ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΒΕΛΛΟΣ ΓΙΩΡΓΟΣ Π21011

ΙΑΝΟΥΑΡΙΟΣ 2024

ΠΕΡΙΕΧΟΜΕΝΑ:

Προ-επεξεργασία Δεδομένων

Αρχικά σε πρώτο στάδιο θα φορτώσουμε τα δεδομένα από το csv αρχείο σε μια μεταβλητή. Για να φορτώσουμε τα δεδομένα χρησιμοποιούμε την βιβλιοθήκη pandas.

1) Σε πρώτο στάδιο πρέπει να αναγνωρίσουμε τα υποσύνολα των αριθμητικών και των κατηγορικών χαρακτηριστικών.

Με την μέθοδο .info() της pandas πάνω στα δεδομένα βλέπουμε τις στήλες, τις γραμμές και των τύπο των δεδομένων.

```
/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/patternRec.py"
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
```

Με την μέθοδο .head() της pandas πάνω στα δεδομένα βλέπουμε τις 5 πρώτες γραμμές αυτά.

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos
/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/patternRec.py"
   longitude  latitude  housing_median_age  total_rooms  ...  households  median_income  median_house_value  ocean_proximity
0    -122.23     37.88                41.0        880.0  ...       126.0         8.3252            452600.0         NEAR BAY
1    -122.22     37.86                21.0       7099.0  ...      1138.0         8.3014            358500.0         NEAR BAY
2    -122.24     37.85                52.0       1467.0  ...       177.0         7.2574            352100.0         NEAR BAY
3    -122.25     37.85                52.0       1274.0  ...       219.0         5.6431            341300.0         NEAR BAY
4    -122.25     37.85                52.0       1627.0  ...       259.0         3.8462            342200.0         NEAR BAY

[5 rows x 10 columns]
```

Με την μέθοδο .describe() της pandas πάνω στα δεδομένα παίρνουμε τα στατιστικά κάθε στήλης του πίνακα.

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos
/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/patternRec.py"
          longitude      latitude  housing_median_age   total_rooms  ...    population    households  median_income  median_house_value
count  20640.000000  20640.000000        20640.000000  20640.000000  ...  20640.000000  20640.000000   20640.000000        20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081  ...   1425.476744    499.539680       3.870671       206855.816909
std        2.003532      2.135952           12.585558   2181.615252  ...   1132.462122    382.329753       1.899822       115395.615874
min     -124.350000     32.540000            1.000000      2.000000  ...      3.000000      1.000000       0.499900        14999.000000
25%     -121.800000     33.930000           18.000000   1447.750000  ...    787.000000    280.000000       2.563400       119600.000000
50%     -118.490000     34.260000           29.000000   2127.000000  ...   1166.000000    409.000000       3.534800       179700.000000
75%     -118.010000     37.710000           37.000000   3148.000000  ...   1725.000000    605.000000       4.743250       264725.000000
max     -114.310000     41.950000           52.000000  39320.000000  ...  35682.000000   6082.000000      15.000100       500001.000000

[8 rows x 9 columns]
```

Στην στήλη ocean_proximity παρατηρούμε ότι έχει περιγραφές με λόγια στην απόσταση από την θάλασσα. Με την εντολή .value_counts() βλέπουμε όλες τις τιμές.

```
(base) george_vellos@Georges-MacBook
/INFORMATICS_UNIPI/5 SEMESTER/Patter
/Users/george_vellos/opt/anaconda3/b
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

ΚΩΔΙΚΑΣ:

```
cal_housing = pd.read_csv('housing.csv')
# 1 Explore Data
print(cal_housing.info())
print(cal_housing.head())
print(cal_housing.describe())
print(cal_housing["ocean_proximity"].value_counts())
```

2) Στο δεύτερο υποερώτημα θα χρησιμοποιήσουμε διάφορες τεχνικές κλιμάκωσης (scaling) έτσι ώστε να έχουμε καλύτερη απόδοση σε αλγορίθμους μηχανικής μάθησης αλλά και γενικότερα σε τεχνικές ανάλυσης δεδομένων όπως τώρα.
Σε αυτό το ερώτημα κάνουμε χρήση της βιβλιοθήκης pandas, της sklearn και από αυτήν το πακέτο preprocessing.

Αρχικά φορτώνουμε τα δεδομένα και αφαιρούμε την στήλη ocean_proximity μιας και δεν μπορεί να εφαρμοστεί scaling πάνω σε αυτήν την μορφή που είναι τα δεδομένα. Θα μπορούσαμε βέβαια να δώσουμε αριθμητική τιμή για τις πέντε τιμές απόστασης αλλά δεν νομίζω ότι θα είχε κάποιο ιδιαίτερο νόημα.

Η πρώτη τεχνική που θα χρησιμοποιήσουμε είναι: Min-Max Scaling

$$X_{\text{scaled}} = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}$$

Αρχικά, ψάχνουμε την ελάχιστη και την μέγιστη τιμή κάθε στήλης. Στη συνέχεια θα αφαιρέσουμε την ελάχιστη τιμή από την εγγραφή και διαιρούμε το αποτέλεσμα με την διαφορά ελάχιστης και μέγιστης τιμής.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το module preprocessing και από αυτό θα χρησιμοποιήσουμε την κλάση MinMaxScaler(). Στη συνέχεια αποθηκεύομε σε μια μεταβλητή τα δεδομένα που έχουν υποστεί κλιμάκωση και έχουν ‘εκπαιδευτεί’ με την χρήση της μεθόδου fit_transform(). Το επόμενο που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας σε μορφή πίνακα με την βιβλιοθήκη Pandas πιο συγκεκριμένα με την κλάση DataFrame(). Τέλος, εκτυπώνουμε το αποτέλεσμα(εμφανίζω τα 5 πρώτα μόνο).

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Use
rs/george_vellos/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/scaling.py"
   longitude  latitude  housing_median_age  total_rooms  ...  population  households  median_income  median_house_value
0   0.211155  0.567481            0.784314     0.022331  ...    0.008941    0.020556       0.539668            0.902266
1   0.212151  0.565356            0.392157     0.180503  ...    0.067210    0.186976       0.538027            0.708247
2   0.210159  0.564293            1.000000     0.037260  ...    0.013818    0.028943       0.466028            0.695051
3   0.209163  0.564293            1.000000     0.032352  ...    0.015555    0.035849       0.354699            0.672783
4   0.209163  0.564293            1.000000     0.041330  ...    0.015752    0.042427       0.230776            0.674638

[5 rows x 9 columns]
```

Κώδικας:

```
min_max_test = preprocessing.MinMaxScaler()
X_train_minmax_test = min_max_test.fit_transform(cal_housing_numbers)
cal_housing_minmax_scaled_df = pd.DataFrame(X_train_minmax_test, columns=number_columns)
print(cal_housing_minmax_scaled_df.head())
```

Η δεύτερη τεχνική που θα χρησιμοποιήσουμε είναι: Standardization

$$X_{\text{scaled}} = \frac{X_i - X_{\text{mean}}}{\sigma}$$

Αρχικά, υπολογίζουμε το mean και το standar deviation των δεδομένων που θέλουμε να επικανονικοποιήσουμε. Στη συνέχεια αφαιρούμε την μέση τιμή από κάθε εγγραφή και διαιρούμε με το αποτέλεσμα της τυπικής απόκλισης.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το module preprocessing και από αυτό θα χρησιμοποιήσουμε την κλάση StandardScaler(). Στη συνέχεια αποθηκεύομε σε μια μεταβλητή τα δεδομένα που έχουν υποστεί κλιμάκωση και έχουν 'εκπαιδευτεί' με την χρήση της μεθόδου fit_transform(). Το επόμενο που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας σε μορφή πίνακα με την βιβλιοθήκη Pandas πιο συγκεκριμένα με την κλάση DataFrame(). Τέλος, εκτυπώνουμε το αποτέλεσμα(εμφανίζω τα 5 πρώτα μόνο).

