PROJECT: Machine Learning

California Housing Price Prediction

Purpose of The Document

This document specifies requirements for the project “California Housing Price Prediction”. Apart from the functional and non-functional requirements of the project, it also serves as an input for project scoping.

Problem Statement

The purpose of the project is to predict median house values in Californian districts, given many features from these districts.

We begin by building a model of housing prices in California using the California census data. You are provided with the following resources that can be used as inputs for your model:

1. Data with metrics such as the population, median income, median housing price, and so on for each block group in California. This model should learn from data and be able to predict the median housing price in any district, given all the other metrics.
2. Districts or block groups are the smallest geographical unit for which the US Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

Project Guidelines

#	Step	Process
1	Intiation	Define save_fig function to save the plots to disk.
2	Fetch data	1. Write code to fetch California housing data from: https://raw.githubusercontent.com/ageron/handson-ml/master/. The housing data URL under the github folder is stored at datasets/housing/housing.tgz. 2. Fetch and load data in the housing list. 3. Check the housing.head() to review the first few rows of the dataset. 4. Review housing.info() to check other housing related details.
3	Describe data	1. Run housing.describe() to review count, mean, standard deviation, min, max, and percentile values (25%, 50%, 75%) for each column. 2. Ensure that the null values are ignored. 3. The standard row shows the standard deviation, which measures the dispersion of values. 4. The rows with titles 25%, 50%, and 75% show the corresponding percentiles.
4	Plot histograms	Plot histograms for each column which indicate the frequency of different column values.
5	Split data	Use Scikit-Learn library function train_test_split to split the data into 80% training data and 20% test data.
6	Limit the data stratum for median income values	1. To create limited and workable stratums or intervals of median income, derive income_cat feature from median income and within this, mark the ones that are above category 5. 2. This is to eliminate the long tail but limited data at the end of the income_cat scale.
7	Stratified split	1. Stratified data preserves the relative proportion of data categories in the derived data. For example, trying to preserve relative percentage of men vs. women in sampled data. 2. Call the function StratifiedShuffleSplit to perform stratified split of data with respect to income_cat column, so that relative proportion of income categories are preserved in training and test dataset. 3. Compare the preservation of relative proportion of income categories when random split was done vs. when stratified split was done. Stratified split seems to be better at preserving the relative proportions of income categories after the split.

8	Drop income_category column	4. Drop the income category column (note that this was a column derived before from median income).
9	Visualize data	1. Visualize data on the map of California, with respect to longitude and latitude. 2. Use color code to indicate various values of median house value on this graph. The radius of each circle represents the district's population, and the color represents the price.
10	Create scatter plot between various columns	1. Create the scatter plot of these columns with respect to to each other: "median_house_value", "median_income", "total_rooms", "housing_median_age" 2. Create a scatter plot of these columns with respect to each other: "median_house_value", "median_income"
11	Data preparation	1. Create new attribute columns such as, rooms_per_household, bedrooms_per_room and, population_per_household. 2. Drop house value column from the training data. 3. Fix the missing values of total_bedrooms attribute with either of these options: a. Get rid of the corresponding districts. b. Get rid of the whole attribute. c. Set the values to some value (zero, the mean, the median, etc.). 4. Apply Imputer to fill the missing columns with the median value for that column. 5. Convert text label attribute ocean_proximity to a number using OneHotEncoder.
12	Combining attributes	1. Combine some attributes to see if they help in ML algorithm. 2. Create a CombinedAttributesAdder class to add this attribute along with rooms_per_household and population_per_household, which are very useful attributes. 3. Add bedrooms_per_room hyperparameter to find out whether adding this attribute helps the Machine Learning algorithms or not.
13	Feature scaling	1. Apply Scikit-Learn function StandardScaler to scale the features in housing_num, which represents all those columns of the housing dataset, but where all columns are numeric in nature. So, it does not contain ocean_proximity feature which was categorical in nature. 2. Create a function pipeline using Pipeline class of Imputer + CombinedAttributesAdder + StandardScaler

14	Feature Union	1. Apply the function FeatureUnion to combine the function pipelines of numerical attributes with that of categorical attributes. 2. Fit and transform 3. Housing data is now ready for Machine Learning
15	Select and train the model	1. Try Linear Regression on scaled dataset. 2. Check MSE (mean square error) and MAE (mean absolute error)
16	Decision tree regression	1. Try Decision Tree Regression on scaled dataset. 2. Check MSE (mean squared error)
17	Use cross validation	1. Apply k-fold cross validation 2. This model randomly splits the training set into 10 distinct subsets called folds. 3. Then it trains and evaluates the Decision Tree model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The result is an array containing the 10 evaluation scores.
18	Random Forest and SVM Regression	1. Try Random Forest Regression. 2. The results from Random Forest are better than Linear Regression or Decision Tree Regression. 3. However, note that the score on the training set is still much lower than on the validation sets, meaning that the model is still overfitting the training set. Possible solutions for overfitting are to simplify the model, constrain it (i.e., regularize it), or get a lot more training data. 4. Try SVM Regression.
19	Find ideal hyperparameters	1. Use GridSearchCV to find ideal hyperparameters. 2. For Random Forest regressor, clearly the minimum mean square error is for n_estimators (no of trees attempted in Random Forest) = 30 and max_features = 8. So that is the ideal hyperparameter combination for Random Forest regressor. 3. Running GridSearchCV will take some time as it has to try a very large no of hyperparameter combinations. 4. Use RandomizedSearchCV to find ideal hyperparameters. This has better performance than GridSearchCV class when the hyperparameter search space is large.