



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

4/3/2022

Αναγνώριση Προτύπων: Απαλλακτική Εργασία 2021-2022

ΓΕΩΡΓΙΟΣ ΜΠΑΛΤΖΑΚΗΣ

ΑΜ: Π18105

Email: george-baltzakis@hotmail.gr

Εισαγωγή

Η παρούσα εργασία έχει ως στόχο την ανάπτυξη αλγορίθμων μηχανικής μάθησης για την προσέγγιση της διάμεσης τιμής ενός ακινήτου σε μια ευρύτερη γεωγραφική περιοχή της πολιτείας της Καλιφόρνια βάσει ενός συνόλου αντικειμενικών χαρακτηριστικών των ακινήτων στην εν λόγω περιοχή. Κάθε τέτοια περιοχή αποτελεί στην πραγματικότητα την μικρότερη γεωγραφική οντότητα που καταγράφηκε στην σχετική απογραφή του 1990 με πληθυσμιακό εύρος μεταξύ των 600 και 3000 κατοίκων. Το σχετικό σύνολο των δεδομένων είναι αποθηκευμένο στο αρχείο **“housing.csv”**.

Η διαδικασία εκπαίδευσης των εμπλεκόμενων μηχανισμών μηχανικής μάθησης θα πρέπει να βασιστεί σε ένα σύνολο αντικειμενικών γνωρισμάτων των ακινήτων που υπάρχουν σε κάθε γεωγραφική περιοχή το οποίο περιλαμβάνει τα παρακάτω χαρακτηριστικά:

- i. το γεωγραφικό μήκος (longitude) του κέντρου της περιοχής.
 - ii. το γεωγραφικό πλάτος (latitude) του κέντρου της περιοχής.
 - iii. την διάμεση ηλικία των ακινήτων (housing_median_age) της περιοχής.
 - iv. το συνολικό πλήθος δωματίων (total_rooms) των ακινήτων της περιοχής.
 - v. το συνολικό πλήθος υπνοδωματίων (total_bedrooms) των ακινήτων της περιοχής.
 - vi. τον πληθυσμό (population) της περιοχής
 - vii. το πλήθος των νοικοκυριών (households) της περιοχής
 - viii. το διάμεσο εισόδημα (median_income) των κατοίκων της περιοχής
 - ix. την εγγύτητα προς τον ωκεανό (ocean_proximity) της περιοχής
- ώστε να προσεγγιστεί:
- x. η διάμεση τιμή (median_house_value) των ακινήτων της περιοχής.

Η υλοποίηση των μηχανισμών μάθησης, η προ επεξεργασία και οπτικοποίηση των δεδομένων, πραγματοποιήθηκε με τη βοήθεια της γλώσσας Προγραμματισμού **Python** και μερικών βιβλιοθηκών της. Πιο συγκεκριμένα, χρησιμοποιήθηκαν:

- Python 3.7.11
- Jupyter 1.0.0
- Pandas 1.3.5
- NumPy 1.21.2
- TensorFlow 2.0.0
- Scikit-Learn 1.0.2
- Matplotlib 3.5.1

1. Προ Επεξεργασία Δεδομένων:

a. Εισαγωγή Δεδομένων (Έλεγχος για διπλότυπες εγγραφές)

```
1 import numpy as np
2 import pandas as pd
3 from sklearn import preprocessing
4 import matplotlib.pyplot as plt
5 from sklearn.model_selection import KFold
6 from sklearn.metrics import mean_absolute_error
7 from tensorflow.keras.models import Sequential
8 from tensorflow.keras.layers import Dense
9 from tensorflow.keras.optimizers import Adam
10 from tensorflow.keras.callbacks import EarlyStopping

1 # Import Data
2 original_data = pd.read_csv('housing.csv')
3 data = pd.read_csv('housing.csv')
4
5 # Drop Duplicates
6 data.drop_duplicates()
7
8 # A First View of Data
9 print("\n\tData Info\n")
10 data.info()
11 print("\n\n\tData Head\n")
12 data.head()
```