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos/Desktop
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/scaling.py"
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value
0  -1.327835  1.052548            0.982143    -0.804819       -0.970325   -0.974429   -0.977033       2.344766            2.129631
1  -1.322844  1.043185           -0.607019     2.045890        1.348276    0.861439    1.669961       2.332238            1.314156
2  -1.332827  1.038503            1.856182    -0.535746       -0.825561   -0.820777   -0.843637       1.782699            1.258693
3  -1.337818  1.038503            1.856182    -0.624215       -0.718768   -0.766028   -0.733781       0.932968            1.165100
4  -1.337818  1.038503            1.856182    -0.462404       -0.611974   -0.759847   -0.629157      -0.012881            1.172900
```

Κώδικας:

```
standar_test = preprocessing.StandardScaler()
X_train_standar_test = standar_test.fit_transform(cal_housing_numbers)
cal_housing_standard_scaled_df = pd.DataFrame(X_train_standar_test, columns=number_columns)
print(cal_housing_standard_scaled_df.head())
```

Η τρίτη τεχνική που θα χρησιμοποιήσουμε είναι: Min-Max Scaling

$$X_{\text{scaled}} = \frac{X_i - \max(|X|)}{\max(|X|)}$$

Αρχικά, πρέπει να επιλέξουμε την μέγιστη απόλυτη τιμή από τις εγγραφές μια συγκεκριμένης μέτρησης. Στη συνέχεια διαιρούμε κάθε εγγραφή της στήλης με την μέγιστη τιμή.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το module preprocessing και από αυτό θα χρησιμοποιήσουμε την κλάση MaxAbsScaler (). Στη συνέχεια αποθηκεύομε σε μια μεταβλητή τα δεδομένα που έχουν υποστεί κλιμάκωση και έχουν ‘εκπαιδευτεί’ με την χρήση της μεθόδου fit_transform(). Το επόμενο που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας σε μορφή πίνακα με την βιβλιοθήκη Pandas πιο συγκεκριμένα με την κλάση DataFrame(). Τέλος, εκτυπώνουμε το αποτέλεσμα(εμφανίζω τα 5 πρώτα μόνο).

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos/Des
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/scaling.py"
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value
0  -0.982951  0.902980            0.788462     0.022380        0.020016    0.009024    0.020717       0.555010            0.905198
1  -0.982871  0.902503            0.403846     0.180544        0.171606    0.067289    0.187110       0.553423            0.716999
2  -0.983032  0.902265            1.000000     0.037309        0.029480    0.013901    0.029102       0.483823            0.704199
3  -0.983112  0.902265            1.000000     0.032401        0.036462    0.015638    0.036008       0.376204            0.682599
4  -0.983112  0.902265            1.000000     0.041378        0.043445    0.015834    0.042585       0.256412            0.684399
```

Κώδικας:

```
maxabs_test = preprocessing.MaxAbsScaler()
X_train_maxabs_test = maxabs_test.fit_transform(cal_housing_numbers)
cal_housing_maxabs_scaled_df = pd.DataFrame(X_train_maxabs_test, columns=number_columns)
print(cal_housing_maxabs_scaled_df.head())
```

Η τέταρτη τεχνική που θα χρησιμοποιήσουμε είναι: Robust Scaling

$$X_{\text{scaled}} = \frac{X_i - X_{\text{median}}}{IQR}$$

Αρχικά, σε αυτήν την μέθοδο scaling θα χρειαστούμε δυο στατιστικές μετρήσεις των δεδομένων: την Median και την Inter-Quartile-Range. Αφού υπολογίσουμε αυτές τις δυο τιμές αφαιρούμε την διάμεση τιμή από κάθε εγγραφή και στη συνέχεια διαιρούμε το αποτέλεσμα με την IQR.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το module preprocessing και από αυτό θα χρησιμοποιήσουμε την κλάση RobustScaler (). Στη συνέχεια αποθηκεύομε σε μια μεταβλητή τα δεδομένα που έχουν υποστεί κλιμάκωση και έχουν 'εκπαιδευτεί' με την χρήση της μεθόδου fit_transform(). Το επόμενο που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας σε μορφή πίνακα με την βιβλιοθήκη Pandas πιο συγκεκριμένα με την κλάση DataFrame(). Τέλος, εκτυπώνουμε το αποτέλεσμα(εμφανίζω τα 5 πρώτα μόνο).

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/scaling.py"
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value
0  -0.986807  0.957672            0.631579    -0.733422       -0.871795   -0.899787   -0.870769       2.197582            1.880448
1  -0.984169  0.952381           -0.421053     2.924276        1.911681    1.316631    2.243077       2.186664            1.232041
2  -0.989446  0.949735            1.210526    -0.388178       -0.698006   -0.714286   -0.713846       1.707732            1.187941
3  -0.992084  0.949735            1.210526    -0.501691       -0.569801   -0.648188   -0.584615       0.967177            1.113523
4  -0.992084  0.949735            1.210526    -0.294074       -0.441595   -0.640725   -0.461538       0.142854            1.119724
```

Κώδικας:

```
robust_test = preprocessing.RobustScaler()
X_train_robust_test = robust_test.fit_transform(cal_housing_numbers)
cal_housing_robust_scaled_df = pd.DataFrame(X_train_robust_test, columns=number_columns)
print(cal_housing_robust_scaled_df.head())
```

Η πέμπτη και τελευταία τεχνική που θα χρησιμοποιήσουμε είναι: Normalization

$$X_{\text{scaled}} = \frac{X_i - X_{\text{mean}}}{X_{\max} - X_{\min}}$$

Αρχικά, αυτή η μέθοδος είναι πολύ παρόμοια με την Min-Max, αλλά αντί να αφαιρούμε την ελάχιστη τιμή αφαιρούμε την μέση τιμή. Στη συνέχεια διαιρούμε το αποτέλεσμα με την διαφορά ελάχιστης και μέγιστης τιμής.
Προκειμένου να μπορέσουμε να εφαρμόσουμε την κανονικοποίηση θα εισάγουμε κάποιες τιμές που λοιπουν από τα δεδομένα. Είναι ουσιαστικά αυτό που ζητάει το τέταρτο υποερώτημα αλλά προκείμενου να εφαρμόσου και αυτήν την μέθοδο θα εισάγω την μέση τιμή σε αυτές που τους λείπει. Στις παραπάνω τεχνικές δεν θεώρησα αναγκαίο να χειριστώ τις τιμές που έλειπαν μιας και θα χρειαστεί να το κάνω σε επόμενο υποερώτημα.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το module preprocessing και από αυτό θα χρησιμοποιήσουμε την κλάση RobustScaler (). Στη συνέχεια αποθηκεύομε σε μια μεταβλητή τα δεδομένα που έχουν υποστεί κλιμάκωση και έχουν ‘εκπαιδευτεί’ με την χρήση της μεθόδου fit_transform(). Το επόμενο που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας σε μορφή πίνακα με την βιβλιοθήκη Pandas πιο συγκεκριμένα με την κλάση DataFrame(). Τέλος, εκτυπώνουμε το αποτέλεσμα(εμφανίζω τα 5 πρώτα μόνο).

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos/
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/scaling.py"
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value
0  -0.000270  0.000084            0.000091     0.001944        0.000285    0.000711    0.000278       0.000018            0.999998
1  -0.000341  0.000106            0.000059     0.019797        0.003084    0.006696    0.003174       0.000023            0.999772
2  -0.000347  0.000107            0.000148     0.004166        0.000540    0.001409    0.000503       0.000021            0.999990
3  -0.000358  0.000111            0.000152     0.003733        0.000689    0.001635    0.000642       0.000017            0.999991
4  -0.000357  0.000111            0.000152     0.004754        0.000818    0.001651    0.000757       0.000011            0.999987
```

Κώδικας:

```
cal_housing = pd.read_csv('housing.csv')
median_value = cal_housing['total_bedrooms'].median()
cal_housing['total_bedrooms'].fillna(median_value, inplace=True)
number_columns = ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population',
'households', 'median_income', 'median_house_value']
cal_housing_numbers = cal_housing[number_columns]
```

.
.
.

```
normal_test = preprocessing.Normalizer()
X_train_normal_test = normal_test.fit_transform(cal_housing_numbers)
cal_housing_minmax_scaled_df = pd.DataFrame(X_train_normal_test, columns=number_columns)
```

```
print(cal_housing_minmax_scaled_df.head())
```

3) Στο τρίτο υποερώτημα θα χρησιμοποιήσουμε την One Hot Vector κωδικοποίηση ώστε τα υποσύνολο των κατηγορικών χαρακτηριστικών δεδομένων να λάβουν διανυσματική αναπαράσταση.

