ΕΥΣΤΡΑΤΙΟΣ ΚΑΡΚΑΝΗΣ – Π19064

ΧΑΡΑΛΑΜΠΟΣ ΧΡΙΣΤΟΦΟΡΙΔΗΣ – Π19188

ΓΕΩΡΓΙΟΣ ΣΕΪΜΕΝΗΣ – Π19204

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΣΕΠΤ. 2022

# Εισαγωγή

Η ανάπτυξη της εφαρμογής έγινε με τη γλώσσα Python, με το interface του Jupyter, χρησιμοποιώντας το Visual Studio Code και διάφορες χρήσιμες βιβλιοθήκες, που βοήθησαν πολύ με την οπτικοποίηση και την παλινδρόμηση των δεδομένων. Παρακάτω θα εξηγηθούν αναλυτικά οι χρήσεις των βιβλιοθηκών και η σημασία τους στη διευκόλυνση πολλών διαδικασιών. Συγκεκριμένα, οι βιβλιοθήκες που χρησιμοποιούνται είναι οι:

* numpy
* pandas
* matplotlib
* seaborn
* sklearn

Όλες οι παραπάνω είναι απαραίτητες για την εκτέλεση της εφαρμογής.

# Προ-επεξεργασία Δεδομένων

## ΜΕΡΟΣ 1ο: Αναγνώριση της Πληροφορίας

Το αρχείο CSV, που περιέχει τα δεδομένα, διαβάστηκε με τη χρήση της βιβλιοθήκης pandas, ώστε να έχουμε πλήρη εικόνα των μετρήσεων των δεδομένων. Όπως φαίνεται και παρακάτω, τα δεδομένα που έχουμε είναι 8 (οκτώ) στήλες τύπου αριθμού κινητής υποδιαστολής και μίας στήλης τύπου αλφαριθμητικού.

A screenshot of a computer

Description automatically generated with medium confidence

Εικόνα 1: Τα πρώτα 5 (πέντε) δεδομένα του αρχείου CSV.

Text

Description automatically generated

Εικόνα 2: Αναγνώριση των τύπων δεδομένων των στηλών.

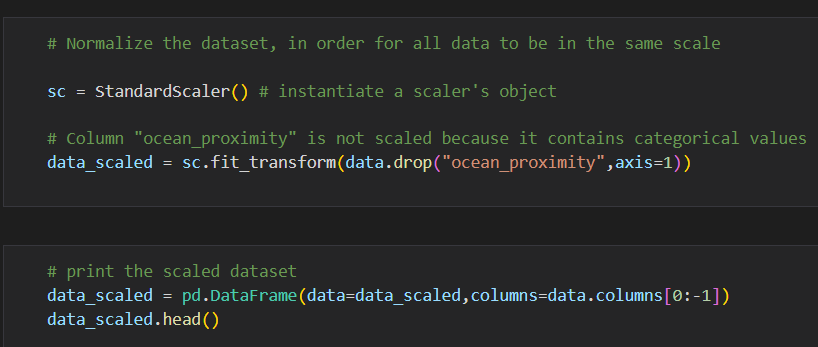
## Μέρος 2ο: Κλιμάκωση Δεδομένων

Εφ’ όσον έχουμε οκτώ στήλες στις οποίες εμπεριέχονται αριθμοί, θα πρέπει να κάνουμε κλιμάκωση. Δηλαδή, θα πρέπει όλες οι τιμές να ανήκουν στο ίδιο διάστημα. Όσο για την άλλη στήλη, που δεν έχει από μόνη της αριθμητικά δεδομένα, θα ακολουθήσουμε άλλη μεθοδολογία, ώστε να έχουμε αριθμητικές τιμές. Παρατηρούμε ότι έχει 5 (πέντε) διαφορετικές τιμές:

* <1H OCEAN
* INLAND
* ISLAND
* NEAR BAY
* NEAR OCEAN

Άρα, θα διαγράψουμε τη στήλη που έχει αλφαριθμητικές τιμές και αντ’ αυτής, θα βάλουμε πέντε επιπλέον στήλες. Αυτές οι πέντε στήλες θα αντιστοιχούν στα παραπάνω ονόματα.