Data Info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28640 entries, 0 to 28639
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   longitude              28640 non-null float64
1   latitude               28640 non-null float64
2   housing_median_age     28640 non-null float64
3   total_rooms            28640 non-null float64
4   total_bedrooms        28433 non-null float64
5   population             28640 non-null float64
6   households             28640 non-null float64
7   median_income          28640 non-null float64
8   median_house_value     28640 non-null float64
9   ocean_proximity        28640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Data Head

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

b. Έλεγχος για εγγραφές που περιέχουν κενά πεδία:

1. Αντικατάσταση των πεδίων αυτών με τη διάμεση τιμή του συγκεκριμένου χαρακτηριστικού.

```
1 # 4) Search for null values
2 for col in data.columns:
3
4     # Get indecies of "data DataFrame" rows, if data['col'] has null values
5     null_values_index = data[col].index(pd.isnull(data[col]))
6
7     # If null_rows_index isn't empty
8     if len(null_values_index) > 0:
9
10        # Get col's median
11        median = data[col].median()
12
13        # Replace the NaN values with median value of this column
14        for data_values_index in null_values_index:
15            data.at[data_values_index, col] = median
16
```

- c. Εφαρμογή *Min Max* Αλγορίθμου Κλιμάκωσης (scaling) στο διάστημα $[0,1]$, για τα αριθμητικά Δεδομένα.

```
1 # 2) MinMax Scaling to [0, 1]
2 data_columns_for_scaling = ['housing_median_age', 'total_rooms', 'total_bedrooms',
3                               'population', 'households', 'median_income', 'median_house_value']
4 data[data_columns_for_scaling] = preprocessing.MinMaxScaler().fit_transform(data[data_columns_for_scaling])
```

