MINISTRY OF EDUCATION OF REPUBLIC OF MOLDOVA
Technical University of Moldova
Faculty of Computers, Informatics and Microelectronics
Department of Software Engineering and Automatics

Report

on Artificial Intelligence Fundamentals
Laboratory Work nr. 3

Performed by: **Vasile Boaghi, Ion Dodon**, FAF-172
Verified by: **Mihail Gavrilița**, asist. univ.

Chișinău, 2021

Contents

Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, forecasting, or error reduction,[clarification needed] linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.

- If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares cost function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the least squares approach can be used to fit models that are not linear models.

Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.

The Task

Having some unformatted historical data about apartment complexes and their prices, create a program that could predict the price of an apartment given a set of characteristics of the apartment complex it is part of. The program should create a model that would perform linear regression on the available data. The program should be able to predict the *medianCompexValue* based on the given characteristics. After the model is created, it is needed to perform some simple statistical tests to determine the accuracy of the model.

Solution Description

We used the sklearn library to create the linear regression model. The first step was to read the data from the *apartmentComplexData.txt* file. 80% of this data was chosen randomly and it has been used for training purposes and the remaining 20% were used for testing purposes. But before splitting the data, we had to normalize it and for this we used the *preprocessing.MinMaxScaler()* function from the *sklearn* library. The data has been read from the *.txt* file using pandas and we got a *DataFrame* which is a table representation is pandas. We named the columns accordingly as specified in the task paper.

Then, we analyzed the data to see what are the most important columns that will help the most to train the model. For this we plotted on variable on x axis and the $medianComplexValue$ on y axis. And so on for each variable. The most clear graph was with the values from 8th column on x axis. This gave us a clue that the values fro column 8 will help us the most. Also, we demonstrated that $8'th$ column has a bigger correlation with the $medianComplexValue$ by using the $pd_data.corr()$ function. This function shows the correlation between each column.

We also made a general description of the data that includes the following values for each column: mean, std, min, 25%, 50%, 75%, max. This helps us have a general idea about the values from our database.

The next step was to create the model and fit it with those 80% of data. In the process of fitting the model, we used all the variable/columns - expect the last one, because the last one is the $medianComplexValue$.

The we used the remaining 20% of data to make predictions. After have the predictions we used the $r2_score$ function from *sklearn* to determine how well our model performs. And finally we plote the predictions and the true values from the test to see graphically the difference between them.

Code and Mentions

First of all we have to import the libraries that we need in order to perform the linear regression.

- pandas - used to work work with data in a table manner
- numpy - used to work with arrays, numbers
- sklearn - the library that offers us the posibility to create a linear regression model
- sklearn.metrics - functions used to measure the metrics of the linear regresion model
- matplotlib.pyplot - used to plot grafs

Listing 1: Used imports

```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import *
import matplotlib.pyplot as plt
from sklearn import preprocessing
```

We use the package sklearn and its associated preprocessing utilities to normalize the data.

