FAF.FIA16.1 -- Artificial Intelligence Fundamentals

> **Lab 3:** Linear Regression
> **Performed by:** Furdui Alexandru, group FAF-192
> **Verified by:** Mihail Gavrilita, asist. univ.

Imports and Utils

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
```

Task 1 -- Import your data. Analyze it via common statistical approaches. Cleanse the data if necessary.

```
read_file = pd.read_csv ('apartmentComplexData.txt')
read_file.to_csv ('apartmentComplexData.csv', index=None)

file_data = pd.read_csv('apartmentComplexData.csv', usecols=[2, 3, 4, 5, 6, 8])

file_data.columns = ['complexAge', 'totalRooms', 'totalBedrooms', 'complexInhabi
#clean data by replacing nan values with mean and removing duplicates
file_data.fillna(file_data.mean())
file_data.drop_duplicates()

#split the data into training and testing sets
X = file_data[['complexAge', 'totalRooms', 'totalBedrooms', 'complexInhabitants'
y = file_data['medianCompexValue']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_
```

The goal of this task is to make data more consistent and easy to be trained on. To achieve this, i did 2 things- removed NaN values by replacing them with data mean and removed data duplicates by using "*dorp_duplicates*" function from pandas lib. Also the data was split into training and test sets

Also, in order to complete next tasks, i had to analyze data, and plot the data using common statistical approach. I used the median, mean and standart deviation

```
mean_values = file_data.mean()
median_values = file_data.median()
std_deviation = file_data.std()
```

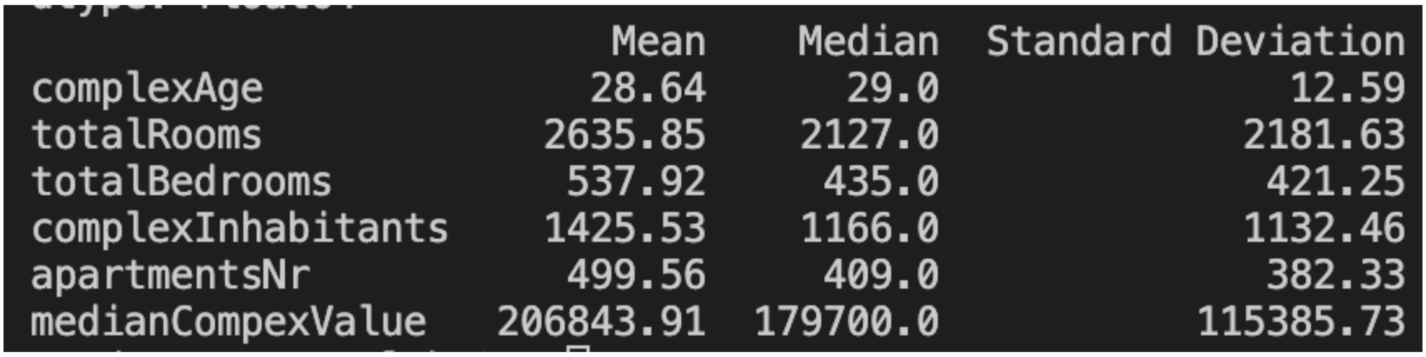
- Mean: this is the average value basically
- Median: this is the middle number of a set of data, a pivot, if data has even length, the median is the average of those 2 values
- Standart deviation: this is a measure of how spread the values are in a dataset. The bigger the value is, it indicates that values are spread out, however the smaller the value is, the closer it is to the mean, which is better in our case as we want ot have the best prediction.

```
mean_rounded = mean_values.round(2)
median_rounded = median_values.round(2)
std_rounded = std_deviation.round(2)

#create the table to display data
data = pd.DataFrame({'Mean': mean_rounded, 'Median': median_rounded, 'Standard D

print(data)
```

As result of this code, we will be shown a table that looks like this:



```
                         Mean     Median  Standard Deviation
complexAge              28.64       29.0               12.59
totalRooms            2635.85     2127.0             2181.63
totalBedrooms          537.92      435.0              421.25
complexInhabitants    1425.53     1166.0             1132.46
apartmentsNr           499.56      409.0              382.33
medianCompexValue   206843.91   179700.0           115385.73
```

Task 2 -- Train your model by applying linear regression.

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

By using the model LinearRegression from sklearn lib, it is 2 lines of code to train your model

Task 3 -- Show the prediction power of your model by attempting to predict the price of a new house

```
linear_prediction = np.array([[52.0, 2491.0, 474.0, 1098.0, 468.0]])

#predicting the apartment using linear regression
linear_prediction_median_compex_value = regressor.predict(linear_prediction)

print('Predicted medianCompexValue using linear regression: ', linear_prediction
#this will print 266947.60
```

By passing some valid data to the trained model, we will get the prediction as result.

Task 4 -- Re-train your model. Use Ridge, Lasso or Elastic Net regularization.

```
ridge_regressor = Ridge(alpha=10)
ridge_regressor.fit(X_train, y_train)
ridge_y_pred = ridge_regressor.predict(X_test)

ridge_prediction = np.array([[52.0, 2491.0, 474.0, 1098.0, 468.0]])
ridge_prediction_median_compex_value = ridge_regressor.predict(ridge_prediction)

print('Predicted medianCompexValue using Ridge: ', ridge_prediction_median_compe
#this will print 266947.46
```

Retraining the model using Ridge and predicting using the same values passed to the linear regresion model for prediction

Task 5 -- Score and compare the scores of the models you have implemented. Interpret the result.

```
regressor_y_pred = regressor.predict(X_test)
linear_r2 = r2_score(y_test, regressor_y_pred)
ridge_r2 = r2_score(y_test, ridge_y_pred)
print("R2 score for ridge: ",  ridge_r2)  #0.175653768
print("R2 score for linear: ", linear_r2) #0.175653829
```

In order to compare these 2 models accuracy, I used r2 score for both of them. R2 score basically means the the credibility level and it can have values between 0 and 1(0-100%) -- the bigger the value, the better, this score should not be less than zero as that will not mean anything good.

Analyzing the result, we can see that the linear regression performed a little bit better, but the difference is hardly making any difference. One possible way to make a bigger difference of this score, is to use a larger alpha value on ridge model to increase the the regularization strength, however, chosing a variable that is too large may result in overfitting, which is not good as well.

Conclusions:

Concluding, I can say that even if linear regression performed a bit better, it's not a big difference between the 2. As mentioned in task 5, a solution would be to increase alpha value to increase regularization strength, but it might overtif the model.

Bibliography:

https://datatofish.com/convert-text-file-to-csv-using-python-tool-included/ -- Convert from text to csv

https://realpython.com/linear-regression-in-python/ -- How to implement linear regresion model

https://machinelearningmastery.com/ridge-regression-with-python/ -- How to implement ridge regression model

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html -- sklearn lib documentation