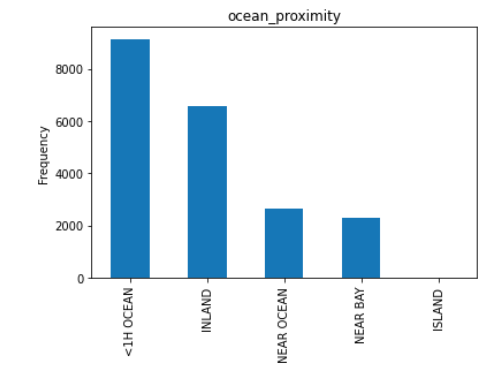
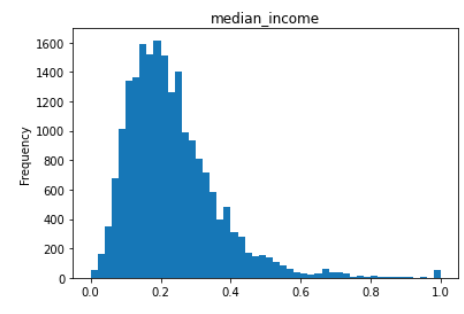
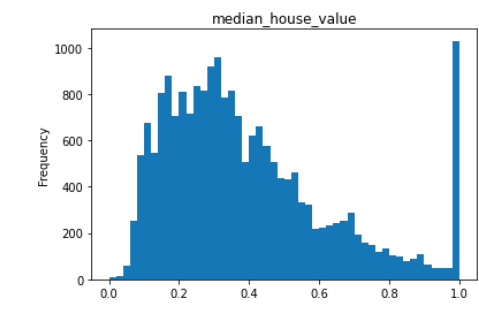
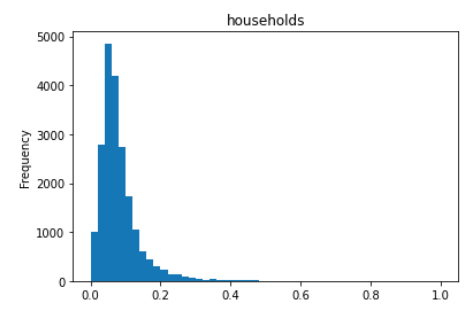
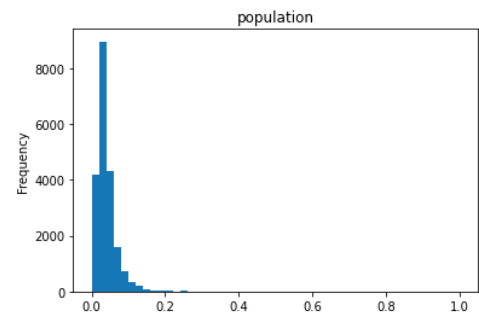
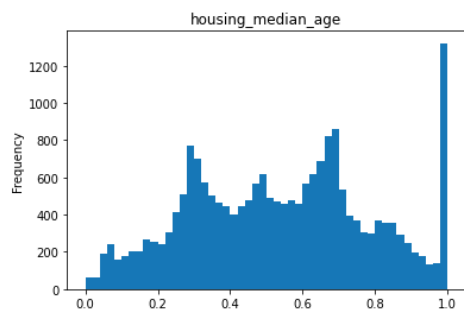
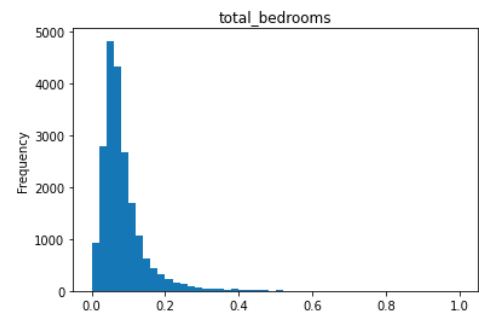
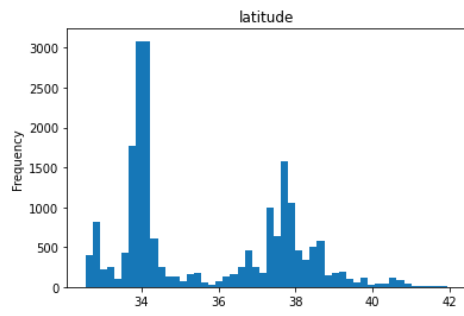
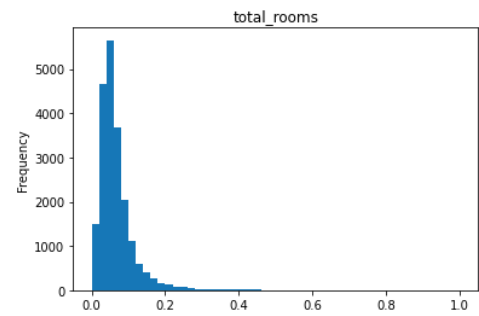
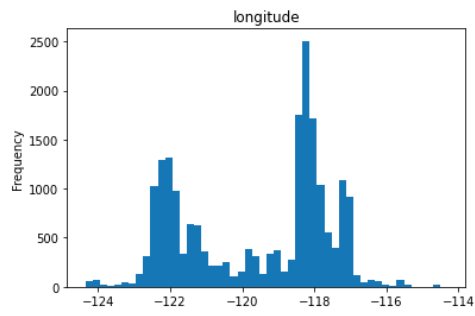
- d. Εφαρμογή *One Hot Vector Encoding* για το κατηγορικό χαρακτηριστικό.

```
1 # 3) One Hot Vector Encoding
2 data.groupby('ocean_proximity')['ocean_proximity'].count()
3
4 # We save this Series for Visualization
5 ocean_proximity = data.ocean_proximity
6
7 # Create a One Hot Vector Encoding DataFrame
8 one_hot_vector_encoding = pd.get_dummies(data.ocean_proximity, prefix = 'Ocean_Proximity')
9
10 # Join that to the data DataFrame
11 data = data.join(one_hot_vector_encoding)
12
13 # Delete the column: ocean_proximity
14 data.drop(['ocean_proximity'], axis = 1, inplace = True)
```