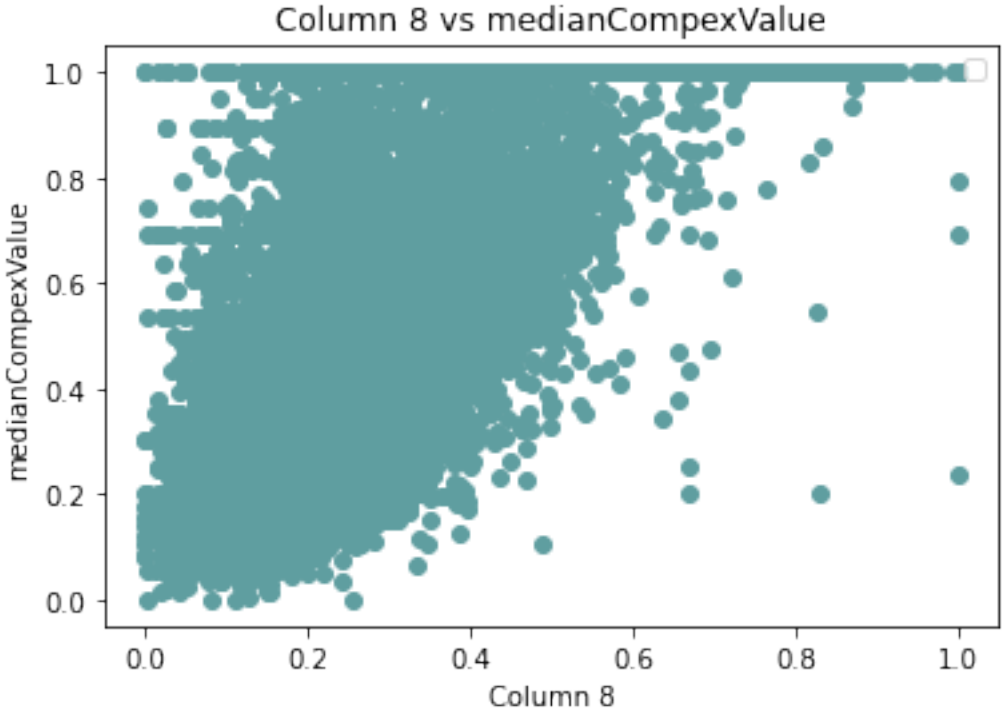
Listing 2: Data normalization

```
data = pd_data.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
data_scaled = min_max_scaler.fit_transform(data)
pd_data = pd.DataFrame(data_scaled)

pd_data.columns = [
        '1',
        '2',
        'complexAge',
        'totalRooms',
        'totalBedrooms',
        'complexInhabitants',
        'apartmentsNr',
        '8',
        'medianCompexValue'
]
```

We ploted on the x axis the values from the 8th column and on the y axis the median complex value. The distribution of values has a clear slope. Or, in other words, there is a positive relation between the $medinaComplexValue$ and the values from the $8th$ column. Clearly we can see there is a correlation between the $8th$ column and the $medianComplexValue$.

Figure 1: The relation between values from 8th column and medianComplexValue



The first thing we need to do before training the model is split the data into a training set and a test set. The training set is what we will train the model on and the test set is what we will test it on. The convention is to have 20% dedicated to testing and the remaining 80% to training, but these are not hard limits.

Listing 3: Splitting the data

```
# take randomly 80% of the data for training purposes
# and the ramaining 20% remain for test purposes
msk = np.random.rand(len(pd_data)) < 0.8
train_data = pd_data[msk]
test_data = pd_data[~msk]
```

We used the $linear_model.LinearReggression()$ from *sklearn* to create our linear regression model. The $fit()$ method is used to train the model. As we can see, all 8 columns were used at the training phase and most important - column 8 should be used, because it has a bigger correlation with the $medianComplexValue$. The $fit()$ method takes two arguments. the first one represents the variables and the second one represents the true values.

Listing 4: Create and fit the model

```
reg = linear_model.LinearRegression()
reg.fit(
    train_data[[
      '1',
      '2',
      'complexAge',
      'totalRooms',
      'totalBedrooms',
      'complexInhabitants',
      'apartmentsNr',
      '8',
    ]],
    train_data['medianCompexValue']]
)
```

Now we use those 20% of testing data to make predictions.

Listing 5: Making predictions

```
test_predictions = reg.predict(
    test_data[[
      '1',
      '2',
      'complexAge',
      'totalRooms',
      'totalBedrooms',
      'complexInhabitants',
      'apartmentsNr',
      '8'
    ]]
)


array([[0.6304773 ],
       [0.34444398],
       [0.35718367],
       ...,
       [0.09704677],
       [0.16952976],
       [0.01845385]])
```

The R-Squared metric provides us a way to measure the goodness of fit or how well our data fits the model. The higher the R-Squared metric, the better the data fit our model.

