Warsaw University of Technology

FACULTY OF
ELECTRONICS AND INFORMATION TECHNOLOGY

EARIN

Laboratory 4
Group 21
Variant 1
Regression and Classification

Tymon Żarski 310992
Bartosz Peczyński 310703

WARSAW April 18, 2024

Contents

1. Introduction 3

1.1. Assigned task 3

1.2. Chosen models 3

2. Model creation 4

2.1. Data preparation 4

2.2. Finding best parameters of the models 4

2.3. Found best parameters 5

2.4. Model training 6

3. Summary 8

1. Introduction

1.1. Assigned task

Write a model to predict house prices on the California housing dataset.

The task is to develop a model to predict house prices using the California housing dataset. Firstly, data preparation involves analyzing features and removing redundant ones, such as those dependent on each other, and normalizing the dataset. The data is then split into training and testing sets. Next, two models are chosen for fitting, linear regression and random forest regression, which are suitable for regression tasks. The choice of models is explained in the report. Model training involves selecting an appropriate evaluation metric and trying different parameters for each model using a cross-validation framework with 4 folds. The parameters tried for both models are documented in separate tables, and the performance of both models with the best parameters is compared.

1.2. Chosen models

We could choose among many regression models that might have been suitable for our task. We selected two, amongst which the first one is simple and fast: **linear regression** model. Although we suspected it would not provide good results compared to more complex models, we wanted to see what this difference looks like. The second algorithm we selected is **Random Forest Regression** model, which is more complex and slower but is expected to provide better results.

2. Model creation

2.1. Data preparation

We made a PCA analysis on a dataset, and we determined that 6 out of 8 columns almost entirely cover the Explained Variance Ratio. This can be seen on Figure 2.1. We then tried to estimate which components are correlated with others so they can be removed. For this reason, we have removed two columns: AverageOccupation and AverageBedrooms. Both of those columns are connected with the already existing column "AverageRooms" and another, "Population", which have been preserved. We received three datasets on which we did the experiments:

1. Original dataset,
2. Dataset from PCA analysis,
3. Preprocessed Dataset (with dropped two columns and scaled).

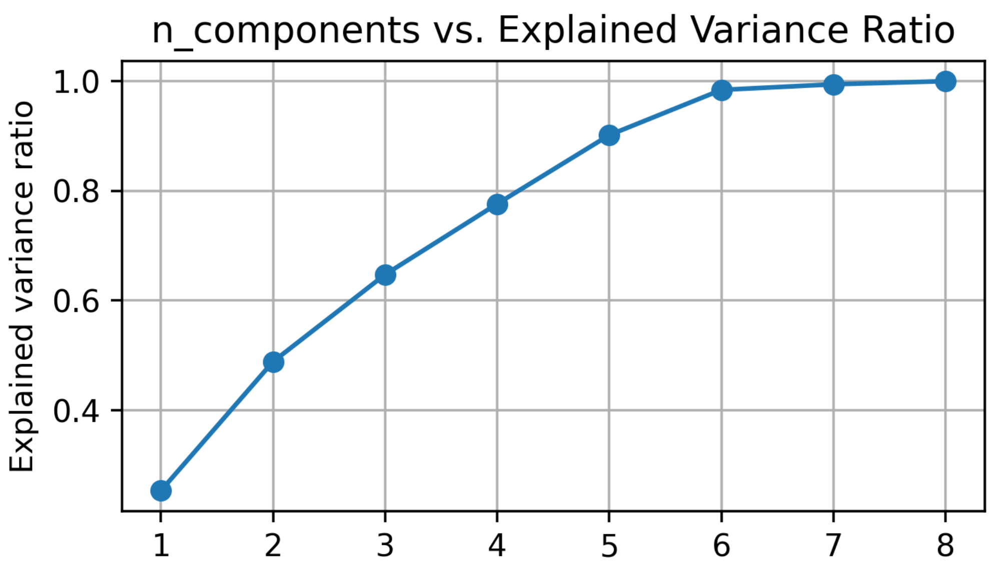

Figure 2.1. Enter Caption

2.2. Finding best parameters of the models

To find the best parameters for all models, we have written a solution for finding their values. It uses a Linear Regression model and a Random Forest regression model, and it tests them on three different datasets: the entire training dataset, its modification that has been transformed using Principal Component Analysis (PCA), and a preprocessed dataset, in which we dropped two columns. Here's a step-by-step explanation:

1. If the *should_test_parameters* flag is set to True, the code will test the models with different parameters,
2. It defines a list of models and their parameters to test. The linear regression model tests whether or not it fits the intercept. The Random Forest Regression Model tests different numbers of estimators, maximum depths, and minimum sample splits,

3. It defines a list of datasets to test the models. Each dataset is a tuple containing the training and testing data, as well as a name for the dataset,
4. It prints the name of each dataset and then initializes empty dictionaries to store the best scores and parameter values,
5. For each model, it gets the model object and the list of parameters to test,
6. For each set of parameters, it iterates over the parameters and their values,
7. It sets each parameter's value to that value on the model, then processes it by evaluating it using cross-validation. If the found score is better than the score already in the dictionary, it updates this score and parameter value,
8. After testing all parameters for a model, it prints a message indicating that it has finished testing that model.

Here's the pseudocode in LaTeX format:

Algorithm 1 Test Model Parameters

```
if should_test_parameters is True then
    for all dataset in datasets do
        Print name of a dataset
        for all model_entry in models do
            for all parameter_set in model_entry.parameters do
                for all parameter, values in parameter_set do
                    for all value in values do
                        If the value provides a better score than the previous value,
                        set parameter value in the best parameters dict.
                        If the parameter is worse, we break the loop
                    end for
                end for
            end for
            Print best parameters found.
        end for
    end for
end if
```

This pseudocode represents the logic of the Python code in a simplified and language-agnostic way.

2.3. Found best parameters

After the performed experiments, which can be found on: Table 2.3, Table 2.4, Table 2.5, we have chosen three best parameters: criterion equal to *"squared_error"*, max depth equal to *1000*, minimal number of samples splits equal to *4*, number of estimators equal to 100 and as empirical good default value for regression problems we chose maximal number of features as number of features in the preprocessed dataset. The trained model scores are presented below. We have also seen that increasing the number of estimators or changing the criterion from *"squared_error"* to *"absolute_error"* increases the resulting score, but also the time needed for training.