2. Οπτικοποίηση Δεδομένων:

- a. Προβολή Ιστογράμματος για κάθε χαρακτηριστικό.

```
1 ##
2 # -- Data Visualization --
3 ##
4
5 # 1) Frequency Histograms of each column
6 #
7 # For all numeric columns
8 for i in data.columns[:-5]:
9     plt.figure()
10    plt.hist(data[i], bins=50)
11    plt.gca().set(title=i, ylabel='Frequency')
12
13 # For Categorical feature: ocean_proximity
14 plt.figure()
15 plt.title('ocean_proximity')
16 plt.ylabel('Frequency')
17 ocean_proximity.value_counts().plot(kind='bar')
18 plt.show()
```



β. Προβολή Πολλαπλών Ιστογραμμάτων.

```
1 # 2) Histogram Overlapping
```

```
2
```

```
3 # First between total_rooms and total_bedrooms columns
```

```
4 plt.figure(figsize=(13, 4))
```

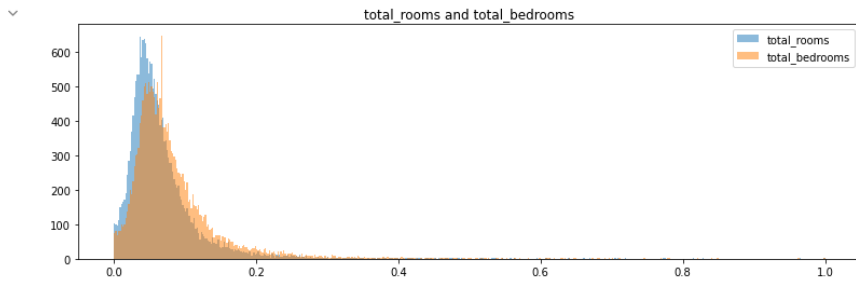
```
5 plt.title('total_rooms and total_bedrooms')
```

```
6 plt.hist(data.total_rooms, bins = 500, alpha = 0.5, label = 'total_rooms')
```

```
7 plt.hist(data.total_bedrooms, bins = 500, alpha = 0.5, label = 'total_bedrooms')
```

```
8 plt.legend(loc='upper right')
```

```
9 plt.show()
```



```
1 # Second between total_rooms, total_bedrooms and median_house_value
```

```
2 plt.figure(figsize=(13, 4))
```

```
3 plt.title('total_rooms, total_bedrooms and median_house_value')
```

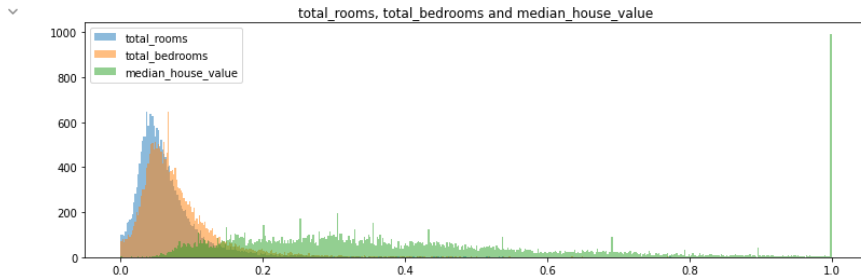
```
4 plt.hist(data.total_rooms, bins = 500, alpha = 0.5, label = 'total_rooms')
```

```
5 plt.hist(data.total_bedrooms, bins = 500, alpha = 0.5, label = 'total_bedrooms')
```

```
6 plt.hist(data.median_house_value, bins = 500, alpha = 0.5, label = 'median_house_value')
```

```
7 plt.legend(loc='upper left')
```

```
8 plt.show()
```



```
1 # And Last between total_rooms, total_bedrooms, median_house_value and median_income
```

```
2 plt.figure(figsize=(13, 4))
```

```
3 plt.title('total_rooms, total_bedrooms, median_house_value and median_income')
```

```
4 plt.hist(data.total_rooms, bins = 500, alpha = 0.5, label = 'total_rooms')
```

