver.edu

***Abstract*—This report explores the application of machine learning techniques in predicting housing prices and loan approval based on housing data from California and synthetic data of loan approval. Using a combination of regression and classification models, this report evaluates different models' performance in estimating housing costs and determining loan eligibility. Data preprocessing, exploratory data analysis, regularization, normalization, and feature contributions were applied to optimize model accuracy. The results from these models can offer predictive insights for home buyers and organizations approving loans, and further demonstrate real-world application of two different areas in Machine Learning — regression and classification. The study will conclude by discussing the limitations of our models and and potential improvements.**

I. Introduction

This sections introduces both datasets, housing and loan respectively, covering the features of each dataset, the preprocessing techniques employed, and an illustration of Exploratory Data Analysis (EDA) performed.

A. Introduction to the Housing Dataset

The housing dataset consists of 10 columns and 20,640 rows. The features include longitude, latitude, median house age, total rooms, total bedrooms, population, households, median income, median house value, and ocean proximity. The target variable, or variable we will be attempting to predict from our models, is the median house value — which is continuous.

While not certain, I believe each row contains data on a given tract, county, neighborhood, or some other geographical subset. This is inferred from the data, and a sample record can be seen in "Table 1".

Furthermore, I analyzed the number of null values in the dataset, with the corresponding frequency in each respective column. Additionally, I displayed the datatype of each column. See "Table 2" for reference.

I employed three different techniques to work around the null values. The first was removing rows with null values. The second technique was filling in the null values with the mean total bedrooms across the dataset. The final, and most creative, technique was finding the average proportion between total rooms and total bedrooms across the dataset. From there, filling in the null values — all being in the total_bedrooms column — with the average proportion multiplied by the row's corresponding total_rooms value. Regarding the object datatype for ocean proximity, I used Scikit-learn's LabelEncoder to convert unique strings to a corresponding integer label, which could then be used by the models.

Finally, within the introductory phase I plotted histograms of each column's frequency distribution, along with the mean and median. For the sake of this report, I won't include those plots as nothing was deemed abnormal.

TABLE I
Sample Record from Housing Dataset - Transposed

Feature	Value
longitude	-122.23
latitude	37.88
housing_median_age	41.0
total_rooms	880.0
total_bedrooms	129.0
population	322.0
households	126.0
median_income	8.3252
median_house_value	452600.0
ocean_proximity	NEAR_BAY

TABLE II
Housing Dataset Summary: Column Completeness and Datatypes

Column Name	Non-Null Count	Null Count	Datatype
longitude	20640	0	float64
latitude	20640	0	float64
housing_median_age	20640	0	float64
total_rooms	20640	0	float64
total_bedrooms	20433	207	float64
population	20640	0	float64
households	20640	0	float64
median_income	20640	0	float64
median_house_value	20640	0	float64
ocean_proximity	20640	0	object

B. Introduction to the Loan Dataset

The loan dataset consists of 14 columns and 45,000 rows. The features include age, gender, education, income, employment experience, home ownership, loan amount, loan intent, loan interest rate, loan percent income, credit history length, credit score, previous loans on file, and loan status. The target value for this dataset is loan status, which is either 0 (not

approved) or 1 (approved). A sample record can be seen in "Table 3".

Furthermore, I analyzed the number of null values in the dataset, with the corresponding frequency in each respective column. Additionally, I displayed the datatype of each column. See "Table 4" for reference.

This dataset didn't have any null values, so there was nothing to do in that sense. Regarding the object datatypes, I used .replace() to transform object columns to numeric. This was different than using LabelEncoder above, but this was creative liberty that made it easier to understand what each unique value corresponds to — considering there was more than one object datatype for this dataset.

Finally, within the introductory phase I plotted histograms of each column's frequency distribution, along with the mean and median. Again, for the sake of this report I won't include those plots as nothing was deemed abnormal.