Έχοντας παρατηρήσει τα δεδομένα ξέρουμε ότι η στήλη ocean_proximity περιέχει κατηγορικά δεδομένα (με πέντε διαφορετικές κατηγορίες). Δηλαδή περιγράφει την απόσταση σε μορφή μη αριθμητική. Με την κωδικοποίηση αυτή θα μεταφράσουμε την απόσταση σε διανυσματική αναπαράσταση. (οι βιβλιοθήκες που θα χρησιμοποιήσουμε είναι η pandas και η sklearn)

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /U
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/OneHotVector.py"
['NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
```

Αρχικά θα προβάλουμε τις διαφορετικές ομάδες τιμών της ocean_proximity για να δούμε τι περιέχει. Στη συνέχεια αποθηκεύομε την στήλη στην οποία θα εφαρμόσουμε την κωδικοποίηση. Ορίζουμε ένα αντικείμενο encoder από την κλάση OneHotEncoder και ‘εκπαιδεύουμε’ τα δεδομένα προκείμενου να τα μετασχηματίσουμε σε δυαδική αναπαράσταση. Τα δεδομένα μας τα μετατρέπομε σε πίνακα και έπειτα δημιουργούμε ένα dataframe με τα χαρακτηριστικά αυτά. Τέλος, κάνουμε drop το ocean_proximity και εισάγουμε το καινούργιο dataframe με τις καινούργιές μας τιμές.

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Users/george_vellos/Desktop
/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/OneHotVector.py"
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  ...  (<1H OCEAN,)  (INLAND,)  (ISLAND,)  (NEAR BAY,)  (NEAR OCEAN,)
0        -122.23     37.88                41.0        880.0           129.0  ...           0.0        0.0        0.0          1.0            0.0
1        -122.22     37.86                21.0       7099.0          1106.0  ...           0.0        0.0        0.0          1.0            0.0
2        -122.24     37.85                52.0       1467.0           190.0  ...           0.0        0.0        0.0          1.0            0.0
3        -122.25     37.85                52.0       1274.0           235.0  ...           0.0        0.0        0.0          1.0            0.0
4        -122.25     37.85                52.0       1627.0           280.0  ...           0.0        0.0        0.0          1.0            0.0
...          ...       ...                 ...          ...             ...  ...           ...        ...        ...          ...            ...
20635    -121.09     39.48                25.0       1665.0           374.0  ...           0.0        1.0        0.0          0.0            0.0
20636    -121.21     39.49                18.0        697.0           150.0  ...           0.0        1.0        0.0          0.0            0.0
20637    -121.22     39.43                17.0       2254.0           485.0  ...           0.0        1.0        0.0          0.0            0.0
20638    -121.32     39.43                18.0       1860.0           409.0  ...           0.0        1.0        0.0          0.0            0.0
20639    -121.24     39.37                16.0       2785.0           616.0  ...           0.0        1.0        0.0          0.0            0.0

[20640 rows x 14 columns]
```

Κώδικας:

```
cal_housing = pd.read_csv('housing.csv')
unique_ocean_proximity = cal_housing['ocean_proximity'].unique()
print(unique_ocean_proximity)


ohv_categories = cal_housing[["ocean_proximity"]]
encoder = OneHotEncoder()
housing_cat_ohv = encoder.fit_transform(ohv_categories)
array = housing_cat_ohv.toarray()
encoderDF = pd.DataFrame(array, columns=encoder.categories_)
```

```
dataset = pd.concat([cal_housing, encoderDF], axis=1)
dataset = dataset.drop(['ocean_proximity'], axis=1)
print(dataset)
```

4) Στο τέταρτο υποερώτημα θα αναγνωρίσουμε άμα υπάρχουν αριθμητικά χαρακτηριστικά με ελλιπείς τιμές και στη συνέχεια θα συμπληρώσουμε τις τιμές που απουσιάζουν με την διάμεση τιμή.

Εκτελώντας την εντολή info παρατηρούμε ότι το total_bedrooms έχει 20433 εγγραφές και όχι 20640 όπως οι άλλες στήλες. Λείπουν 207 τιμές τις οποίες θα συμπληρώσουμε με την διάμεση τιμή. Αρχικά θα βρούμε την διάμεση τιμή της total_bedrooms και στη συνέχεια θα την εισάγουμε στις κενές τιμές. Τέλος εκτυπώνοντας τις κενές τιμές παρατηρούμε ότι μας τυπώνει 0.

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition %
PI/5 SEMESTER/Pattern recognition/patternRec.py"
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
Κενές τιμές πριν:
207
Κενές τιμές μετά:
0
```

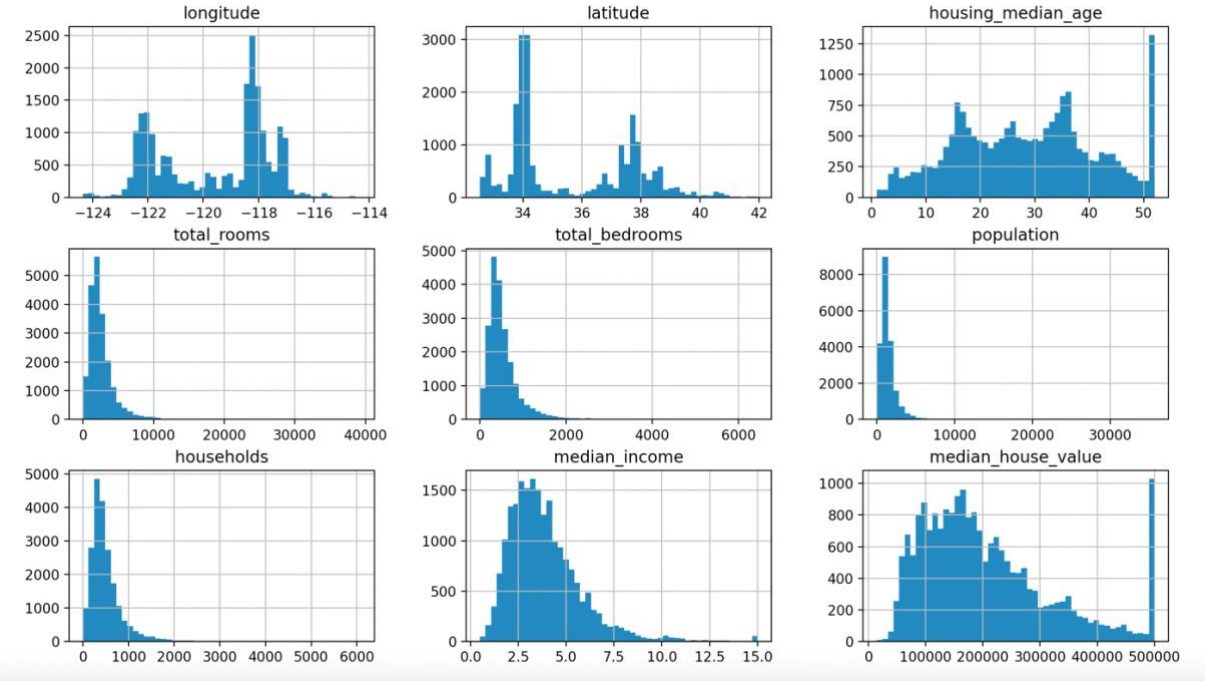
Κώδικας:

```
# 4
print(cal_housing.info())
print("Κενές τιμές πριν: ")
print(cal_housing['total_bedrooms'].isna().sum())
median_value = cal_housing['total_bedrooms'].median()
cal_housing['total_bedrooms'].fillna(median_value, inplace=True)
print("Κενές τιμές μετά: ")
print(cal_housing['total_bedrooms'].isna().sum())
```

Οπτικοποίηση δεδομένων

1) Σε αυτό το υποερώτημα θα αναπαραστήσουμε γραφικά ιστογράμματα συχνοτήτων (που αντιστοιχούν στις συναρτήσεις πυκνότητας πιθανότητας) για κάθε μία από τις 10 μεταβλητές που εμπλέκονται στο πρόβλημα.

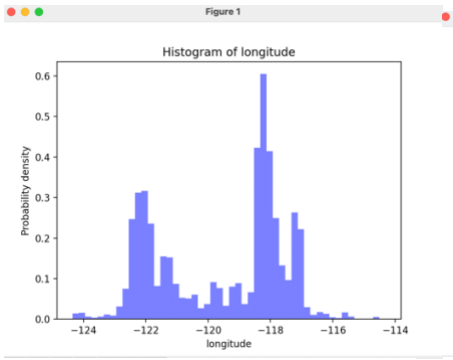
Αρχικά προβάλλοντας απλά το ιστόγραμμα με τις τιμές κάθε στήλης(χωρίς την ocean_proximity) θα πάρουμε:

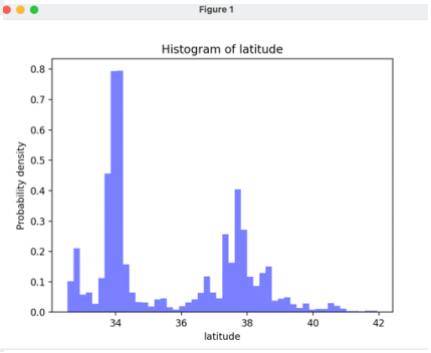


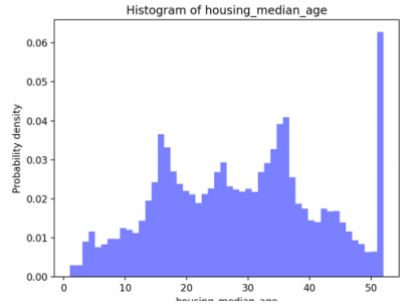
Για να προβάλουμε ιστογράμματα συχνοτήτων που αντιστοιχούν στις συναρτήσεις πυκνότητας πιθανότητας θα χρησιμοποιήσουμε την βιβλιοθήκη matplotlib. Στη συνέχεια θα χρησιμοποιήσουμε την seaborn και θα υπολογίσουμε το Kernel Density Estimation.

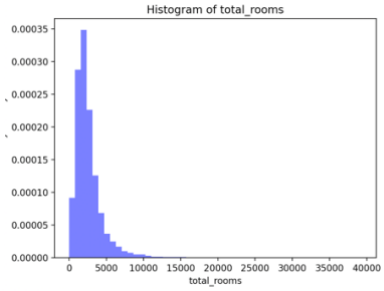
Σε πρώτο στάδιο από την βιβλιοθήκη matplotlib καλούμε την συνάρτηση hist και στις παραμέτρους θέτουμε το density=True. Εκτελώντας το μέσα σε μια επανάληψη παίρνουμε όλα τα ιστογράμματα για τις 9 στήλες του dataset.

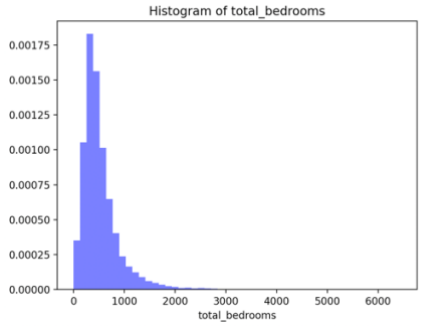
Παράδειγμα εκτέλεσης:

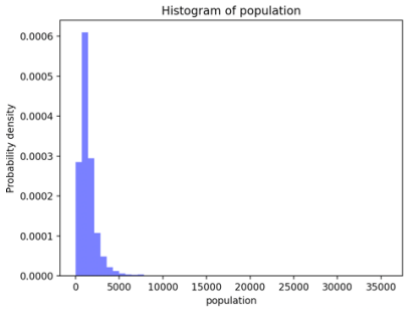


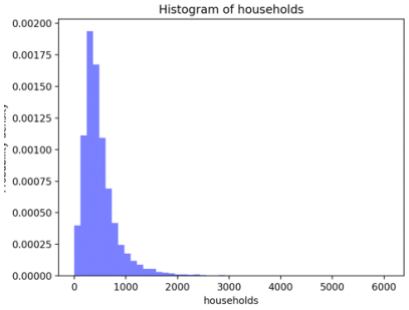


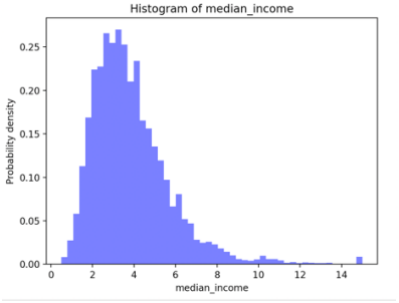


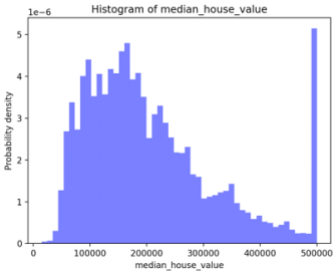










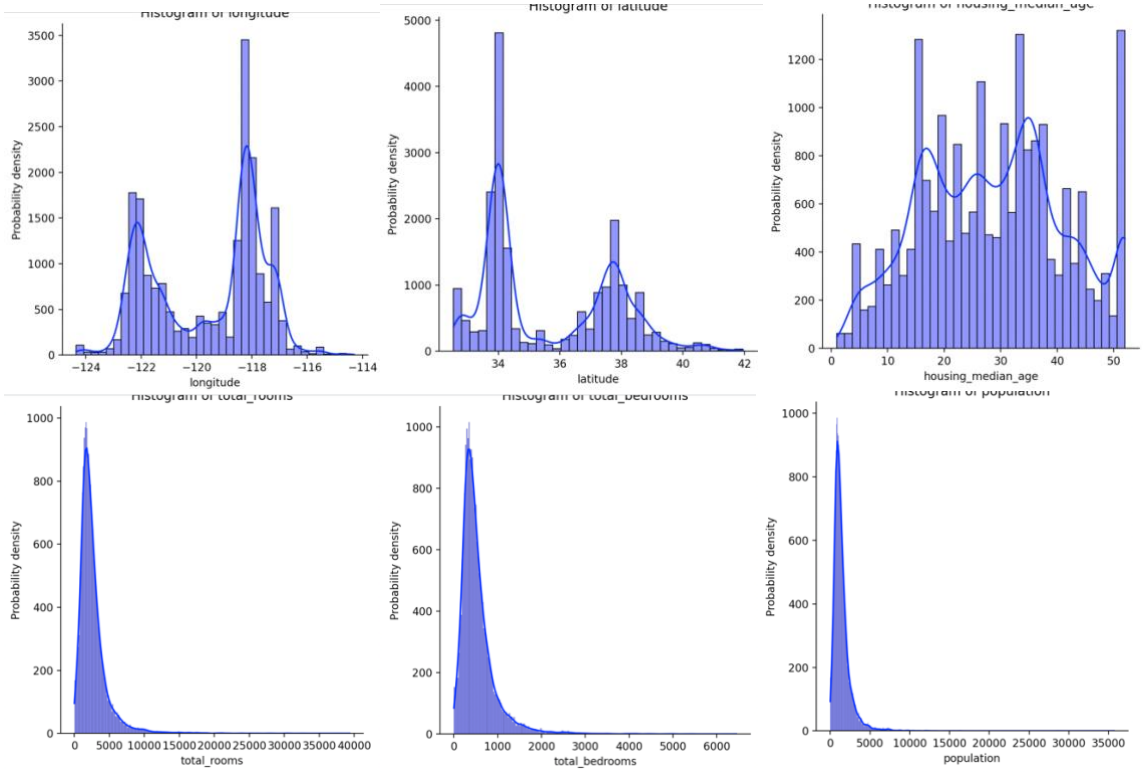


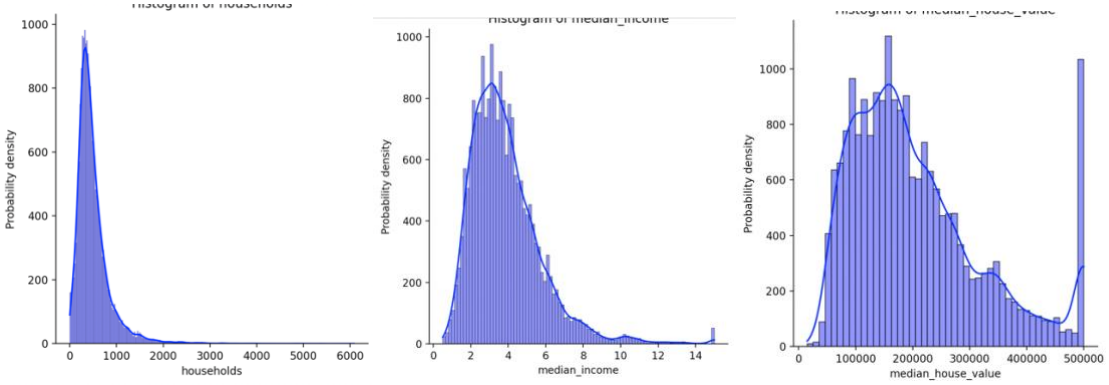
Κώδικας:

```
features = ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households',
'median_income', 'median_house_value']
for feature in features:
    plt.hist(cal_housing[feature], bins=50, density=True, alpha=0.6, color='b')
    plt.title(f'Histogram of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Probability density')
    plt.show()
```

Ένας άλλος τρόπος για να υπολογίσουμε την πυκνότητα πιθανότητας είναι με την βιβλιοθήκη της seaborn. Πιο συγκεκριμένα καλούμε την βιβλιοθήκη displot για να δημιουργήσουμε το ιστόγραμμα και στις παραμέτρους θέτουμε το kde=True για να εμφανίσει την πυκνότητα πιθανότητας των τιμών κάθε μεταβλητής(από κάθε στήλη) από το dataset πάνω στο γράφημα. Εκτελώντας το μέσα σε μια επανάληψη παίρνουμε όλα τα ιστογράμματα για τις 9 στήλες του dataset.

Παράδειγμα εκτέλεσης:





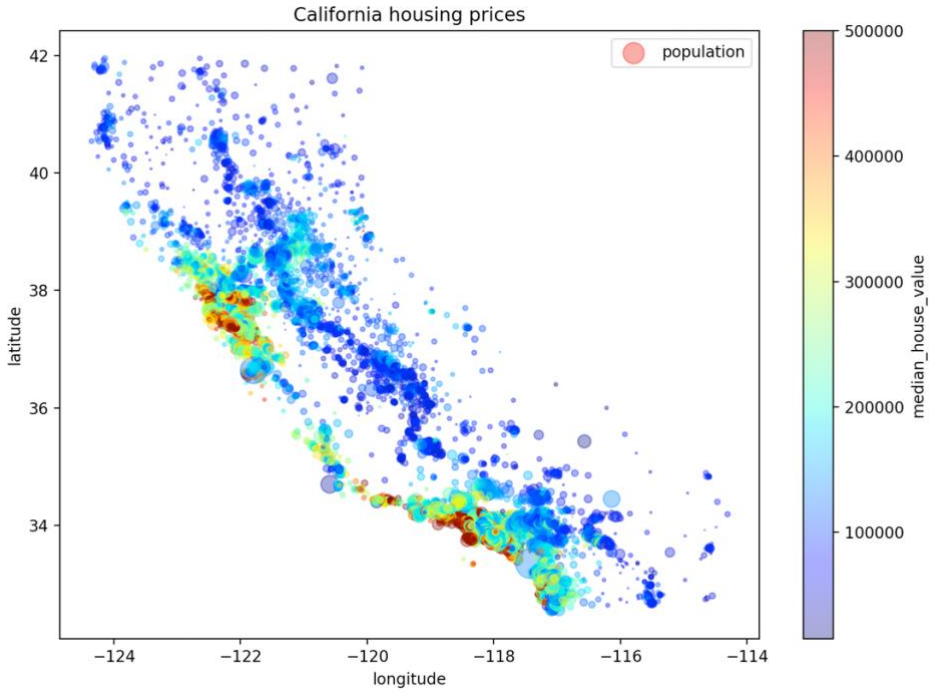
Κώδικας:

```
for feature in features:
    sns.displot(cal_housing[feature], kde=True, color='b')
    plt.title(f'Histogram of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Probability density')
    plt.show()
```

2) Σε αυτό το υποερώτημα θέλουμε να δημιουργήσουμε δισδιάστατα γραφήματα των δεδομένων στα οποία να αναπαρίστανται με ευδιάκριτο τρόπο συνδυασμοί 2, 3 ή και 4 μεταβλητών.

Ένα διάγραμμα που μπορούμε να δημιουργήσουμε είναι μια απεικόνιση του χάρτη της Καλιφόρνιας. Βάζοντας τις συντεταγμένες, τον πληθυσμό σε κάθε περιοχή και την μέση τιμή ακινήτων.
Αρχικά, θα εμφανίσουμε ένα scatter plot στο οποίο θα υπάρχουν κύκλοι ανάλογα με τον πληθυσμό σε αυτό το σημείο (παράμετρος s). Οι κύκλοι παίρνουν χρώμα ανάλογα με το πόσο ακριβά είναι τα ακίνητα στην κάθε περιοχή.

Παράδειγμα εκτέλεσης:

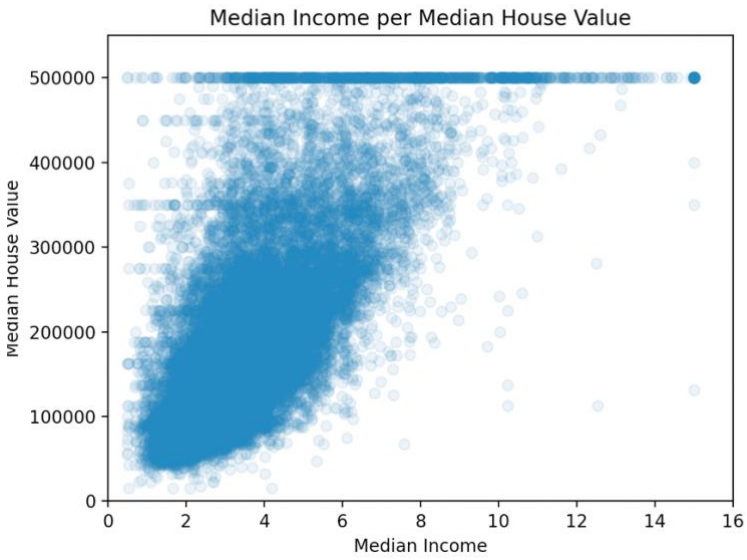


Κώδικας:

```
cal_housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
    s=cal_housing["population"]/100, label="population", figsize=(10,7),
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
    sharex=False)
plt.title('California housing prices')
plt.legend()
plt.show()
```

Ένα άλλο διάγραμμα που μπορούμε να δημιουργήσουμε είναι μια απεικόνιση μεταξύ median_income και median_house_value. Δημιουργούμε ένα scatterplot και το προβάλουμε.

Παράδειγμα εκτέλεσης:

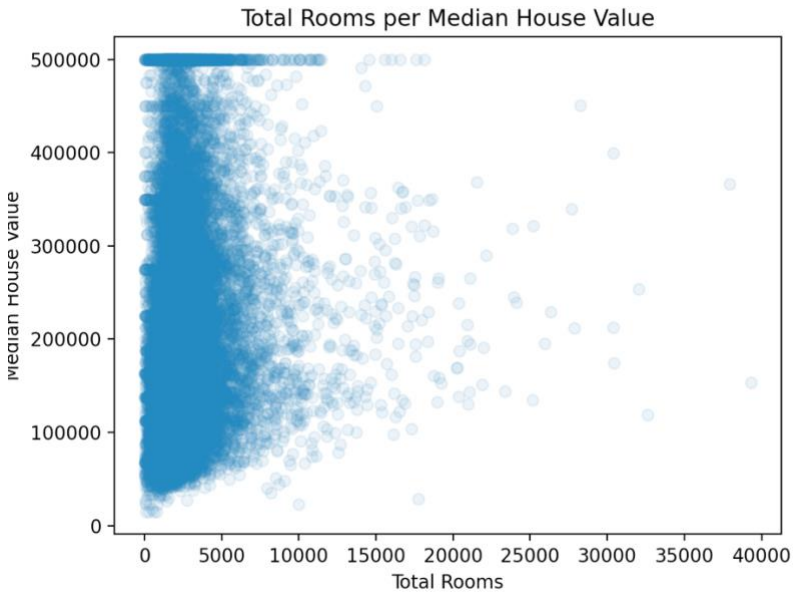


Κώδικας:

```
plt.scatter(cal_housing["median_income"], cal_housing["median_house_value"], alpha=0.1)
plt.xlabel("Median Income")
plt.ylabel("Median House Value")
plt.title("Median Income per Median House Value")
plt.axis([0, 16, 0, 550000])
plt.show()
```

Ένα άλλο διάγραμμα που μπορούμε να δημιουργήσουμε είναι μια απεικόνιση μεταξύ total_rooms και median_house_value. Δημιουργούμε ένα scatterplot και το προβάλουμε.

Παράδειγμα εκτέλεσης:



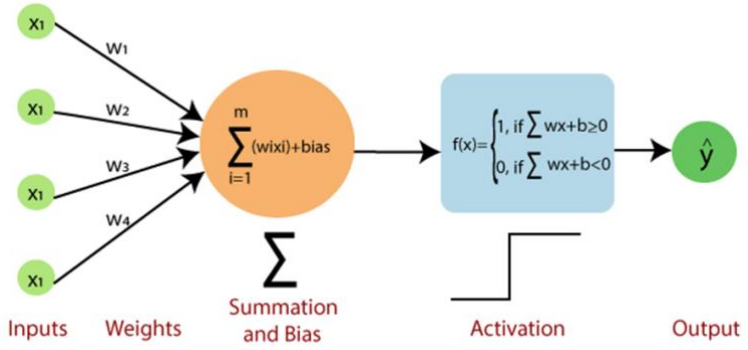
Κώδικας:

```
plt.scatter(cal_housing["total_rooms"], cal_housing["median_house_value"], alpha=0.1)
plt.xlabel("Total Rooms")
plt.ylabel("Median House Value")
plt.title("Total Rooms per Median House Value")
plt.show()
```

Παλινδρόμηση Δεδομένων

1) Στο πρώτο υποερώτημα υλοποιούμε ένα αλγόριθμο Perceptron, ώστε ο εκπαιδευμένος μηχανισμός μάθησης να υλοποιεί μία γραμμική συνάρτηση διάκρισης της μορφής $\boldsymbol{g}$: $\mathbb{R}^{\boldsymbol{l}} \rightarrow \{-\mathbf{1}, +\mathbf{1}\}$**,** όπου l είναι η διάσταση του τελικού χώρου των χαρακτηριστικών. Θεωρούμε κατώφλι την αξία των ακινήτων. Στο

συγκεκριμένο ερώτημα δεν χρησιμοποιούμε καμία έτοιμη βιβλιοθήκη όπως η sklearn για την ανάπτυξη του Perceptron, αλλά ούτε για την εκπαίδευση των δεδομένων (k- fold cross validation) και τον υπολογισμό σφάλματος (MSE και MAE).



Αρχικά, σε ένα καινούργιο αρχείο perceptron.py δημιουργούμε την κλάση Perceptron και κάνουμε import την μόνη βιβλιοθήκη που θα χρειαστούμε για να κάνουμε πράξεις, την numpy. Στην κλάση δημιουργούμε ένα constructor μέσα στον οποίο δηλώνουμε τα παρακάτω attributes: lr (learning_rate μπορεί να πάρει τιμές από 0.0 -0.1), n_iters(ο αριθμός των iterations), activation_func (καλείται από τον Perceptron γα να ξεκινήσει η διαδικασία) και το weight με το bias που τα θέτουμε ίσα με None (θα χρησιμοποιηθούν στην συνέχεια).
Στη συνέχεια δημιουργούμε μια μέθοδο fit που ουσιαστικά θα εκπαιδεύει τον αλγόριθμο μας. Από εκεί θα περνάμε σαν παραμέτρους το self και τα X, y που είναι τα training samples και τα training labels. Το πρώτο που ορίζουμε είναι οι διαστάσεις του X selector που είναι n*m. Μετά θέτουμε όλα τα weights ίσα με 0 μέσω του feature και τα bias ίσα με 0. Το επόμενο βήμα που έχουμε να κάνουμε είναι να ελέγξουμε ότι το y αποτελείται μόνο από κλάσεις 0 ή 1.
Έπειτα δημιουργούμε τις επαναλήψεις με τις οποίες θα κάνουμε update στα δεδομένα εκπαίδευσης μας. Η πρώτη θα τρέχει μέχρι το τέλος των iterations και η δεύτερη χρησιμοποιεί την συνάρτηση enumerate για να πάρουμε το index και το τρέχων sample προκειμένου να εφαρμόσουμε τον κανόνα ενημέρωσης. Υπολογίζουμε την predicted value και εφαρμόζουμε την ενημέρωση για να υπολογίσουμε το linear outpur. Καλούμε την συνάρτηση ενεργοποίησης και παίρνουμε την predicted value. Τέλος, ενημερώνουμε τα weights και το biased βασιζώμενοι στο error μεταξύ predicted και actual output.
Βασιζόμαστε στην παρακάτω φωτογραφία:

For each training sample x_i :

$$w := w + \Delta w$$

$$\Delta w := \alpha \cdot (y_i - \hat{y}_i) \cdot x_i$$

α : learning rate in [0, 1]

Η συνάρτηση predict θα έχει ως παράμέτρους το self με το X που είναι τα training samples.

Πρώτα εκτελούμε την γραμμική συνάρτηση και έπειτα την συνάρτηση ενεργοποίησής.

$$\hat{y} = g(f(w,b)) = g(w^T x + b)$$

Έχοντας ολοκληρώσει την δημιουργία της κλάσης πρέπει να την καλέσουμε μέσω αντικειμένου και έπειτα να κατασκευάσουμε έναν αλγόριθμο k- fold cross validation και τέλος να υπολογίζουμε το error με MSE και MAE. Όλα αυτά γίνονται πίσω στο κεντρικό μας αρχείο.
Αρχικά κάνουμε import την κλάση από το αρχείο perceptron.py και φορτώνουμε το αρχείο csv με τα δεδομένα μας τα οποία θα χωρίσω σε training samples και test samples.
Το επόμενο που κάνουμε είναι να φτιάξουμε μια συνάρτηση k-fold cross validation στην οποία έχουμε ορίσει ότι το k θα είναι ίσο με 10. Μέσα σε μια επανάληψη k φορών δηλώνουμε την αρχή και το τέλος του test data από το κάθε fold. Αφού οριστούν οι πίνακες εκπαίδευσης X_train και y_train για τα training samples και για τα training labels δημιουργούμε ένα αντικείμενο από την κλάση Perceptron και περνάμε ως παραμέτρους στον constructor learning rate και αριθμό iterations. Καλούμε τις μεθόδους εκπαίδευσης και στην συνέχεια υπολογίζουμε το μέσα τετραγωνικό σφάλμα (MSE) και το μέσο απόλυτο σφάλμα (MAE) βάση των παρακάτω εξισώσεων:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \qquad MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|$$

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition %
/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/patterr
Mean classification accuracy: 0.889050387596899
Mean Squared Error: 0.11094961240310078
Mean Absolute Error: 0.11094961240310078
```

Κώδικας:

patternRec.py (main αρχείο)

```
from perceptron import Perceptron


if __name__ == "__main__":

    cal_housing = pd.read_csv('housing.csv')
```

```
def accuracy(y_true, y_pred):
    accuracy = np.sum(y_true == y_pred) / len(y_true)
    return accuracy

def mean_squared_error(y_true, y_pred):
    mse = np.mean((y_true - y_pred) ** 2)
    return mse

def mean_absolute_error(y_true, y_pred):
    mae = np.mean(np.abs(y_true - y_pred))
    return mae


X = cal_housing.iloc[:, :-2].values
y = cal_housing.iloc[:, -2].values

k = 5
fold_size = len(X) // k

accuracies = []
mses = []
maes = []

# k-fold cross validation
for i in range(k):
    test_start = i * fold_size
    test_end = (i + 1) * fold_size
    train_indices = list(range(test_start)) + list(range(test_end, len(X)))

    X_train, X_test = X[train_indices], X[test_start:test_end]
    y_train, y_test = y[train_indices], y[test_start:test_end]

    p = Perceptron(learning_rate=0.01, n_iters=1000)
    p.fit(X_train, y_train)
    predictions = p.predict(X_test)

    acc = accuracy(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    mae = mean_absolute_error(y_test, predictions)
```

```
        accuracies.append(acc)
        mses.append(mse)
        maes.append(mae)


    mean_accuracy = np.mean(accuracies)
    mean_mse = np.mean(mses)
    mean_mae = np.mean(maes)


    print("Mean classification accuracy:", mean_accuracy)
    print("Mean Squared Error:", mean_mse)
    print("Mean Absolute Error:", mean_mae)
```

perceptron.py

```
import numpy as np

class Perceptron:
    def __init__(self, learning_rate=0.01, n_iters=1000):
        self.lr = learning_rate
        self.n_iters = n_iters
        self.activation_func = self._unit_step_func
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        y_ = np.array([1 if i > 0 else 0 for i in y])

        for _ in range(self.n_iters):

            for idx, x_i in enumerate(X):
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_predicted = self.activation_func(linear_output)
                update = self.lr * (y_[idx] - y_predicted)
```

```
            self.weights += update * x_i
            self.bias += update

    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        y_predicted = self.activation_func(linear_output)
        return y_predicted

    def _unit_step_func(self, x):
        return np.where(x >= 0, 1, 0)
```

2) Στο δεύτερο υποερώτημα αναπτύσσουμε Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (Least Squares), ώστε ο εκπαιδευμένος μηχανισμός μάθησης να υλοποιεί μία γραμμική παλινδρόμηση της μορφής $\boldsymbol{g}$: $\mathbb{R}^{\boldsymbol{l}} \rightarrow \{-\mathbf{1}, +\mathbf{1}\}$, όπου l είναι η διάσταση του τελικού χώρου των χαρακτηριστικών. Σε αυτό το υποερώτημα δεν κάνουμε χρήση έτοιμης βιβλιοθήκης.