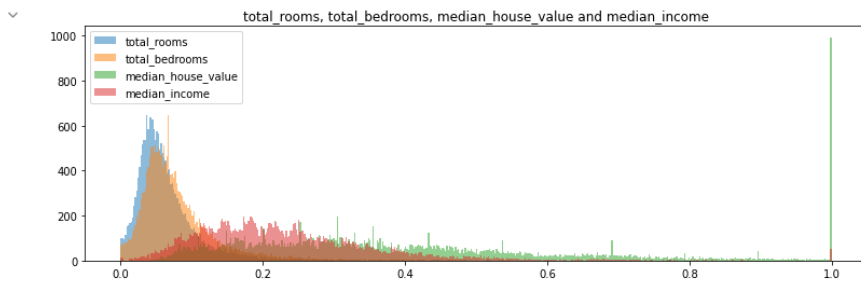
```
5 plt.hist(data.total_bedrooms, bins = 500, alpha = 0.5, label = 'total_bedrooms')
```

```
6 plt.hist(data.median_house_value, bins = 500, alpha = 0.5, label = 'median_house_value')
```

```
7 plt.hist(data.median_income, bins = 500, alpha = 0.5, label = 'median_income')
```

```
8 plt.legend(loc='upper left')
```

```
9 plt.show()
```



3. Παλινδρόμηση Δεδομένων:

- a. Προετοιμασία και διαχωρισμός δεδομένων (για *Least Mean Squares* και *Least Squares*)
 - i. *K-Fold Cross Validation*

```
1 # Dataset Preparation
2 independent_columns = ['longitude', 'latitude', 'housing_median_age', 'total_rooms',
3                        'total_bedrooms', 'population', 'households', 'median_income',
4                        'Ocean_Proximity_<1H OCEAN', 'Ocean_Proximity_INLAND', 'Ocean_Proximity_ISLAND',
5                        'Ocean_Proximity_NEAR BAY', 'Ocean_Proximity_NEAR OCEAN']
6 dependent_column = 'median_house_value'
7
8 X = data[independent_columns].to_numpy(dtype=float)
9 Y = data[dependent_column].to_numpy(dtype=float)
10
11 # 10-Fold Cross-Validation
12 train_index, test_index = my_KFold(X, 10)
13
14 # Define the training and testing sets
15 X_train, X_test = X[train_index], X[test_index]
16 Y_train, Y_test = Y[train_index], Y[test_index]
```

```
1 # k-Fold Cross-Validation
2 def my_KFold(vx, k):
3
4     kf = KFold(n_splits = k)
5     vtrain_index, vtest_index = next(kf.split(vx))
6
7     return vtrain_index, vtest_index
```