TABLE III
SAMPLE RECORD FROM LOAN DATASET - TRANSPOSED

Feature	Value
person_age	22.0
person_gender	Female
person_education	Master
person_income	71948.0
person_emp_exp	0
person_home_ownership	RENT
loan_amnt	35000.0
loan_intent	PERSONAL
loan_int_rate	16.02
loan_percent_income	0.49
cb_person_cred_hist_length	3.0
credit_score	561
previous_loan_defaults_on_file	No
loan_status	1

TABLE IV
LOAN DATASET SUMMARY: COLUMN COMPLETENESS AND DATATYPES

Column Name	Non-Null Count	Null Count	Datatype
person_age	45000	0	float64
person_gender	45000	0	object
person_education	45000	0	object
person_income	45000	0	float64
person_emp_exp	45000	0	int64
person_home_owner...	45000	0	object
loan_amnt	45000	0	float64
loan_intent	45000	0	object
loan_int_rate	45000	0	float64
loan_percent_income	45000	0	float64
cb_person_cred_hist...	45000	0	float64
credit_score	45000	0	int64
previous_loan_def...	45000	0	object
loan_status	45000	0	int64

II. LINEAR REGRESSION FOR HOUSING DATA

This section will discuss the various linear regression models I employed for the housing dataset, as well as the finetuning I did to achieve the most optimal model. Again, as mentioned in the introduction, I employed three different techniques to work around the null values. The first was removing rows with null values. The second technique was filling in the null values with the mean total bedrooms across the dataset. The final, and most creative, technique was finding the average proportion between total rooms and total bedrooms across the dataset. From there, filling in the null values — all being in the total_bedrooms column — with the average proportion multiplied by the row's corresponding total_rooms value. I will refer to each of these as Model 1, 2, and 3 respectively.

A. The Three Different Models Used

For each of these models I employed a 75%/25% train/test split. Without any additional model finetuning, these were the results I observed — displayed in "Table 5".

TABLE V
MODEL PERFORMANCE METRICS

Model	R^2	RMSE
Linear Regression Model 1	0.637	69996.90
Linear Regression Model 2	0.641	70091.32
Linear Regression Model 3	0.643	69926.79

Interestingly enough, the third model had the best performance. While not much difference in the models' performances, I proceeded by finetuning model three.

B. Brute Force Feature Contributions

From here, I decided to retrain model three without each feature as a brute force approach of understanding features which were helping/hurting the model's performance. The results are in "Table 6" below.

TABLE VI
IMPACT OF FEATURE REMOVAL ON MODEL PERFORMANCE

Removed Feature	R^2	RMSE
longitude	0.583	75524.10
latitude	0.575	76298.93
housing_median_age	0.629	71331.53
total_rooms	0.642	70005.04
total_bedrooms	0.638	70425.03
population	0.622	71968.68
households	0.643	69882.65
median_income	0.389	91481.54
ocean_proximity	0.643	69910.83
ocean_proximity, households	0.643	69869.20

After analyzing the results, I retrained the model without ocean_proximity and households because the models without these features performed better. The results can be seen in the same table. While not a significant difference in performance, the RMSE is certainly lower in the final model.

C. Pipeline, Scaling, and Polynomial Features

Then, I began experimenting with a pipeline where I would scale the data using StandardScaler, and ultimately training multiple regression models with different polynomial degrees. The results are shown in "Table 7".

While degree two has a lower R^2 and RMSE, it has a lower Cross Validation (CV) score also, indicating the model

TABLE VII
POLYNOMIAL REGRESSION MODEL PERFORMANCE

Degree	R^2	RMSE	Avg CV Score
1	0.643	69869.37	0.631
2	0.698	64325.85	0.532
3	0.444	87239.24	-2.237
4	-15.854	480363.57	-714.143
5	-5119.629	8369685.08	-64638.609

is overfitting. The margin between R^2 and CV can be seen in "Fig. 1", and the margin grows significantly between degrees one and two despite a higher R^2.