Για αρχή, κλιμακώνουμε τα δεδομένα μας, ώστε όλα να ανήκουν στο σύνολο [-3, 3].



Εικόνα 3: Κλιμάκωση των στηλών με τις αριθμητικές τιμές.

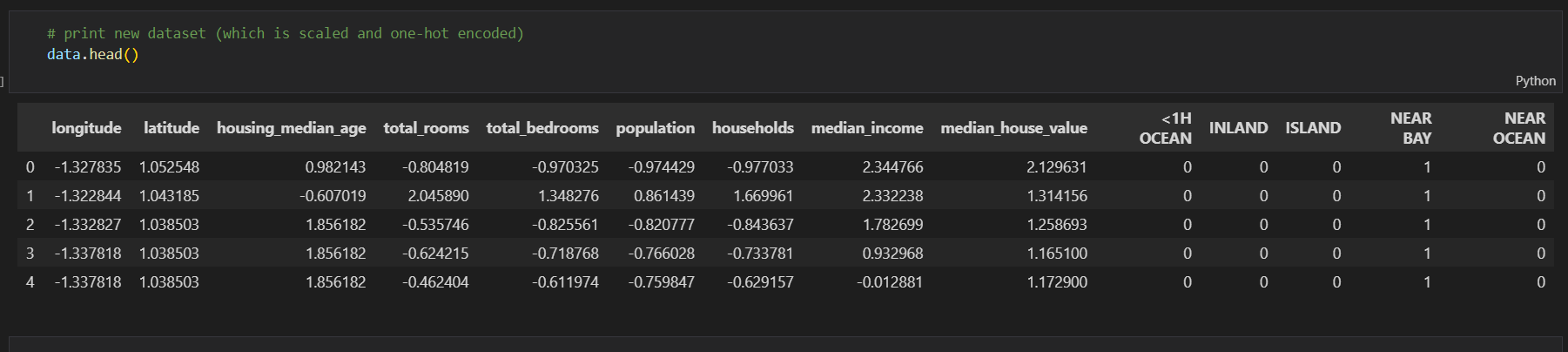
Ύστερα, «μαζεύουμε» τα δεδομένα της στήλης στην οποία εμπεριέχονται τα αλφαριθμητικά και τα κάνουμε όλα πέντε ξεχωριστές στήλες.

Text

Description automatically generated

Εικόνα 4: Συμμάζεμα των δεδομένων της στήλης με τα αλφαριθμητικά.

Οπότε, η τελική μορφή του πίνακα, μετά την κλιμάκωση των δεδομένων, είναι η εξής:

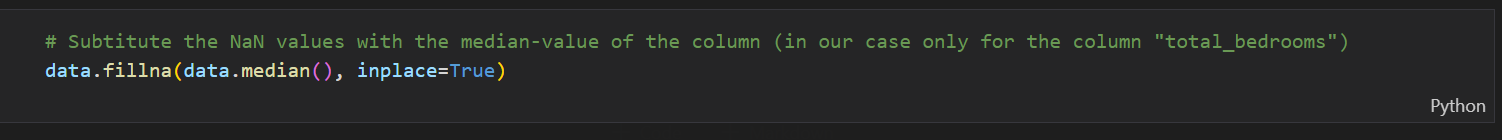


Εικόνα 5: Τα δεδομένα του πίνακα μετά την κλιμάκωση.