- b. *Least Mean Squares* Αλγόριθμος Γραμμικής Παλινδρόμησης:
 - i. Εκπαίδευση (Train) με *Gradient Descent* και συνάρτηση απώλειας *Mean Squared Error*.

```
1 # Cost of Mean Squared Error (MSE)
2 def mean_squared_error(y_true, y_predicted):
3
4     # Calculating the loss or cost
5     cost = np.sum((y_true-y_predicted)**2) / len(y_true)
6     return cost
```

```

1 # Gradient descent with MSE minimization
2 def gradient_descent(x, y, epochs = 1000, learning_rate = 0.00006,
3     stopping_threshold = 1e-6):
4
5     # Initializing weight, bias, learning rate and iterations
6     current_weights = np.ones(len(x[0]))
7     current_bias = 0.01
8     epochs = epochs
9     learning_rate = learning_rate
10    n = float(len(x))
11
12    costs = []
13    previous_cost = None
14
15    # Estimation of optimal parameters
16    for i in range(epochs):
17
18        # Making predictions
19        y_predicted = np.dot(x, current_weights) + current_bias
20
21        # Calculation the current cost
22        current_cost = mean_squared_error(y, y_predicted)
23
24        # If the change in cost is less than or equal to
25        # stopping_threshold we stop the gradient descent
26        if previous_cost and abs(previous_cost-current_cost)<=stopping_threshold:
27            break
28
29        previous_cost = current_cost
30
31        costs.append(current_cost)
32
33        # Calculating the gradients
34        weight_derivative = -(2/n) * np.dot(x.T,(y-y_predicted))
35        bias_derivative = -(2/n) * sum(y-y_predicted)
36
37        # Updating weights and bias
38        current_weights = current_weights - (learning_rate * weight_derivative)
39        current_bias = current_bias - (learning_rate * bias_derivative)
40
41        # Printing the parameters for each Epoch
42        print('\n-----\n')
43        print('Epoch:', i+1, '\n\nMSE Cost:', current_cost, '\nMAE Cost:', mean_absolute_error(y,
44            y_predicted), '\nWeights:\n', current_weights, '\nBias:', current_bias)
45    return current_weights, current_bias, costs

```

```

1 #
2 # 1) LMS Training
3 #
4 # Estimating weights and bias using Gradient Descent with MSE minimization
5 ep = 500
6 estimated_weights, estimated_bias, costs = gradient_descent(X_train, Y_train, epochs=ep)
7
8 print('\n-----\n')
9 print('Final Estimated Weights: ', estimated_weights, '\n\nFinal Estimated Bias: ', estimated_bias)

```

Epoch: 1

MSE Cost: 6773.161026707132

MAE Cost: 82.29455364968523

Weights:

[-0.17827258 1.34917855 1.00529871 1.00066246 1.00082664 1.00039892
1.00081518 1.00230198 1.00481644 1.00310121 1.0000027 1.00053714
1.00141786]

Bias: 0.019875346437962238

Epoch: 2

MSE Cost: 5002.0922248690185

MAE Cost: 70.65578160611246

Weights:

[0.83403302 1.0485425 1.00074646 1.00009303 1.00011676 1.00005675
1.00011538 1.00033319 1.00077154 1.00034256 1.00000052 1.00005457
1.00022746]

Bias: 0.011396652645228773

Epoch: 3

MSE Cost: 3694.364849691922

MAE Cost: 60.77321327020683

Weights:

[-0.0360078 1.30628716 1.00465906 1.00058215 1.00072719 1.00035139
1.00071739 1.00203424 1.00434109 1.00261927 1.00000253 1.00044823
1.00127832]