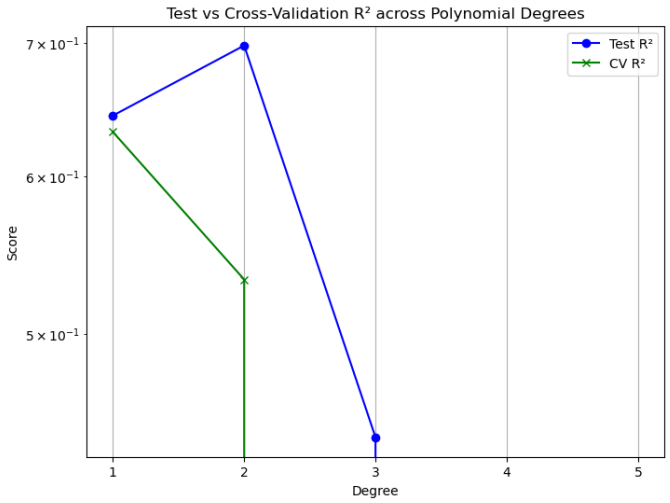

Fig. 1. R^2 vs. Cross Validation per Degree

D. Visualization

Finally, I assessed the performance of the model by plotting actual versus predicted home values. While the model clusters pretty well on the line near the middle of our data, there are some strong outliers near the beginning and ends of our data, indicating the model isn't performing well on the ends of the data. For reference, look to "Fig. 2"

III. LOGISTIC REGRESSION FOR LOAN DATA

This section will discuss the various logistic regression models I employed for the loan dataset, as well as the finetuning I did to achieve the most optimal model.

A. Base Model

I initially trained a very simple logistic regression model with optimal accuracy, and observed an accuracy of 0.873. A seemingly good accuracy, but when looking at the Confusion Matrix it's evident that our model is favored to predict 0 because there are significantly more 0 values. This resulted in a lot of 0s predicted when the actual label is 1. The Confusion Matrix can be seen in "Fig 3." and the Precision Recall curve can be seen in "Fig 4.".