## Μέρος 3ο: Καθαρισμός των Δεδομένων

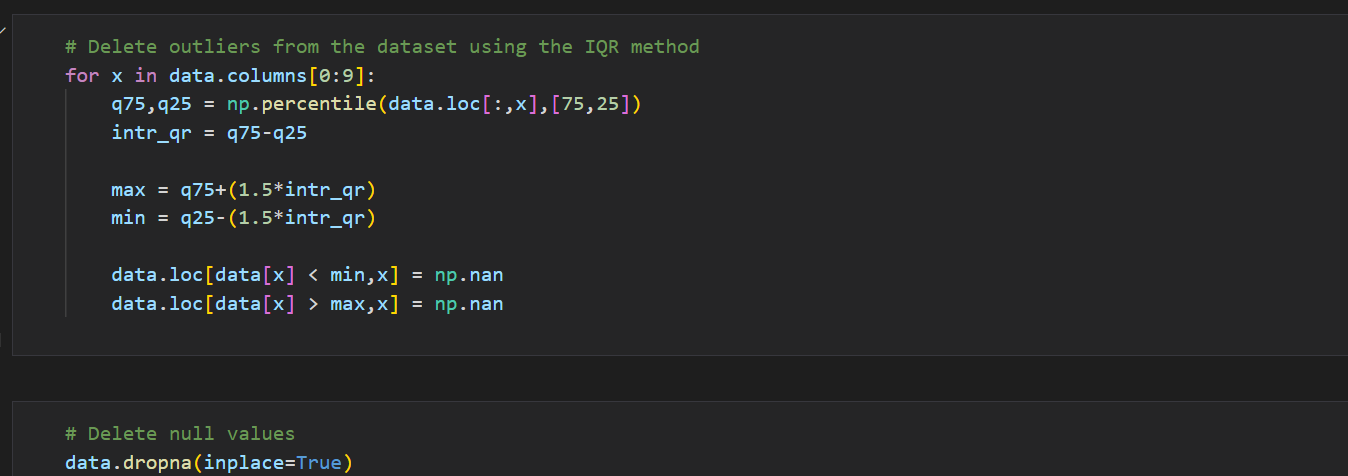
Στο τελικό στάδιο της προ-επεξεργασίας δεδομένων, αποφασίσαμε να ασχοληθούμε και με τις τιμές που είτε **δεν υπήρχαν** (NaN τιμές σε κάποιο πεδίο εγγραφών), είτε ήταν σε κάποια **ακραία μεγάλη ή μικρή τιμή** (outliers). Αναλυτικότερα:

* Για τις κενές τιμές, **αποφασίσαμε να αντικαταστήσουμε την NaN τιμή με την μέση τιμή (median value)** της εκάστοτε στήλης. Ο λόγος που δεν επιλέξαμε να διαγράψουμε ολόκληρη τη στήλη, ήταν επειδή υπήρχε ο κίνδυνος μεγάλης απώλειας πληροφορίας, συνεπώς οδηγούσε στην πιθανή αλλοίωση των αποτελεσμάτων μας.



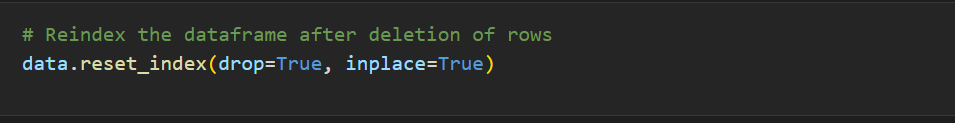
Εικόνα 6: Συμπλήρωση τιμών στο πεδίο "total\_bedrooms", το οποίο είχε NaN τιμές.

* Για τις **ακραίες τιμές** που συναντήσαμε (outliers), **αποφασίσαμε να διαγράψουμε οριστικά τις γραμμές** **στις οποίες βρίσκονταν**. Σε αυτή την περίπτωση, δεν υπήρχε ο κίνδυνος μεγάλης απώλειας, καθώς τέτοιου είδους «ανωμαλίες» είναι εξαιρετικά σπάνιες σε ένα σύνολο δεδομένων (σε εμάς ήταν κάπου στις 300 εγγραφές). Για την διαδικασία εξάλειψης χρησιμοποιήσαμε την μέθοδο IQR.



Εικόνα 7: Μέθοδος IQR για την εξάλειψη ακραίων τιμών.

* Τέλος, για να αποφύγουμε τυχόν προβλήματα με τους δείκτες (index) του dataframe, λόγω των πολλών διαγραφών, κάναμε re-index.