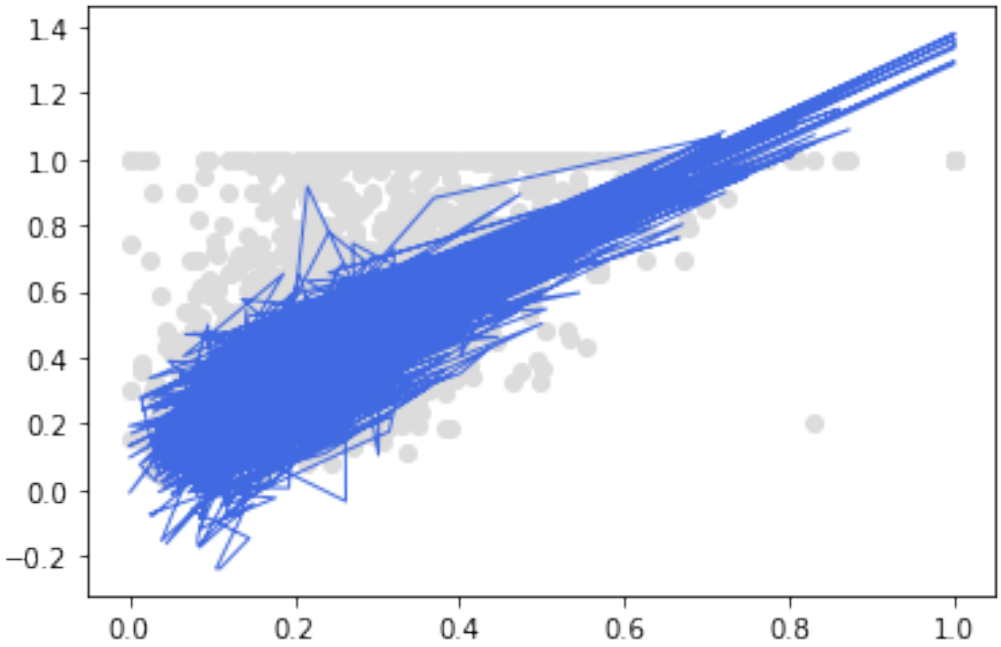
Listing 6: Measure the goodness

```
r2score = r2_score(y_true, y_pred)
print("r2_score = ", r2score)


r2_score =  0.629817486179757
```

We will plot the testing data with its true values and then with blue we will plot the testing data with the predicted values.

Figure 2: Plotting the predictions



Other measurements:

Listing 7: More measurements

```
variance = explained_variance_score(y_true, y_pred)
print("explained_variance_score = ", variance)

max_err = max_error(y_true, y_pred)
print("max_error = ", max_err)

r2score = r2_score(y_true, y_pred)
print("r2_score = ", r2score)

mse = mean_squared_error(y_true, y_pred)
print("mean_squared_error = ", mse)

```

```
mae = mean_absolute_error(y_true, y_pred)
print("mean_absolute_error = ", mae)


explained_variance_score =  0.6298200606688624
max_error =  438435.776657823
r2_score =  0.629817486179757
mean_squared_error =  4903325354.047457
mean_absolute_error =  51059.171527043225
```

Conclusions

Working on this laboratory work we have gained knowledge about how to create a linear regression model. But more than that. We also understood that in order to make a good model first of all we should analyze what data we have, and determine what parts of this data may help the most to train the model. We also found out that it is recommended that the data should be normalized before being used. We familiarized ourselves with different appraches to create a linear regression model. One approach is to used a simple neural network with one single layer. We used instead *sklearn* because it is easier to use and as by our tests is give a better result. We concluded that a neural network for linear regression is more difficult to adjust to our needs.

We have also learned that we have to use part of the given data for training purposes and the other part for testing purposes. Also, we found out that there are functions intended for measuring the goodness of our model. In our case we used the $r2_score$ function.

Now, we know how to create a model with the help of which we can make predictions.

References

[1] Vasile Boaghi, Ion Dodon. Source code for the laboratory work. Accessed February 25, 2021. https://github.com/iondodon/FIA/tree/main/Lab3.

[2] Linear regression https://en.wikipedia.org/wiki/Linear_regression.