Αρχικά, η πρώτη συνάρτηση που θα δούμε στο πρόγραμμα είναι η k-fold cross validation η οποία έχει ως παραμέτρους τον αριθμό N των samples και το k ίσο με 10 που είναι ο αριθμός των folds. Η συνάρτηση αυτή υπολογίζει και επιστρέφει δυο λίστες από train indices και test indices για κάθε fold.
Η επόμενη συνάρτηση που θα συναντήσουμε είναι η linear_regression_fit η οποία θα έχει ως παραμέτρους τα input features X, την target variable Y και τον αριθμό των fold που είναι 10. Η συνάρτησση αυτή σε πρώτο στάδιο καλεί την k-fold cross validation και στην συνέχεια πραγματοποιεί μια επανάληψη για κάθε fold και υπολογίζει τις μεταβλητές (m και c) για το least squares.

$$m = \frac{n\sum xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$

$$c = y - mx$$

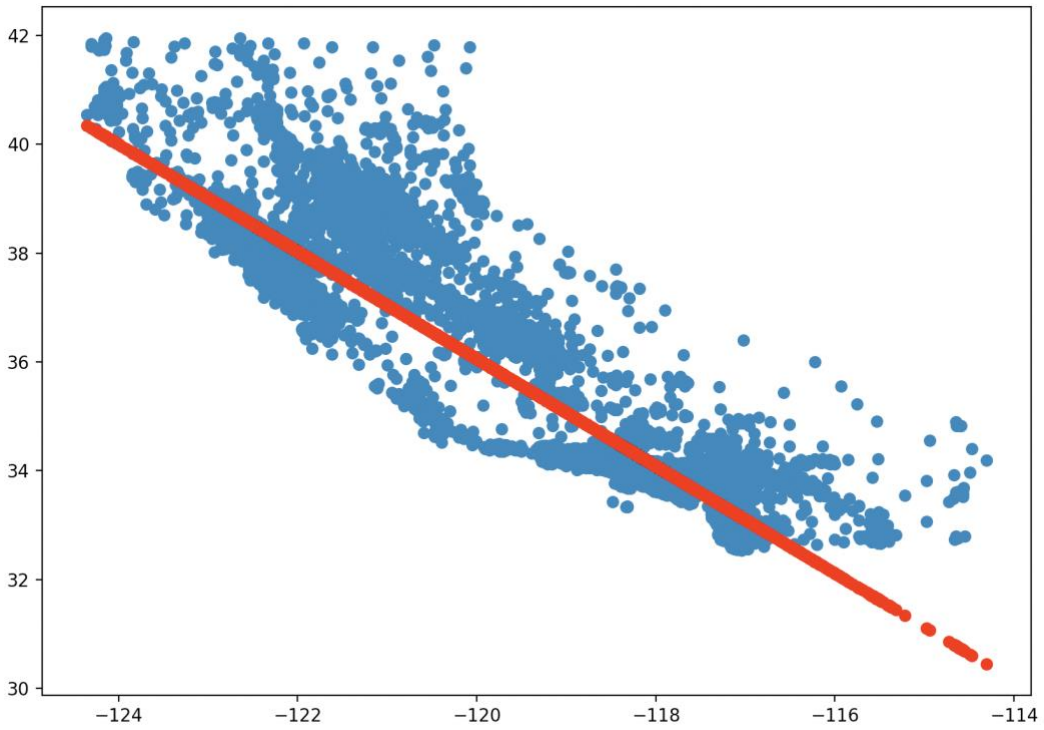
Επίσης μέσα σε αυτήν την συνάρτηση υπολογίζουμε το mse και το msa.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \qquad MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|$$

Από τον main code μας αποθηκεύομε τις τιμές από την πρώτη στήλη και την δεύτερη στήλη, καλούμε την συνάρτηση linear_regression_fit και εκτυπώνουμε το μέσο λάθος για τα mse και τα mae από όλα τα folds. Τέλος, εμφανίζουμε ένα plot με την γραμμική μας ευθεία.

Παράδειγμα εκτέλεσης:

```
(base) george_vellos@Georges-MacBook-Air-2 Pattern recognition % /Users/george_vellos/opt/anaconda3/bin/python "/Use
rs/george_vellos/Desktop/INFORMATICS_UNIPI/5 SEMESTER/Pattern recognition/patternRec.py"
Mean Squared Error (MSE) Scores: [0.8817645505747463, 0.7653214104899981, 0.1546397170060548, 0.09413279709420724, 0.786530156
6106522, 0.5870634771213011, 1.1224928591721752, 0.30559356431238377, 1.2274194164503671, 1.356856208136826]
Mean Absolute Error (MAE) Scores: [0.6658117402011529, 0.6491066846970038, 0.3616817973951053, 0.26438099119440917, 0.65227205
69811259, 0.48568569368812337, 0.9267241504958407, 0.5061349840306107, 1.0487133698906503, 0.9361602957691048]
Average MSE: 0.7281814156968711
Average MAE: 0.6496671764343127
```



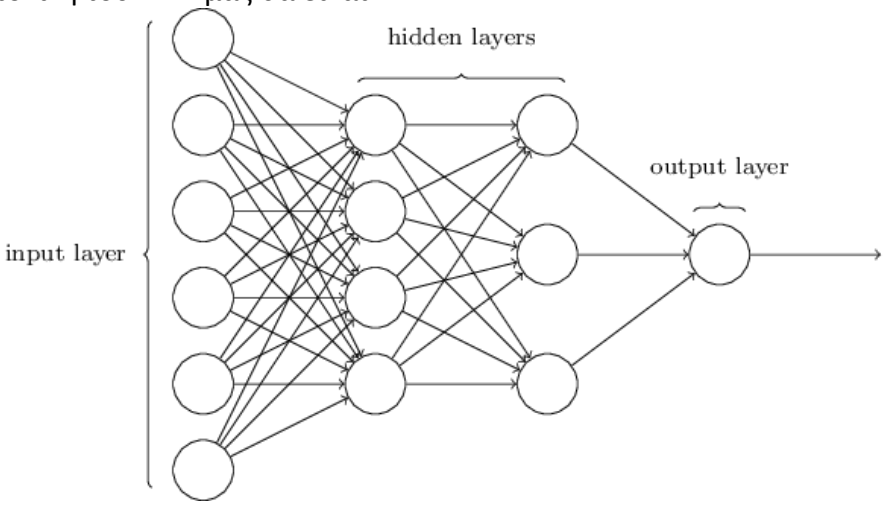
Κώδικας εκτέλεσης:

```python
# least squares
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


def kfold_indices(N, K):
    indices = np.arange(N)
    M = N // K
    if N % K != 0:
        raise ValueError("The number of elements within vector Indices must be fully divided by K")
    else:
        train_indices = []
        test_indices = []
```

```
        for k in range(K):
            start = k * M
            end = (k + 1) * M if k < K - 1 else N
            test_indices.append(indices[start:end])
            train_indices.append(np.setdiff1d(indices, test_indices[-1]))
    return train_indices, test_indices


def linear_regression_fit(X, Y, K=10):
    mse_scores = []
    mae_scores = []


    train_indices, test_indices = kfold_indices(len(X), K)

    for train_index, test_index in zip(train_indices, test_indices):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        Y_train, Y_test = Y.iloc[train_index], Y.iloc[test_index]

        X_train_mean = np.mean(X_train)
        Y_train_mean = np.mean(Y_train)

        num = np.sum((X_train - X_train_mean) * (Y_train - Y_train_mean))
        den = np.sum((X_train - X_train_mean) ** 2)
        m = num / den
        c = Y_train_mean - (m * X_train_mean)

        Y_pred = m * X_test + c

        mse = np.mean((Y_test - Y_pred) ** 2)
        mae = np.mean(np.abs(Y_test - Y_pred))

        mse_scores.append(mse)
        mae_scores.append(mae)


    plt.scatter(X_test, Y_pred, color='red')
    return mse_scores, mae_scores
```

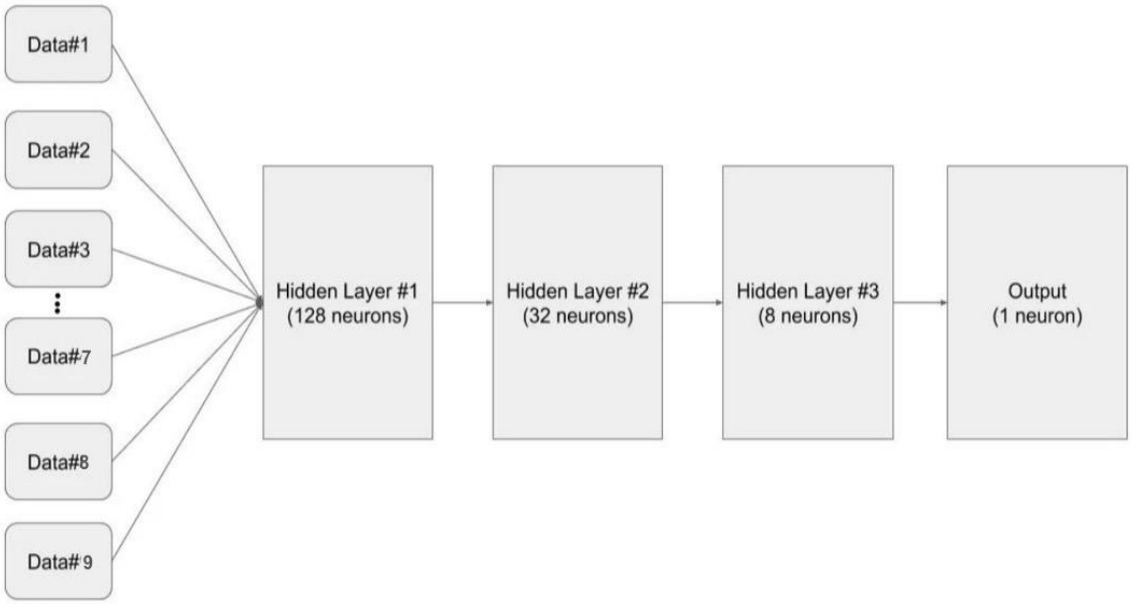
```
if __name__ == "__main__":

    X = cal_housing.iloc[:, 0]
    Y = cal_housing.iloc[:, 1]

    plt.scatter(X, Y)

    mse_scores, mae_scores = linear_regression_fit(X, Y)

    print("Mean Squared Error (MSE) Scores:", mse_scores)
    print("Mean Absolute Error (MAE) Scores:", mae_scores)
    print("Average MSE:", np.mean(mse_scores))
    print("Average MAE:", np.mean(mae_scores))

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.title("Linear Regression Fit")
    plt.show()
```