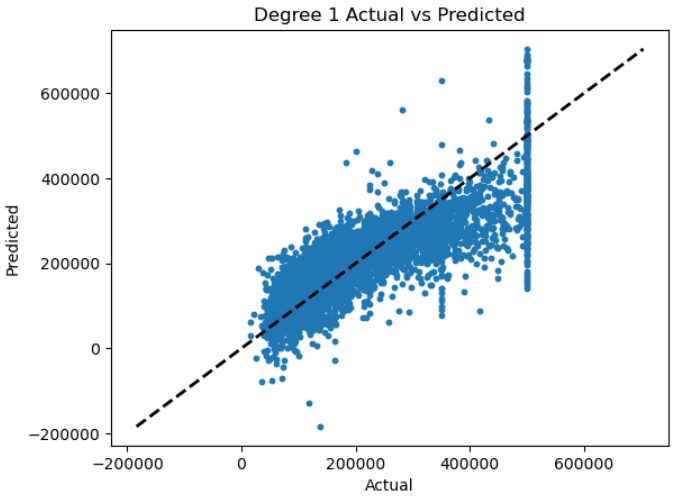

Fig. 2. Actual vs. Predicted Home Values

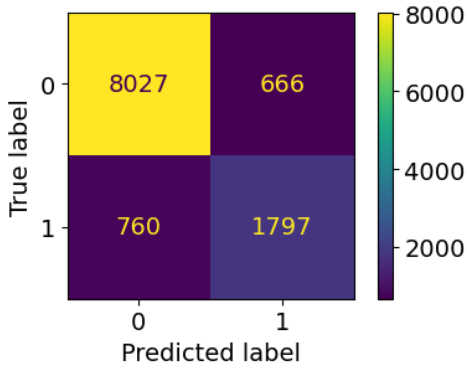

Fig. 3. Base Model Confusion Matrix

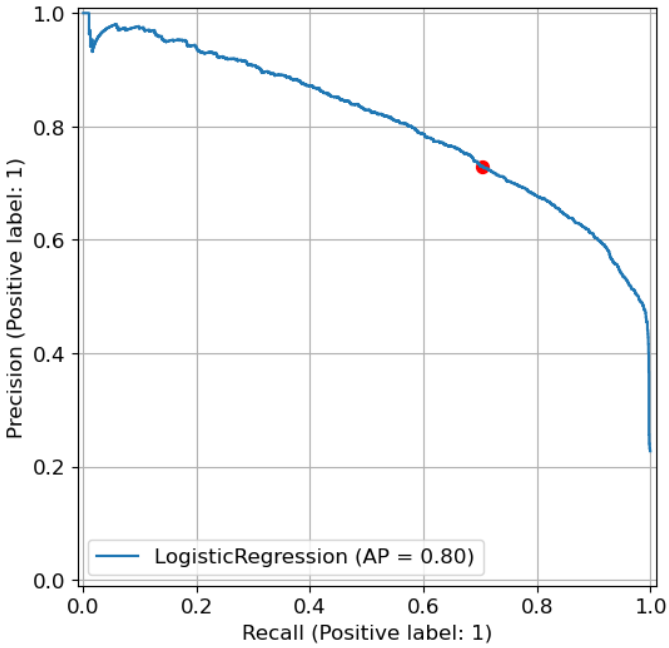

Fig. 4. Base Model Precision Recall

B. Brute Force Feature Contributions

Similar to the linear regression model, I employed a brute force approach to feature contributions and further enhanced accuracy. Again, I removed the features that hurt the accuracy and got an accuracy of 0.877. The full table is "Table 8".

TABLE VIII
IMPACT OF FEATURE REMOVAL ON MODEL PERFORMANCE

Feature Removed	Accuracy
person_age	0.875
person_gender	0.878
person_education	0.873
person_income	0.874
person_emp_exp	0.878
person_home_ownership	0.876
loan_amnt	0.875
loan_intent	0.875
loan_int_rate	0.855
loan_percent_income	0.872
cb_person_cred_hist_length	0.877
credit_score	0.874
previous_loan_defaults_on_file	0.827

C. Optimal Proportional Accuracy

I used the precision recall curve from above, as well as a threshold value, to try and obtain a better proportional accuracy. The first model can be seen in "Fig. 5" where we notice a flip. Before, the model was biased to choose 0, but now the biased to select 1. I used the precision/recall and threshold curve in "Fig 6." to select a more optimal threshold, which can be seen in "Fig. 7" where true values 0 and 1 have approximately 80% accuracy respectively, with a total accuracy of 0.814. So while this model has a lower accuracy, it is a better model because it isn't biased towards selecting one label.

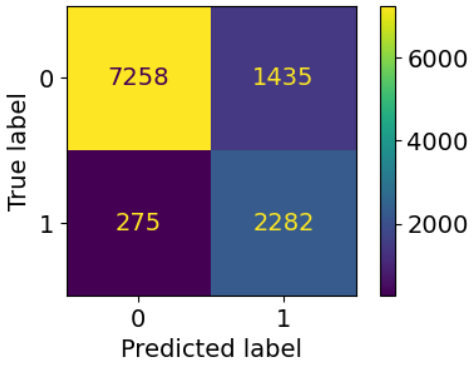

Fig. 5. Confusion Matrix

IV. SHAP - FEATURE CONTRIBUTION

SHAP (SHapley Additive exPlanations) is a method for interpreting model predictions by explaining how much each feature effects the prediction.

