Project report: Housing Price Prediction

Sameek Bhattacharya [11776897]
sameekbhattacharya@my.unt.edu

Harshavardhan Sasikumar [11800735]
harshavardhansasikumar@my.unt.edu

Ram Gopal Anne [11662303]
RamGopalAnne@my.unt.edu

➢ Introduction

The housing industry is one of the the important sectors of the economy as property values are associated with individual wealth and the economy. These figures are not easy to estimate mostly due to location, number of rooms, and population density among other variables. Looking at this task from a different stand point, the challenges posed by the ever continuous and ever-changing reality of the real estate market is enough to confuse anyone! However, the field of artificial intelligence and more specifically machine learning has proven to be a robust and influential approach to the world of complex data analytics, data mining and reliable cost estimates.

Such information includes things like mean earnings, age variables of the property, jurisdiction and so on. The targets here are to build and deploy predictive models which can be relied upon in addressing real-world issues mostly concerning the valuation of assets and may address or relevant for forecasting of the possible future occurrences. This project's emphasis is on the creation of a development machine learning model to estimate the selling price of houses in California, given the California Housing Dataset.

The data set that contains housing price covering house features such as median income and property age and geographical coordinates and other salient house value determinants would be used. The aim is to construct a valid model that can apply to real use cases and be able to make valuation of the properties.

Problem Statement:

The purpose of this project is to create an effective machine learning model that can consistently foretell the housing prices in the state of California. The pricing of houses is subjected to a number of determinants which include the area, the type of house, its age, degree of amenities available and demographic factors like population and income distribution. The focus is to research and model the interactions among these variables within the context of California Housing Dataset, which is optimal in depicting these features, with the target being the average house value.

➢ Data Description and Preprocessing

The California housing dataset is a standard and widely used dataset for regression task such as predicting the housing price of California state. It has several attributes to do with social economic characteristics of California neighborhoods based on 1990 census.

The dataset has 8 attributes one of which is numeric while the rest have been scaled to simplify the process:

- MedInc: The neighborhood's median income (in tens of thousands of dollars).
- HouseAge: The neighbourhood’s average age of homes and other buildings (in years).
- AveRooms: Mean number of rooms per household
- AveBedrms: Measured average bedroom count per household.
- Population: Number of people living within neighbourhood.
- AveOccup: Average family size/number of occupants per flat.
- Latitude: Neighborhood latitude (north or south distance of the equator).
- Longitude: Distance from the neighbourhood to the poles.

Target:

- **MedHouseVal**: Median house value for the neighborhood (in hundreds of thousands of dollars).

Size:

- **Data Points:** 20,640 districts.
- **Features:** 8

➢ Feature Engineering

In this project, we have checked for missing values in each of the columns. There were no missing values in the dataset. The dataset only consisted of numerical data of different scales. We have performed feature scaling using StandardScaler to scale the dataset using their mean and standard deviation.

The formula used for StandardScalar is:

$z = (x - \mu) / \sigma$, where

x: The original feature value.

μ: The mean of the feature values in the training data.

σ: The standard deviation of the feature values in the training data.

z: The standardized feature value.

Then we have split the dataset into 82 percent for training and 9 percent for validation and another 9 percent for testing purposes. After that we have trained the models.

➢ Model Building and Evaluation

We built four models to predict the housing prices for this dataset. The models are:

1. Linear regression
2. Gradient Boosting regression
3. AdaBoost Regressor
4. Neural network regression

Metrics used:

Mean Absolute Error (MAE):

The mean absolute error, or MAE, calculates the average size of errors between expected and actual values without taking direction into account. It offers a clear understanding of prediction accuracy and is both intuitive and resilient to outliers.

Mean Squared Error (MSE):

The average of squared errors between expected and actual values is assessed by the Mean Squared Error (MSE). Larger deviations are penalized more severely when the errors are squared, which makes it sensitive to outliers and helpful for pointing out important prediction errors.

R^2 Score (Coefficient of Determination):

The coefficient of determination, or R2 score, quantifies how closely the predictions match the real data. It shows the percentage of variation that the model can account for; a value nearer 1 denotes greater performance. The R2 value is especially useful for comparing models.

We initially trained the models with the training set and validated on the validation set and calculated MAE, MSE and R^2 score for all the models.