3) Σε αυτά υποερώτημα θα υλοποιήσουμε ένα πολυστρωματικό νευρωνικό δίκτυο, ώστε ο εκπαιδευμένος μηχανισμός μάθησης να υλοποιεί μία μη-γραμμική παλινδρόμηση της μορφής $\boldsymbol{g}$: $\mathbb{R}^{\boldsymbol{l}} \rightarrow \mathbb{R}$,όπου l είναι η διάσταση του τελικού χώρου των χαρακτηριστικών. Θα εκπαιδεύσουμε ένα νευρωνικό δίκτυο για regression predictions χρησιμοποιώντας την βιβλιοθήκη Keras με back-end την TensorFlow. Η αρχιτεκτονική του MLP μας θα είναι:



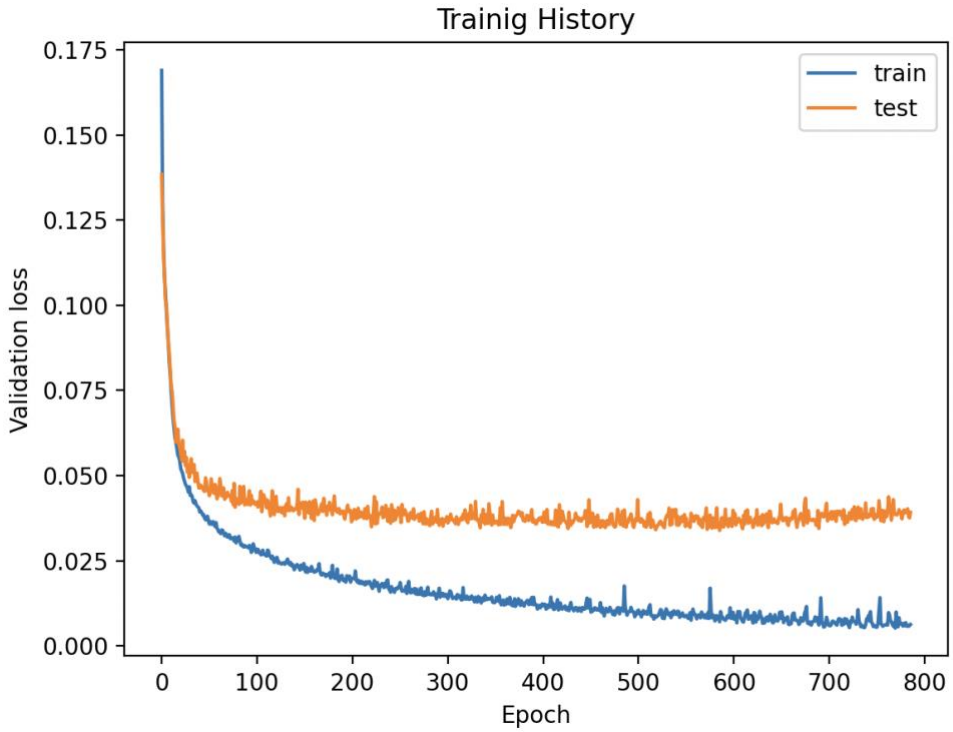
Δηλαδή θα έχουμε ως είσοδο τις 9 στήλες με τα δεδομένα,3 hidden layers με 128,32,8 νευρώνες και μια έξοδο:

Το πρώτο που κάνουμε είναι να εισάγουμε τις κατάλληλες βιβλιοθήκες, φορτώνουμε τα δεδομένα μας τα οποία έχουν υποστεί προεπεξεργασία από προηγούμενα βήματα. Σπάμε τα δεδομένα μας σε training και validation με την training_test_split με αναλογία 80/20. Στη συνέχεια δημιουργούμε το νευρωνικό μας δίκτυο μέσω της keras. Η πρώτη στρώση αποτελείται από 128 νευρώνες, η συνάρτηση ενεργοποίησης είναι η ReLu και η παράμετρος input_dim είναι ίση με τον αριθμό των στηλών. Η δεύτερη στρώση αποτελείται από 32 νευρώνες και έχει συνάρτηση ενεργοποίησης την ReLu. Η τρίτη και τελευταία hidden layer περιέχει 8 νευρώνες και είναι ReLu. Η στρώση εξόδου περιέχει 1 νευρώνα και έχει γραμμική συνάρτηση ενεργοποίησής. Το επόμενο που δημιουργούμε είναι ένα compile model το οποίο αρχικοποιείται με τον optimizer Adam και μετά γίνεται compiled. Στο μοντέλο ορίζεται και η συνάρτηση για τον υπολογισμό της απώλειας(MAE) που σημαίνει ότι επιδιώκεται η ελαχιστοποίηση της μέσης ποσοστιαίας διαφοράς μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Συμπαντικό είναι ο αριθμός των epoch που θα ορίσουμε είναι πολύ σημαντικός καθώς η επιλογή πολύ μεγάλου αριθμού μπορεί να δημιουργήσει overfitting στα trainig data. Αλλά και η επιλογή μικρού αριθμού epoch μπορεί να δημιουργήσει under-fitting. Για να μην δημιουργηθεί το πρώτο πρόβλημα δημιουργήσαμε μια early stoping μέθοδο η οποία μας επιτρέπει να ορίσουμε ένα μεγάλο



αριθμό epoch καθώς μόλις παρατηρηθεί ότι δεν υπάρχει βελτίωση στην απόδοση του μοντέλου.
Τέλος προβάλουμε ένα γραφήματα με το πως πήγε η εκπαίδευση και εκτυπώνουμε το R-Squared για το trainig set.

Παράδειγμα εκτέλεσης:




```
Epoch 785/10000000
166/166 - 0s - loss: 0.0059 - val_loss: 0.0375 - 134ms/epoch - 806us/step
Epoch 786/10000000
166/166 - 0s - loss: 0.0062 - val_loss: 0.0392 - 134ms/epoch - 808us/step
Epoch 786: early stopping
516/516 [==============================] - 0s 368us/step
129/129 [==============================] - 0s 372us/step
Validation Set R-Square= 0.8404413043206272
(tf) (base) george_vellos@Georges-MacBook-Air-2 Pattern recognition %
```

Κώδικας εκτέλεσης:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score


# Τα ξανακάνω απλά για να φανούν και εδώ

dataset = pd.read_csv('housing.csv')

median_value = dataset['total_bedrooms'].median()
dataset['total_bedrooms'].fillna(median_value, inplace=True)

# Encode categorical variables using one-hot encoding
dataset = pd.get_dummies(dataset, columns=['ocean_proximity'])


X = dataset.iloc[:,0:9]
Y = dataset.iloc[:,9]

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.3, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)

# Create model
model = Sequential([
    Dense(128, activation="relu", input_dim=X_train.shape[1]),
    Dense(32, activation="relu"),
    Dense(8, activation="relu"),
    Dense(1, activation="linear")
])

# Compile model
model.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=1e-3, decay=1e-3 / 200))

# Patient early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=200)

# Fit the model
```

```
history = model.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=10000000, batch_size=100,
verbose=2, callbacks=[es])

# Calculate predictions
PredTestSet = model.predict(X_train)
PredValSet = model.predict(X_val)

# Plot loss history
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.title('Trainig History')
plt.xlabel('Epoch')
plt.ylabel('Validation loss')
plt.legend()
plt.show()

ValR2Value = r2_score(Y_val, PredValSet)
print("Validation Set R-Square=", ValR2Value)
```

Βιβλιογραφία

[1] Εισαγωγή στην Αναγνώριση Προτύπων με MATLAB, S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras, 2011, Εκδόσεις Broken Hill Publishers Ltd.

[2] https://mathesis.cup.gr/courses/course-v1:ComputerScience+CS5.1+23B/about

[3] https://pandas.pydata.org/docs/

[4] https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

[5] https://keras.io/api/