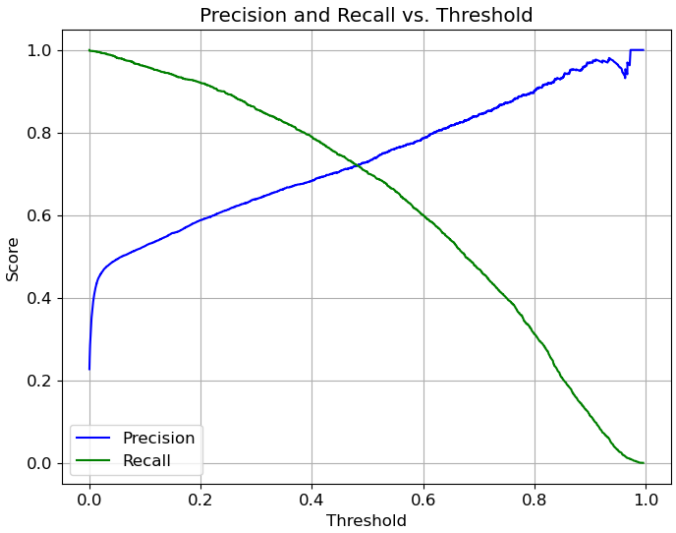

Fig. 6. Precision/Recall vs. Threshold

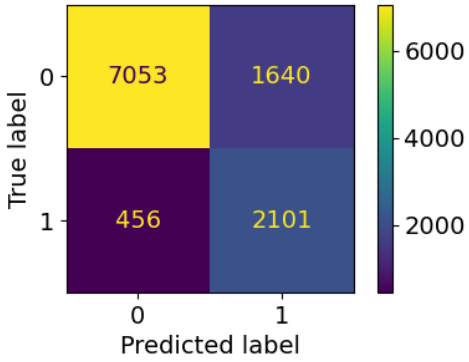

Fig. 7. Proportional Model Confusion Matrix

A. Housing Data

"Fig. 8" displays the SHAP for the housing dataset, indicating the top two features are latitude and longitude, while the bottom two features are ocean proximity and total rooms. The feature contributions are spread out fairly linearly.

B. Loan Data

"Fig. 9" displays the SHAP for the loan dataset, indicating the top two features are previous loan defaults on file and loan interest rate, while the bottom two features are gender and credit history length. The features contributions are exponential, as previous loans accounts for nearly all of the feature contributions, while the rest of the features have minimal impact.

V. ADVANCED MODELS

The advanced models that will be discussed are Decision Trees, Random Forest, Gradient Boosting, and Neural Networks. I employed each of these on both the housing and loans datasets. While each model has advantages and disadvantages for both datasets, I employed all of these models to gain experience using each with different datasets.