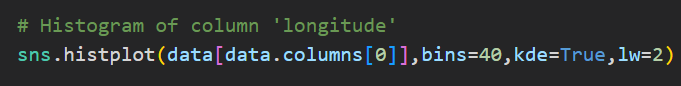


Εικόνα 8: Μέθοδος Re-Index μετά την διαγραφή των δεδομένων.

# Οπτικοποίηση Δεδομένων

## Μέρος 1ο: Ιστογράμματα Συχνοτήτων

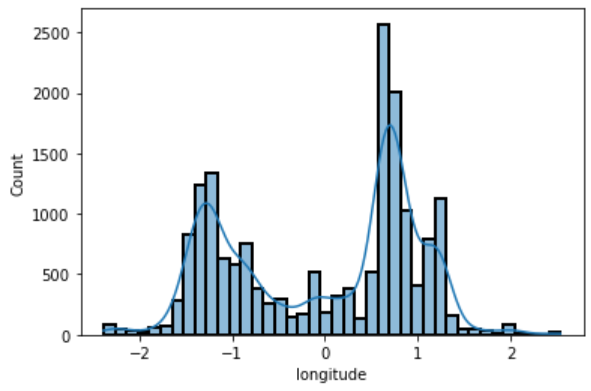
Στο πρώτο κομμάτι της οπτικοποίησης, κληθήκαμε να οπτικοποιήσουμε τα **ιστογράμματα συχνοτήτων** για κάθε μία από τις στήλες/μεταβλητές μας. Χρησιμοποιήσαμε την βιβλιοθήκη *seaborn* και *matplotlib* της Python, που έχουν ειδική συνάρτηση δημιουργίας ιστογραμμάτων.



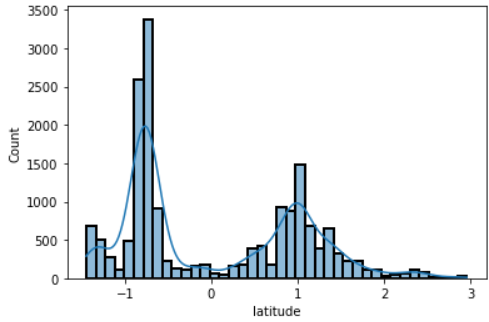
Εικόνα 9: Παράδειγμα για το πώς παράγουμε ένα ιστόγραμμα με την βιβλιοθήκη.

Τα ιστογράμματα που παρήχθησαν είναι τα εξής:

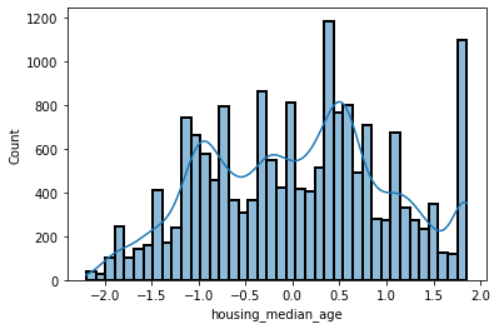
#### LONGITUDE:



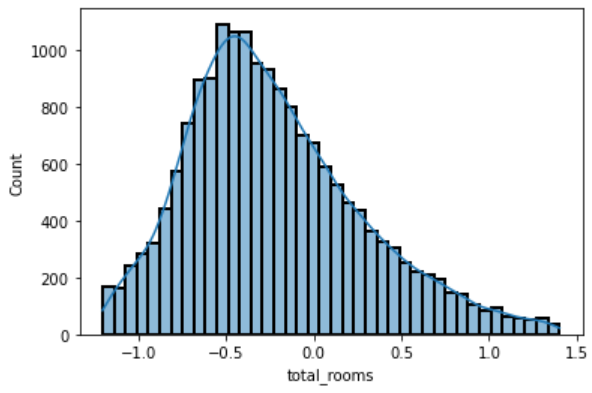
#### LATITUDE:



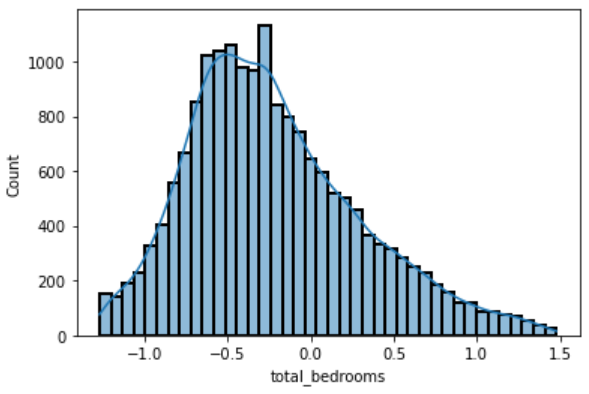
#### HOUSING MEDIAN AGE:



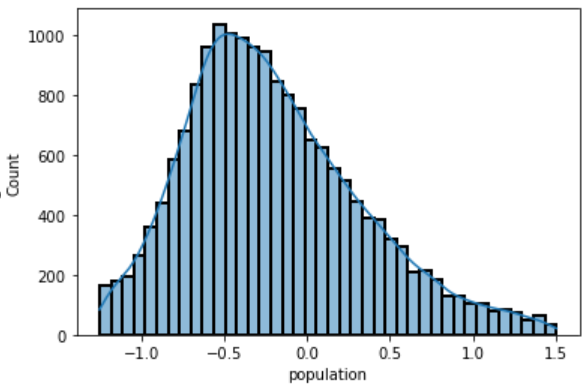
#### TOTAL ROOMS:



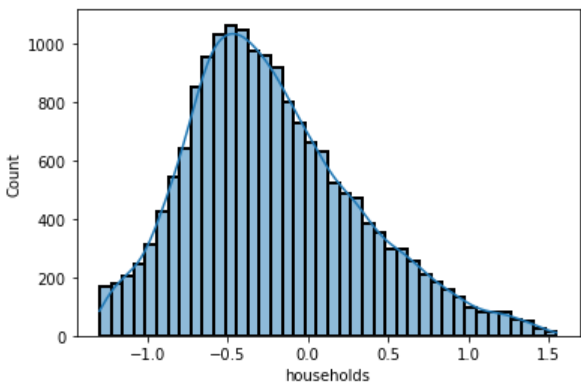
#### TOTAL BEDROOMS:



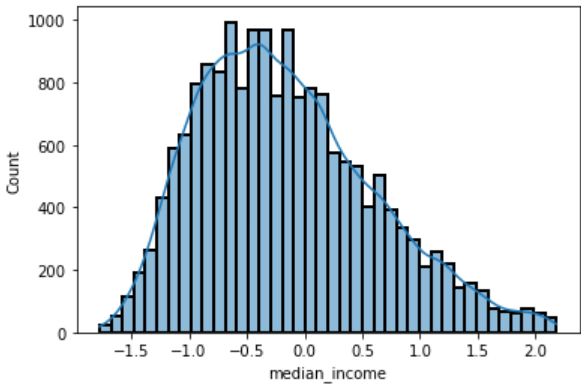
#### POPULATION:



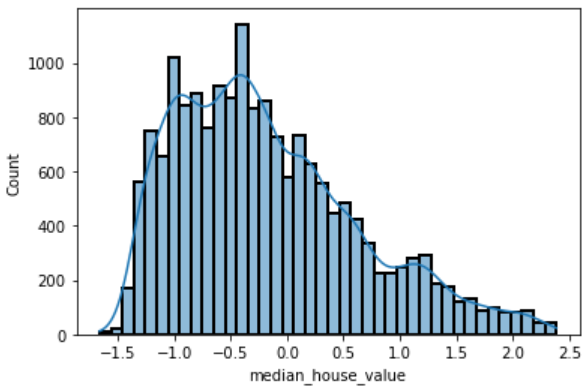
#### HOUSEHOLDS:



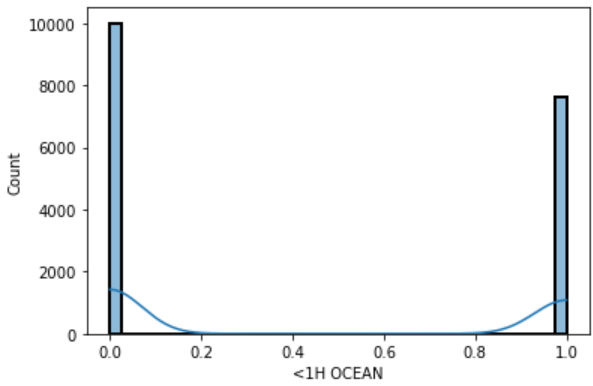
#### MEDIAN INCOME:



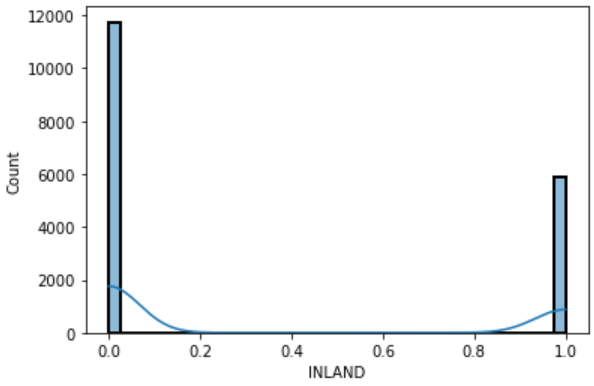
#### MEDIAN HOUSE VALUE:



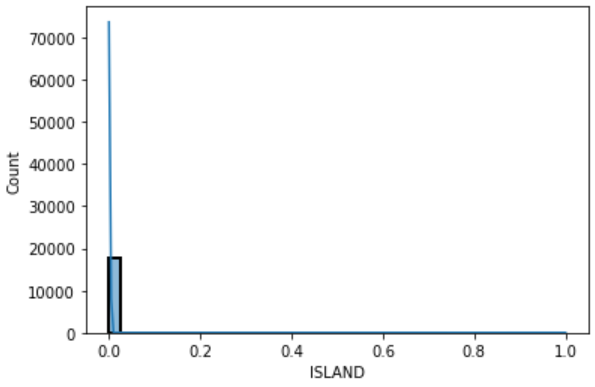
#### <1H OCEAN:



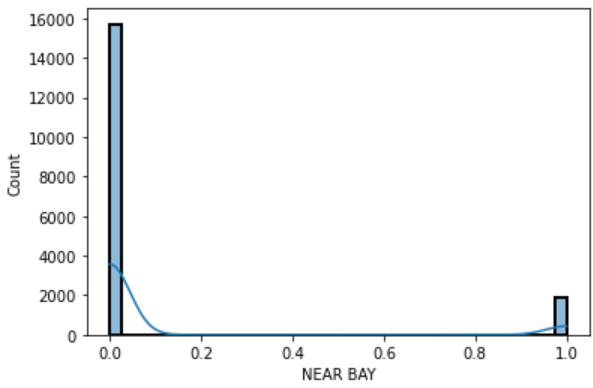
#### INLAND:



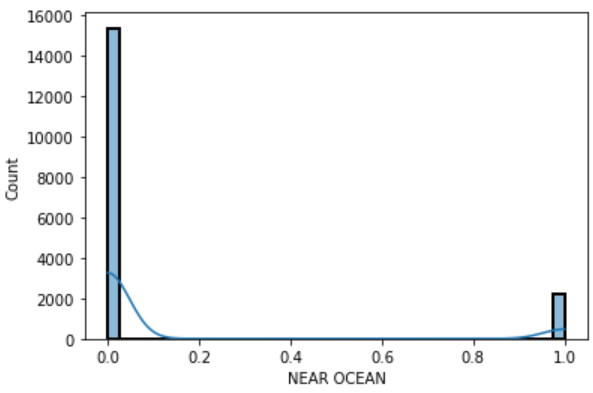
#### ISLAND:



#### NEAR BAY:



#### NEAR OCEAN:



## Μέρος 2ο: Δισδιάστατα Γραφήματα

Ιδιαίτερο ενδιαφέρον προσφέρει το γράφημα του **γεωγραφικού πλάτους (latitude) συναρτήσει του γεωγραφικού μήκους (longitude).** Το συγκεκριμένο γράφημα θυμίζει πολύ τον χάρτη της Καλιφόρνια. Για να επιβεβαιωθεί αυτό, μπορούμε να βάλουμε ως **hue** την στήλη «NEAR OCEAN». Με αυτόν τον τρόπο, με το μπλε χρώμα θα αναπαρασταθούν τα σπίτια που δεν είναι κοντά στον ωκεανό, σε αντίθεση με τα πορτοκαλί, τα οποία είναι κοντά στον ωκεανό.

Chart

Description automatically generated

Εικόνα 10: Γεωγραφικό Πλάτος συναρτήσει του Γεωγραφικού Μήκους.

Επιπλέον, με τα παρακάτω γραφήματα, **μπορούμε να δούμε το τι γίνεται με την αύξηση του πληθυσμού στα νοικοκυριά**. Συγκεκριμένα, ένα γεγονός, το οποίο βγάζει νόημα, **είναι η αναλογική αύξηση των νοικοκυριών με τον πληθυσμό** της εκάστοτε περιοχής. Όπως φαίνεται και στο επόμενο γράφημα, η παραπάνω πρόταση επιβεβαιώνεται.

Chart, scatter chart

Description automatically generated

Εικόνα 11: Νοικοκυριά συναρτήσει του Πληθυσμού.

Παρόμοιο, μπορούμε να δούμε ότι γίνεται με τα **συνολικά δωμάτια και με τα υπνοδωμάτια σε κάθε σπίτι**. Δηλαδή, **όσο αυξάνονται τα συνολικά δωμάτια ενός νοικοκυριού, αυξάνονται και τα υπνοδωμάτια**. Βέβαια, μπορούμε να παρατηρήσουμε, παράλληλα, το γεγονός ότι **κάποια νοικοκυριά έχουν σταθερό αριθμό υπνοδωματίων, όσο κι αν αυξηθούν τα συνολικά δωμάτια.** Για να μπλέξουμε και μία τρίτη μεταβλητή στο γράφημα, όπως και στο πρώτο, θα βάλουμε ως **hue** τον πληθυσμό.

Chart, scatter chart

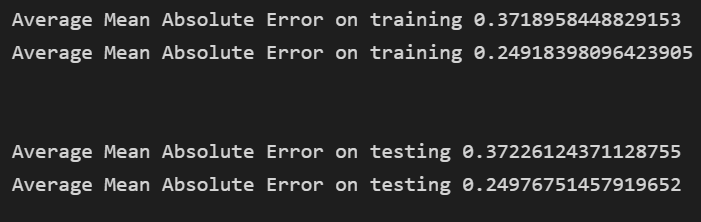
Description automatically generated

Εικόνα 12: Συνολικά Υπνοδωμάτια συναρτήσει των Συνολικών Υπνοδωματίων.

