Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής
Έτος: 2022 - 2023

Μάθημα:
«ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ»
Εργασία: Υπολογιστική Εργασία Μαθήματος
Εξάμηνο: 5ο

<u>Ομάδα εργασίας:</u>

Αιμιλιανός Κουρπάς-Δανάς Π20100,

Αναστάσιος Μελαχροινούδης Π20124

Ημερομηνία παράδοσης : 10.02.2023

Εισαγωγή

Η ανάπτυξη της εφαρμογής έγινε με τη γλώσσα **Python** χρησιμοποιώντας το **Visual Studio Code** και διάφορες χρήσιμες βιβλιοθήκες, που βοήθησαν πολύ.
Οι βιβλιοθήκες είναι:

1) pandas
2) matplotlib
3) sklearn
4) keras
5) seaborn
6) numpy

Οι παραπάνω βιβλιοθήκες εγκαταστάθηκαν απο το terminal με τα αναλογα commands(βιβλιογραφια).

Προεπεξεργασία Δεδομένων

Αρχικά θα πρέπει να γίνει η φόρτωση των δεδομένων.Μολις γινει η φόρτωση ελέγχουμε ότι όλα λειτουργούν με τη συνάρτηση head( ).

```
dataset = pd.read_csv('housing.csv') # Load the database
dataset.head() #with this function i can see the first 5 records of housing csv (rows)
```

	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

Έπειτα θα διαχωρισουμε το χαρακτηριστικό που θελουμε να προσεγγίσουμε το οποιο ειναι το ‘median_house_value’.
Άρα στο X έχουμε τα παντα πέρα από το ‘median_house_value’
και στο z εχουμε μόνο αυτό.

```
X = dataset[['longitude','latitude','housing_median_age',
             'total_rooms','total_bedrooms','population',
             'households','median_income','ocean_proximity']] #every attribute except the target attribute
z = dataset[['median_house_value']] #only the target attribute
```

1) Αριθμητικά - κατηγορικά χαρακτηριστικά

Χρησιμοποιώντας την μέθοδο info( ) μπορούμε να δούμε τον τύπο των δεδομένων κάθε στήλης.

```
dataset.info()
```

```
<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
```

Παρατηρούμε πως το ocean proximity είναι κατηγορηματικό ενώ τα υπόλοιπα αριθμητικά. Άρα θα τα διαχωρίσουμε:

```
categorical = [col for col in X.columns if X[col].dtype=='object']
numerical = [col for col in X.columns if X[col].dtype!='object']
```

2) Scaling Δεδομένων

Η τεχνική που βρήκαμε που είναι η Min-Max και ο σκοπός μας είναι τα δεδομένα να είναι στην κλίμακα 0-1.

```
scaler = MinMaxScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X[numerical]),columns=numerical)
X_temp = X.drop(numerical, axis=1)
X = pd.concat([X_temp, X_scaled], axis=1)
X.head()
```

	ocean_proximity	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
0	NEAR BAY	0.211155	0.567481	0.784314	0.022331	0.019863	0.008941	0.020556	0.539668
1	NEAR BAY	0.212151	0.565356	0.392157	0.180503	0.171477	0.067210	0.186976	0.538027
2	NEAR BAY	0.210159	0.564293	1.000000	0.037260	0.029330	0.013818	0.028943	0.466028
3	NEAR BAY	0.209163	0.564293	1.000000	0.032352	0.036313	0.015555	0.035849	0.354699
4	NEAR BAY	0.209163	0.564293	1.000000	0.041330	0.043296	0.015752	0.042427	0.230776

Αλγοριθμική εξήγηση

scaler = MinMaxScaler(): Αυτή η γραμμή δημιουργεί μια αντικείμενο της κλάσης MinMaxScaler από την sklearn.preprocessing. Η κλάση MinMaxScaler χρησιμοποιείται για τον μετασχηματισμό των αριθμητικών χαρακτηριστικών ενός συνόλου δεδομένων έτσι ώστε οι τιμές για κάθε χαρακτηριστικό να βρίσκονται μεταξύ 0 και 1.

X_scaled=pd.DataFrame(scaler.fit_transform(X[numerical]),columns=numerical): Αυτή η γραμμή εφαρμόζει τη μέθοδο fit_transform στην αριθμητική στήλη numerical. Η μέθοδος fit_transform προσαρμόζει το αντικείμενο MinMaxScaler στα δεδομένα και επιστρέφει τις μετασχηματισμένες τιμές ως δισδιάστατο πίνακα NumPy. Αυτός ο πίνακας περνάει στη συνέχεια στον κατασκευαστή pd.Dataset για τη δημιουργία ενός νέου Dataset X_scaled με τις μετασχηματισμένες τιμές. Οι στήλες του νέου Dataset καθορίζονται από την παράμετρο columns, η οποία χρησιμοποιεί την αριθμητική μεταβλητή για να καθορίσει τα ονόματα των στηλών.

X_temp = X.drop(numerical, axis=1): Αυτή η γραμμή απορρίπτει τις αριθμητικές στήλες από το αρχικό X Dataset και αποθηκεύει το αποτέλεσμα σε ένα νέο Dataset X_temp. Η μέθοδος drop χρησιμοποιείται για την αφαίρεση των στηλών που καθορίζονται από την αριθμητική μεταβλητή και η παράμετρος axis τίθεται σε 1 ώστε η drop να εκτελεστεί σε στήλες και όχι σε γραμμές.