Models:

Linear regression:

It is an easy-to-understand model that assumes that the features and the target have a linear connection. To determine the best-fit line, it minimizes the total of squared errors, offering a standard against which more intricate models can be compared.

We fit the model and got MSE around 0.264 for the training set and 0.812 for the validation set.

Gradient Boosting Regressor (GBR):

The Gradient Boosting Regressor (GBR) is an ensemble technique that produces models one after the other, fixing the mistakes of the earlier models. By concentrating on reducing residual mistakes, it builds a strong learner by combining several weak learners, typically decision trees, to achieve high accuracy.

We used n_estimators = 250 for this model.

AdaBoost Regressor (ABR):

Another ensemble technique that combines several weak learners is the AdaBoost Regressor (ABR), which gives data points with higher prediction mistakes greater weights. It provides reliable predictions by dynamically adjusting the training emphasis to lower bias and variation.

We used n_estimators = 250 for this model.

Neural Networks:

An umbrella of models that follows the construction of the human brain which consists of multiple layers of neurons which have activation functions that processes information. In this project, models of small, medium, and large complexities were implemented so as to take care of the non-linearities in the data that was collected. Neural networks are very effective in pattern recognition but they are very prone to overfitting and thus require a lot of parameter adjustment.

- We constructed three neural network models with increasing complexity levels in order to forecast real estate prices. Simple Neural Network (Simple_NN) comprises one input layer with 8 input features, a single hidden layer with 2 neurons that utilize ReLU activation, and one output layer with one neuron for the linear output prediction.
- The model utilizes the Adam optimizer with a learning rate of 0.1, MSE loss, and RMSE as performance metric, while trained for 100 epochs. This simple structure can be inadequate to learn the complicated patterns because of very low hidden layer size. The Medium Neural Network (Medium_NN) adds up the architecture with presence of 2 hidden layers (32 and 16 ReLU activated neurons), placing it at a more capable position of learning from the data whilst still being simple.
- The same hyperparameters and training settings are employed just like in the case of the simple model. Finally, the Large Neural Network (Large_NN) contains a deeper architecture consisting of four active hidden layers with 256, 128, 64 and 32 neurons respectively.
- This model is capable of learning and capturing intricate structures in the data but however is more prone to overfitting because of it's complexity. All models use model checkpointing to save models of best performance avoiding the overfitted models.

➢ Result Analysis

Comparing the mean squared error of the training and validation set.

Models	MSE of the training set	MSE of the validation set
Linear regressor	0.264	0.812
Gradient boost regressor	0.146	0.169
Adaboost regressor	0.468	0.417
Simple neural network	0.587	0.456
Medium neural network	0.566	0.437
Large neural network	1.015	0.775

- Here, let us examine the MSE of the training set relative to the MSE of the validation set. The trend that should be achieved is the one where the validation set's MSE and the training set's MSE have a smaller distance.

- For instance, it is possible to determine that the validation MSE for the linear regressor is considerably greater than its training set MSE which may be evidence of the model being over fitted.

- Furthermore, models such as the large neural network tend to have an MSE greater than 1 resulting in high degree of accuracy in the cases where data is poorly fitted to a model.

- As per the table we have seen that the Gradient Boost regressor has performed better than all the other techniques with the least MSE and 'almost no' variance between validation MSE and training MSE. Hence, it is most likely to offer better accuracy.

- In comparison of results among different models, it is necessary to provide the absolute value of MSE and the difference between the training and validation MSE score as well. Low differences between both the values in the presence of low overall MSE implies that the model is performing well.

Comparing other metrics for various models:

Models	Mean absolute error	R^2 score
Linear regressor	0.464	0.293
Gradient boost regressor	0.264	0.812
Adaboost regressor	0.570	-0.356

Inference:

- Between the models that we proposed in this study, the Gradient Boost regressor was by far the more accurate and more reliable one as it recorded the lowest MAE (0.264) and the highest R^2 score of 0.812 and can predict the target variable to the extent of 81.2% whereby reasonable expectations can be placed on the predictive power of this model.