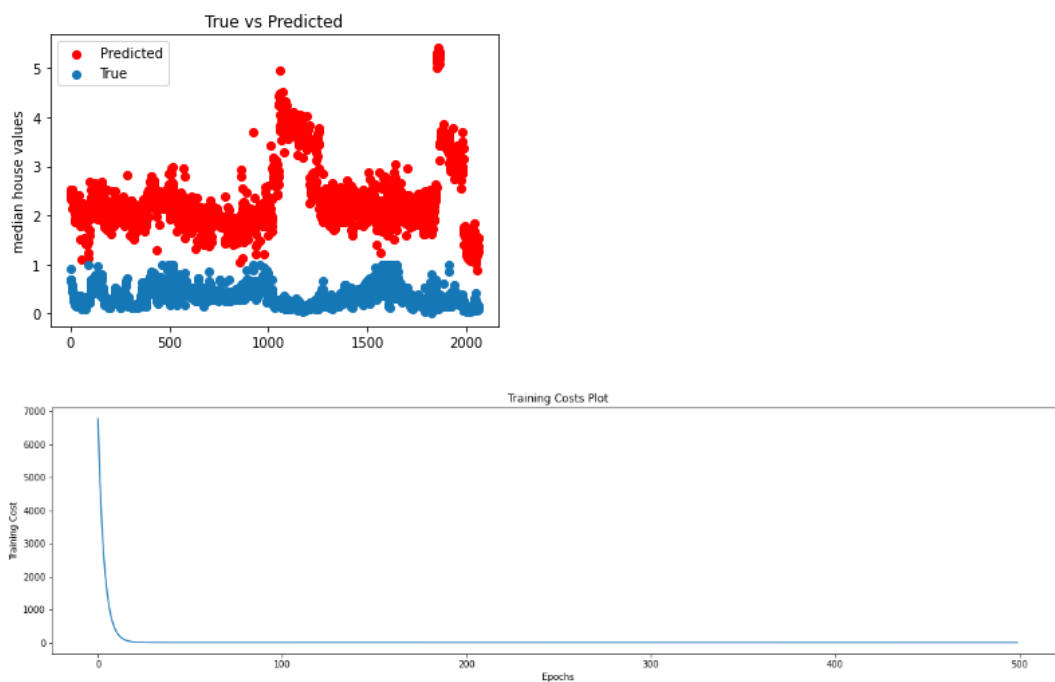
Bias: 0.018689438237653587

ii. Δοκιμή (Test), και προβολή αποτελεσμάτων

```
1 #  
2 # LMS Testing  
3 #  
4 Y_pred = X_test.dot(estimated_weights.T) + estimated_bias  
5  
6 #  
7 print('\n', 'LMS Testing MSE (with', ep, 'training Epochs):', mean_squared_error(Y_test, Y_pred))  
8 print('\n', 'LMS Testing MAE (with', ep, 'training Epochs):', mean_absolute_error(Y_test, Y_pred))  
9  
10 # True vs Predicted Plot  
11 plt.figure()  
12 plt.scatter(list(range(len(Y_pred))), pd.DataFrame(Y_pred), label='Predicted', color='red')  
13 plt.scatter(list(range(len(Y_test))), pd.DataFrame(Y_test), label='True')  
14 plt.ylabel('median house values')  
15 plt.title('True vs Predicted')  
16 plt.legend(loc='best')  
17 plt.show()  
18  
19 # Cost plot  
20 costs = pd.DataFrame(costs)  
21 plt.figure(figsize=(20, 5))  
22 plt.plot(costs)  
23 plt.autoscale()  
24 plt.title('Training Costs Plot')  
25 plt.ylabel('Training Cost')  
26 plt.xlabel('Epochs')  
27 plt.show()
```

LMS Testing MSE (with 500 training Epochs): 4.420479071378372

LMS Testing MAE (with 500 training Epochs): 1.9724113151838392



- c. *Least Squares* Αλγόριθμος Γραμμικής Παλινδρόμησης:
 i. Υλοποίηση Αλγορίθμου

```

1 #
2 # 2) LS Building the model
3 #
4 m = []
5 X_mean = []
6 Y_mean = np.mean(Y_train)
7
8 for i in range(len(X_train[0])):
9     X_mean.append(np.mean(X_train[i]))
10    num = 0
11    den = 0
12    for j in range(len(X_train)):
13        num += (X_train[j, i] - X_mean[i])*(Y_train[j] - Y_mean)
14        den += (X_train[j, i] - X_mean[i])**2
15    m.append(num / den)
16
17 c = Y_mean - np.dot(np.array(m), np.array(X_mean))