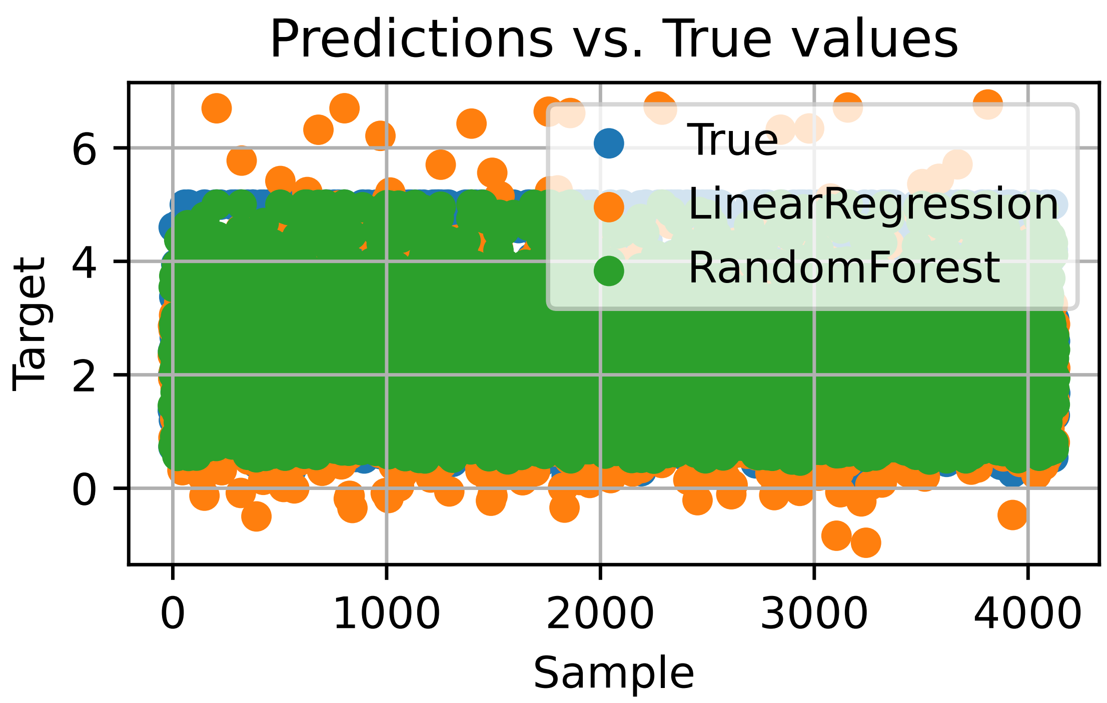

Figure 2.2. Visual difference between predicted and true values

Table 2.1. Model Performance - squared_error

Model	**MSE**	**R2**
Linear Regression	0.5514	0.5771
Random Forest	0.2496	0.8086

Table 2.2. Model Performance - absolute_error

Model	**MSE**	**R2**
Linear Regression	0.5514	0.5773
Random Forest	0.2387	0.8168

2.4. Model training

We selected to use our preprocessed dataset for training, as it provided the best cross-value score for the Random Forest model, and we wanted to compare the quality of the models run on the same dataset. Results can be seen on the Table 2.2.

Table 2.3. Model Performance on Full Training Dataset

Model	**CV score mean**	**CV Score std**
LinearRegression	0.6052	0.0068
RandomForest n_estimators: 500	0.7998	0.0065
RandomForest (n_estimators: 500)	0.7998	0.0065
RandomForest (n_estimators: 100)	0.7999	0.0048
RandomForest (n_estimators: 10)	0.8002	0.0052
RandomForest (max_depth: 1000)	0.7997	0.0056
RandomForest (max_depth: 100)	0.8004	0.0056
RandomForest (max_depth: 10)	0.8007	0.0058
RandomForest (min_samples_split: 6)	0.7996	0.0067
RandomForest (min_samples_split: 4)	0.8003	0.0050
RandomForest (min_samples_split: 3)	0.7998	0.0053

Table 2.4. Model Performance on PCA Dataset

Model	**CV score mean**	**CV Score std**
LinearRegression	0.5996	0.0047
RandomForest (n_estimators: 500)	0.7744	0.0026
RandomForest (n_estimators: 100)	0.7748	0.0028
RandomForest (n_estimators: 10)	0.7748	0.0025
RandomForest (max_depth: 1000)	0.7740	0.0039
RandomForest (max_depth: 100)	0.7738	0.0030
RandomForest (min_samples_split: 6)	0.7745	0.0028
RandomForest (min_samples_split: 4)	0.7748	0.0037
RandomForest (min_samples_split: 3)	0.7734	0.0039

Table 2.5. Model Performance on Preprocessed Dataset

Model	**CV score mean**	**CV Score std**
LinearRegression	0.5989	0.0041
RandomForest (n_estimators: 500)	0.8036	0.0062
RandomForest (n_estimators: 100)	0.8041	0.0065
RandomForest (n_estimators: 10)	0.8028	0.0058
RandomForest (max_depth: 1000)	0.8038	0.0046
RandomForest (max_depth: 100)	0.8018	0.0051
RandomForest (min_samples_split: 6)	0.8039	0.0065
RandomForest (min_samples_split: 4)	0.8040	0.0056
RandomForest (min_samples_split: 3)	0.8030	0.0055

3. Summary

We found out that predicting the prices of houses is a task that can be quickly done using regression algorithms. The R2 score was equal to around 0.809, so it means that values computed by our trained model were, on average, different by 19% from the accurate prices. This is surprising, taking into account the fact that it is not possible to precisely predict exact values, as there are many factors which can influence the price, such as the negotiation skills of the buyer or seller.