# Παλινδρόμηση Δεδομένων

## Μέρος 2ο: Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (Least Squares)

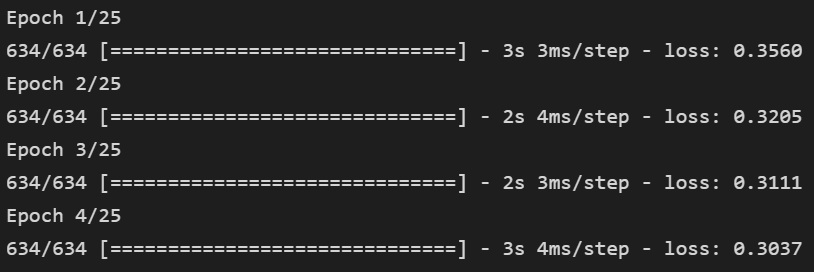
Σε αυτό το κομμάτι της εργασίας χρησιμοποιήσαμε την βιβλιοθήκη *LinearRegression* της Python. Αυτή η βιβλιοθήκη **εκτελούσε αυτόματα τη διαδικασία εύρεσης ευθείας γραμμής,** η οποία απέχει την μικρότερη δυνατή απόσταση από τα σημεία των δεδομένων εκπαίδευσης (δηλαδή τα σφάλματα ελαχιστοποιούνται). **Η ρουτίνα αυτή εκτελείται 10 φορές συνολικά,** όπως μας υποδεικνύεται από την μέθοδο της 10-πλής διεπικύρωσης (10 fold cross validation). Αναλυτικότερα, σε κάθε επανάληψη δημιουργούσαμε ένα καινούριο μοντέλο, το οποίο και εκπαιδεύαμε πάνω στα δεδομένα εκπαίδευσης, και ύστερα ελέγχαμε τις προβλέψεις μας πάνω στα δεδομένα ελέγχου. Τα τελικά σφάλματα που προέκυψαν δεν ήταν ιδιαίτερα ικανοποιητικά, παρ’ όλα αυτά ήταν τα ελάχιστα δυνατά.



Εικόνα 13: Το αποτέλεσμα του Ελάχιστου Τετραγωνικού Σφάλματος.

## Μέρος 3ο: Πολυστρωματικό Νευρωνικό Δίκτυο

Σε αυτό το κομμάτι της εργασίας χρησιμοποιήσαμε τις βιβλιοθήκες Sequential και Dense της Tensorflow/Keras. Οι συγκεκριμένες βιβλιοθήκες μας επέτρεψαν να φτιάξουμε το μοντέλο του νευρωνικού δικτύου μας. Η ρουτίνα αυτή εκτελείται 10 φορές συνολικά όπως μας υποδεικνύεται από την μέθοδο της 10-πλής διεπικύρωσης (10 fold cross validation). Μέσα στην επανάληψη κατασκευάζουμε **δύο διαφορετικά νευρωνικά δίκτυα.** Το ένα υπολογίζει το μέσο απόλυτο σφάλμα και το άλλο υπολογίζει το μέσο τετραγωνικό σφάλμα, τόσο κατά το στάδιο της εκπαίδευσης όσο και κατά το στάδιο του ελέγχου. Το πρόβλημα, που μας δημιουργήθηκε κατά τη διάρκεια του ερωτήματο,ς είναι οι μεγάλοι χρόνοι που χρειάζονταν για να τρέξουν τα μοντέλα μας. Τέλος, τα σφάλματα κατά τη διάρκεια της εκπαίδευσης φαίνονται με την παράμετρο *“verbose = 1”* στην συνάρτηση *fit()*, ενώ τα σφάλματα κατά το στάδιο του ελέγχου εμφανίζονται στο τέλος.



Εικόνα 14: Ενδεικτικό παράδειγμα του πολυστρωματικού νευρωνικού δικτύου.

# Βιβλιογραφικές Πηγές

Οι παρακάτω πηγές βοήθησαν στην συγγραφή και ανάπτυξη του κώδικα της εργασίας. Αναλυτικότερα, φαίνονται παρακάτω με σχετική αναφορά στους υπερσυνδέσμους τους:

1. [Data Preprocessing in Data Mining -A Hands On Guide - Analytics Vidhya.](https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/)
2. [(73) Outlier detection and removal using IQR | Feature engineering tutorial python #4 - YouTube.](https://www.youtube.com/watch?v=A3gClkblXK8&ab_channel=codebasics)
3. [Ελάχιστο Τετραγωνικό Σφάλμα.](https://www.geeksforgeeks.org/python-mean-squared-error/)
4. [Κανόνας της κ-πλής επικύρωσης.](https://www.statology.org/k-fold-cross-validation-in-python/)
5. [Least Squares Linear Regression — ML From Scratch (Part 1).](https://medium.com/international-school-of-ai-data-science/least-squares-linear-regression-ml-from-scratch-part-1-364fae2bd95a)
6. [How to use k-fold Validation with Keras.](https://setscholars.net/how-to-use-kfold-cross-validation-in-keras/)