```

- ii. Δοκιμή (Test), και προβολή αποτελεσμάτων

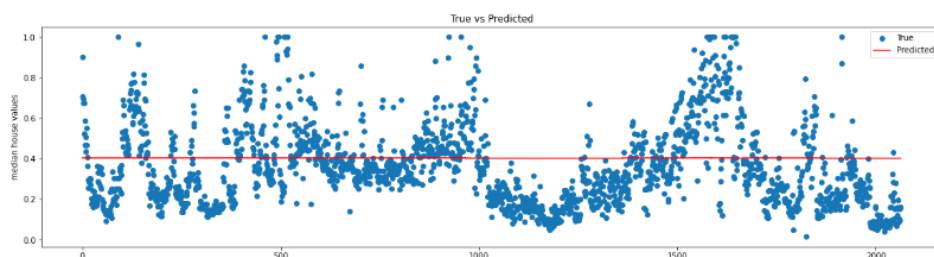
```

1 #
2 # LS Testing
3 #
4 LS_predicted = np.dot(np.array(m), X_test.T) + c
5
6 print('\n', 'LS Testing MSE:', mean_squared_error(Y_test, LS_predicted))
7 print('\n', 'LS Testing MAE:', mean_absolute_error(Y_test, LS_predicted))
8
9 # True vs Predicted Plot
10 plt.figure(figsize=(20, 5))
11 plt.scatter(list(range(len(LS_predicted))), pd.DataFrame(Y_test), label='True')
12 plt.plot(pd.DataFrame(LS_predicted), label='Predicted', color='red')
13 plt.ylabel('median house values')
14 plt.title('True vs Predicted')
15 plt.legend(loc='best')
16 plt.show()

```

LS Testing MSE: 0.042155660171509485

LS Testing MAE: 0.16764192465561106



- d. Μη-Γραμμική Παλινδρόμηση με πολυστρωματικό Νευρωνικό Δίκτυο:
- i. Προετοιμασία Δεδομένων

```
1 #  
2 # Non-Linear Regression with Neural Networks  
3 #  
4  
5 # Preparation  
6  
7 # Split to train, validation, test sets  
8 X_train_validation = X_train.copy()  
9 Y_train_validation = Y_train.copy()  
10  
11 train_index, validation_index = my_KFold(X_train_validation, 10)  
12  
13 X_train = X_train_validation[train_index]  
14 Y_train = Y_train_validation[train_index]  
15  
16 X_validation = X_train_validation[validation_index]  
17 Y_validation = Y_train_validation[validation_index]
```

- ii. Δημιουργία και Εκπαίδευση (Train) Μοντέλου

```
1 # Create the Model  
2 model = Sequential()  
3 model.add(Dense(128, activation="relu", input_dim=13))  
4 model.add(Dense(32, activation="relu"))  
5 model.add(Dense(8, activation="relu"))  
6 # Since the regression is performed, a Dense layer containing a single neuron with a linear  
   activation function.  
7 # Typically, ReLu-based activation are used but since it is performed regression, it is needed a  
   linear activation.  
8 model.add(Dense(1, activation="linear"))  
9  
10 # Compile model: The model is initialized with the Adam optimizer and then it is compiled.  
11 model.compile(loss='mean_absolute_error', optimizer=Adam(lr=1e-3, decay=1e-3 / 200))  
12  
13 # Patient early stopping  
14 es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=200)  
15  
16 # Fit the model  
17 history = model.fit(X_train, Y_train, validation_data=(X_validation, Y_validation), epochs=50,  
   batch_size=100, verbose=2, callbacks=[es])
```

iii. Δοκιμή (Test), προβολή αποτελεσμάτων

```
1 #  
2 # NN Testing  
3 #  
4  
5 # Calculate predictions  
6 NN_predicted = model.predict(X_test)  
7  
8 print('\n', 'NN Testing MSE:', mean_squared_error(Y_test, NN_predicted))  
9 print('\n', 'NN Testing MAE:', mean_absolute_error(Y_test, NN_predicted))  
10  
11 # True vs Predicted Plot  
12 plt.figure(figsize=(20, 5))  
13 plt.scatter(list(range(len(NN_predicted))), pd.DataFrame(Y_test), label='True')  
14 plt.scatter(list(range(len(NN_predicted))), pd.DataFrame(NN_predicted), label='Predicted',  
15             color='red')  
16 plt.ylabel('median house values')  
17 plt.title('True vs Predicted')  
18 plt.legend(loc='best')  
19 plt.show()
```

NN Testing MSE: 185.6908787764589

NN Testing MAE: 0.14050895666509014