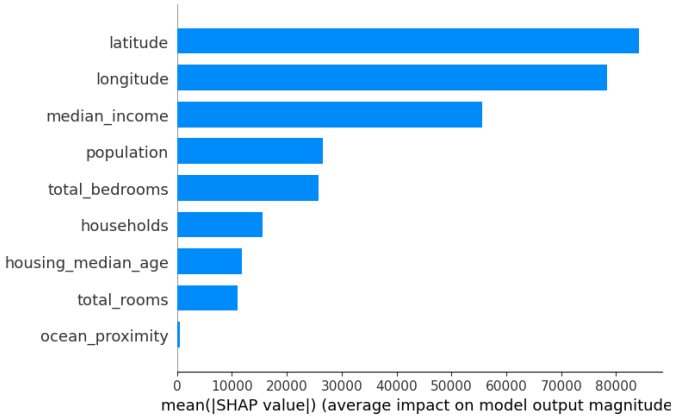

Fig. 8. SHAP of Housing Data

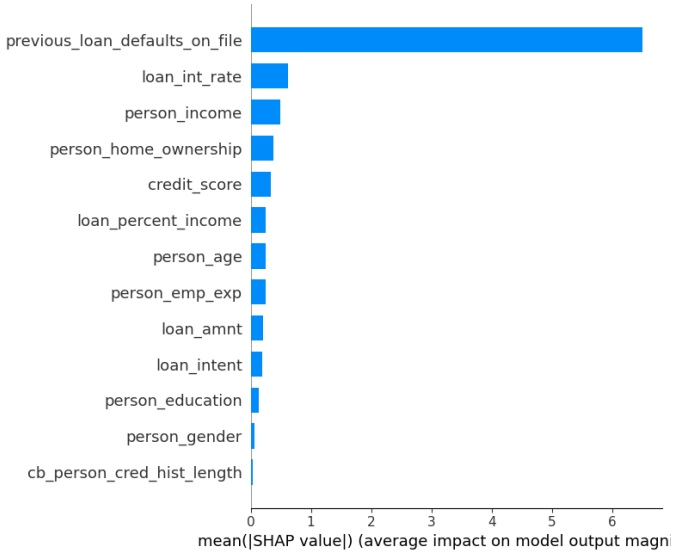

Fig. 9. SHAP of Loan Data

A. Housing Data

Decision Trees are the first method I will discuss, and to be quite transparent, I experimented the least with this model. I experimented with varying depths, and the results can be seen in "Table 9" where the best depth was 10 and R^2 of 0.728.

TABLE IX
MODEL PERFORMANCE OF DECISION TREES AT VARIOUS DEPTHS

Depth	R^2 Score	RMSE
1	0.311	96,461.45
5	0.635	70,268.91
10	0.728	60,617.74
15	0.669	66,874.51
20	0.651	68,692.89

I also employed a basic Random Forest model, which initially had an R^2 score of 0.793 when the max depth was set to 10 with 100 estimators. I again experimented with the depths, and found the best depth to be 30. The results are in "Table 10". Finally, I increased the estimators to 200 and noticed an R^2 score of 0.821. Similarly, I noticed the CV score for this model was 0.811, roughly the same as the R^2, indicating the model wasn't overfitting.

TABLE X
MODEL PERFORMANCE OF RANDOM FOREST AT VARIOUS DEPTHS

Depth	R^2 Score	RMSE
5	0.674	66,767.52
15	0.817	50,115.86
20	0.820	49,642.14
30	0.820	49,614.54

From here, I implemented a Gradient Boosting Regressor model. I experimented with different learning rates, finding the best to be 0.25, and also experimented with depths while holding the learning rate constant. The best depth was 5, resulting in an R^2 of 0.825, which would end up being the best of all the models. For more details on the metrics, look to "Table 11" and "Table 12".

TABLE XI
GRADIENT BOOSTING PERFORMANCE AT VARIOUS LEARNING RATES

Learning Rate	R^2 Score	RMSE
0.01	0.483	83,614.92
0.1	0.772	55,495.91
0.25	0.805	51,275.63
1	0.771	55,680.08

TABLE XII
GRADIENT BOOSTING PERFORMANCE AT VARIOUS MAX DEPTHS

Depth	R^2 Score	RMSE
3	0.805	51,275.63
5	0.825	48,562.99
10	0.815	50,020.40
20	0.706	63,043.39

Finally, I experimented with some different Neural Network architectures. I began with a basic model with two layers, with RELU as my activation and MSE as my error. I observed an R^2 of 0.787 for this initial model. I then implemented a bigger model with three layers, keeping the activation and error the same, as well as a smaller model, with two layers again but not as wide. I noticed overfitting in my second model, and so I concluded with a model that had four layers, but I included regularizers and dropout layers. The results for these four models can be seen in "Table 13". The best model was the fourth model, which still wasn't as good as Gradient Boosting with an R^2 of 0.810.

B. Loan Data

Again, starting with Decision Trees, the best depth observed was 10 with an accuracy of 0.919. Other trained depths can be seen in "Table 14".