- For using the Linear regressor, an MAE of 0.464 and an R^2 at 0.293 can be recorded. In comparison to the Adaboost regressor one, it goes without saying - it performs better. Restrict in the model high waste however, this only explains the variance of about 29.3% suggesting that it has poor prediction power and does not capture non-linear relationships in the data well.

- It has been provided adequate information and the Adaboost regressor has simple model which in any case has the maximum number of absolute errors of 0.570 also included a model with a negative R^2 score -0.356. The simplest of all models should achieve some predicted value R under these conditions, even if no relationship exists only with the mean, predicting average would beat expectations in some cases so negative R^2 tells everything.

- Models such as Linear and Adaboost have proven to be ineffective as such models achieved a high R^2 do not expect because they do not possess flexibility that is quite sufficient to rise above and capture the non-linear patterns in the data. For this reason, modeling small data sets with complex relationships is necessary with the use of ensemble methods such as Gradient Boost.

➢ Conclusion

Overall learnings from the project:

- With the lowest MAE 0.264 and the highest R^2 (R squared) score of 0.812, the Gradient Boost Regressor is the best model as it captures the relationships included in the data with a lot of precision as well as reliability.

- Both Linear regressor and Adaboost regressor show below average performance as the Adaboost regressor outperformed by the linear one and has negative R^2 (R squared) score meaning this one performed the worse. This goes to show that in datasets where non-linear patterns occur, more robust models such as the Gradient Boost model should be used.

- While ensemble approaches (Gradient Boosting and AdaBoost) demonstrated the capacity to combine weak learners to attain high accuracy, the exploration of linear regression as a baseline model underlined the significance of simplicity and interpretability.

- In addition to showcasing its ability to manage intricate patterns, neural networks also highlighted the significance of meticulous architecture design and hyperparameter tweaking.

- The process of model standardization resulted in better performance, which indicated the necessity of preprocessing in allowing for fair comparisons and reliable training of models. This step was especially important with models that are gradient searched such as the neural networks.

- Adopting metrics like MAE, MSE, R^2 and RMSE made it easier to evaluate the performance of each model. Such evaluation and analysis approach were very valid throughout the project as there are large error sensitivity (MSE/RMSE) and interpretability (MAE) relevant to model comparison (R^2).

- We learnt that tracking performance via the training, validation, and testing sets enabled the possibility of discovering the overfitting nature of the model (more so with the larger neural networks), as well as the necessity for having validation to adjust the hyperparameters of the model.

Future possible enhancements to the project:

- For further assessment, k-fold cross validation should be added to reduce variance associated with a single train-test split.
- Try to include SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable model agnostic Explanations) so that you can give more explanations about the model predictions.

- Utilize more advanced algorithms that center for tabular data like XGBoost for better performance.
- Try also more complex architectures such as transformers or AutoML solutions for automatic creation of models.
- Design an interface for the predictions using a web application framework like Flask or FastAPI.

Contribution of each member in the project

Sameek: Worked on Gradient boosting and linear regression models. Analysed the models using various metrics and performed relevant preprocessing steps.

Harsha: Worked on Neural network models (tuned the hyper parameters for better accuracy). Also worked on adaboost regressor and some of the feature engineering steps.

Ram Gopal Anne: Analysed the result of various models and worked on adjusting hyper parameters for all the models.

➢ References

- Scikit-learn. (n.d.). Fetch California housing dataset. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html
- Scikit-learn. (n.d.). AdaBoost regressor. Retrieved from https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
- Scikit-learn. (n.d.). Linear regression. Retrieved from https://scikit-learn.org/dev/modules/generated/sklearn.linear_model.LinearRegression.html
- TensorFlow. (n.d.). Regression with Keras. Retrieved from https://www.tensorflow.org/tutorials/keras/regression
- Scikit-learn. (n.d.). Gradient boosting regression. Retrieved from https://scikit-learn.org/1.5/auto_examples/ensemble/plot_gradient_boosting_regression.html
- Turing. (n.d.). Data normalization with Python and Scikit-learn: Tips & tricks for data science. Retrieved from https://www.turing.com/kb/data-normalization-with-python-scikit-learn-tips-tricks-for-data-science
- Pace, R. K., & Barry, R. (1997). Sparse spatial autoregressions. Statistics and Probability Letters, 33(4), 291-297. https://doi.org/10.1016/S0167-7152(97)00016-4