My base Random Forest had an accuracy of 0.921 when the max depth was set to 10 with 100 estimators. I again

TABLE XIII
NEURAL NETWORK CONFIGURATION AND PERFORMANCE - TRANSPOSED

Metric	Model 1	Model 2	Model 3	Model 4
Layers	2	3	2	4
Layer Dims	(64, 32)	(128, 64, 32)	(32, 16)	(256, 128, 64, 32)
Dropout	None	None	None	0.1
Regularizer	None	None	None	L2(0.0001)
Epochs	100	100	100	200
Loss	0.164	0.122	0.198	0.204
MSE	0.164	0.122	0.198	0.173
Val Loss	0.196	0.197	0.220	0.229
Val MSE	0.196	0.197	0.220	0.198

TABLE XIV
CLASSIFICATION ACCURACY OF DECISION TREES AT VARIOUS DEPTHS

Depth	Accuracy
1	0.773
5	0.913
10	0.919
15	0.912
20	0.902

experimented with the depths, and found the best depth to again be 30. The results are in "Table 15". Finally, I increased the estimators to 200 and noticed an accuracy of 0.926.

TABLE XV
CLASSIFICATION ACCURACY OF RANDOM FORESTS

Depth	Accuracy
5	0.913
15	0.923
20 3	0.925
30	0.925

Similar to the Gradient Boosting Regressor, I experimented with varying learning rates and depths for the Gradient Boosting Classifier, with the best accuracy of 0.932 when the depth was set to 5 and learning rate was set to 0.25. The findings for both results are in "Table 16" and "Table 17".

TABLE XVI
CLASSIFICATION ACCURACY OF GRADIENT BOOSTING AT VARIOUS LEARNING RATES

Learning Rate	Accuracy
0.01	0.893
0.1	0.922
0.25	0.927
1	0.927

For my Neural Networks, I used the exact same architectures as I did for the housing dataset. I observed the same overfitting in the second model, and noticed the best accuracy in model 4 with an accuracy of 0.910, which again wasn't as good as the Gradient Boosting model. The results for these models are demonstrated in "Table 18".

TABLE XVII
CLASSIFICATION ACCURACY OF GRADIENT BOOSTING AT VARIOUS DEPTHS

Depth	Accuracy
3	0.927
5	0.932
10	0.931
20	0.926

TABLE XVIII
NEURAL NETWORK CONFIGURATION AND PERFORMANCE - TRANSPOSED

Metric	Model 1	Model 2	Model 3	Model 4
Layers	2	3	2	4
Layer Dims	(64, 32)	(128, 64, 32)	(32, 16)	(256, 128, 64, 32)
Dropout	None	None	None	0.1
Regularizer	None	None	None	L2(0.0001)
Epochs	100	100	100	200
Loss	0.937	0.980	0.928	0.915
MSE	0.141	0.052	0.162	0.214
Val Loss	0.907	0.889	0.908	0.907
Val MSE	0.217	0.507	0.191	0.212

VI. CLOSING REMARKS

In this final section I will discuss the best results, as well as future improvements for each of these datasets.

A. Housing Data

The best model was the Gradient Boosting Regressor with an R^2 of 0.825 and RMSE of 48,562.

One thing I would hope to explore further would be better understanding the homes that are best/worst predicted. I noticed the endpoints were the most problematic points being predicted within the linear regression models, but I would be curious if this same thing holds for the more advanced models. Similarly, I would like to explore if there are underlying similarities in these homes being best/worst predicted, whether that is certain features or intrinsic properties within the dataset. Finally, I would like to retrain some of the more advanced models using one-hot encoding instead of label encoding. While I only used label encoding for one feature in this dataset, I have observed some models perform better with one-hot encoding — Neural Networks in particular.

B. Loan Data

Like the housing data, the best model was the Gradient Boosting Classifier with an accuracy of 0.932.

To reiterate what I stated above, in the future I would like to use one-hot encoding for object datatypes instead of a method similar to label encoding. This would certainly be more significant for this dataset considering there were multiple of these label encoded features. Also, I would like to do a Confusion Matrix for my more advanced models to see if they are favoring one label over another, as was evident in the logistic regression. Finally, I would like to experiment more with the Neural Network architectures to see if the model could be further optimized.