X = pd.concat([X_temp, X_scaled], axis=1): Αυτή η γραμμή συνενώνει τα δύο Dataset X_temp και X_scaled κατά μήκος των στηλών (axis=1) για να δημιουργήσει ένα νέο Dataset X. Το αποτέλεσμα είναι ένα Dataset που περιέχει όλες τις αρχικές στήλες του X εκτός από τις αριθμητικές στήλες, οι οποίες έχουν αντικατασταθεί από τις "κλιμακωτές" εκδοχές τους.

X.head(): Αυτή η γραμμή εκτυπώνει τις 5 πρώτες γραμμές του Dataset

Εκτελούμε παρόμοια διαδικασία και για το z.

```
#do the same for the z
z = pd.DataFrame(scaler.fit_transform(z),columns=z.columns)
z.head()
```

	median_house_value
0	0.902266
1	0.708247
2	0.695051
3	0.672783
4	0.674638

Στην συνέχεια θα χρησιμοποιήσουμε την μέθοδο describe για να δούμε πως λειτούργησε η κλιμάκωση.

```
X.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000
mean	0.476125	0.328572	0.541951	0.066986	0.083313	0.039869	0.081983	0.232464
std	0.199555	0.226988	0.246776	0.055486	0.065392	0.031740	0.062873	0.131020
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.253984	0.147715	0.333333	0.036771	0.045779	0.021974	0.045881	0.142308
50%	0.583665	0.182784	0.549020	0.054046	0.067349	0.032596	0.067094	0.209301
75%	0.631474	0.549416	0.705882	0.080014	0.100248	0.048264	0.099326	0.292641
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
z.describe()
```

	median_house_value
count	20640.000000
mean	0.395579
std	0.237928
min	0.000000
25%	0.215671
50%	0.339588
75%	0.514897
max	1.000000

Οι μέγιστες τιμές είναι 1 και οι ελάχιστες 0. Οπότε λειτούργησαν επιτυχώς.

3) ONE - HOT encoding vector των κατηγορηματικών χαρακτηριστικών

```
#we use one-hot encoding to face the categorical attribute 'ocean_proximity'
oc_prox = X['ocean_proximity'].unique()
encoder = OneHotEncoder(handle_unknown='ignore',sparse=False) #sparse=False?
X_enc = pd.DataFrame(encoder.fit_transform(X[categorical]),columns=oc_prox)
X_temp = X.drop(categorical,axis=1)
X = pd.concat([X_temp,X_enc],axis=1)
X.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	NEAR BAY	<1H OCEAN	INLAND	NEAR OCEAN	ISLAND
0	0.211155	0.567481	0.784314	0.022331	0.019863	0.008941	0.020556	0.539668	0.0	0.0	0.0	1.0	0.0
1	0.212151	0.565356	0.392157	0.180503	0.171477	0.067210	0.186976	0.538027	0.0	0.0	0.0	1.0	0.0
2	0.210159	0.564293	1.000000	0.037260	0.029330	0.013818	0.028943	0.466028	0.0	0.0	0.0	1.0	0.0
3	0.209163	0.564293	1.000000	0.032352	0.036313	0.015555	0.035849	0.354699	0.0	0.0	0.0	1.0	0.0
4	0.209163	0.564293	1.000000	0.041330	0.043296	0.015752	0.042427	0.230776	0.0	0.0	0.0	1.0	0.0

Αλγοριθμική Περιγραφή

oc_prox = X['ocean_proximity'].unique(): Αυτή η γραμμή εξάγει τις μοναδικές τιμές της στήλης 'ocean_proximity' στο dataset X και τις αποθηκεύει στη μεταβλητή oc_prox.

encoder = OneHotEncoder(handle_unknown='ignore',sparse=False): Αυτή η γραμμή δημιουργεί ένα αντικείμενο της κλάσης OneHotEncoder από την βιβλιοθήκη sklearn.preprocessing. Η κλάση OneHotEncoder χρησιμοποιείται για την εκτέλεση κωδικοποίησης one-hot σε κατηγορικά δεδομένα, μια τεχνική που δημιουργεί μια νέα δυαδική στήλη για κάθε μοναδική τιμή της κατηγορικής μεταβλητής. Η παράμετρος handle_unknown ορίζεται σε 'ignore' για να διασφαλιστεί ότι ο κωδικοποιητής αγνοεί τυχόν άγνωστες κατηγορικές τιμές που ενδέχεται να εμφανιστούν στα δεδομένα. Η παράμετρος sparse τίθεται σε False για να επιστρέψει έναν πίνακα NumPy, αντί για έναν πίνακα, γεγονός που διευκολύνει τον χειρισμό των δεδομένων σε μεταγενέστερα βήματα.

X_enc=pd.DataFrame(encoder.fit_transform(X[categorical]),columns=oc_prox): Αυτή η γραμμή εφαρμόζει τη μέθοδο fit_transform στις κατηγορικές στήλες του X DataFrame, που καθορίζονται από την κατηγορική μεταβλητή. Η μέθοδος fit_transform προσαρμόζει το αντικείμενο κωδικοποιητή στα δεδομένα και επιστρέφει τις μετασχηματισμένες τιμές ως δισδιάστατο πίνακα NumPy. Αυτός ο πίνακας περνάει στη συνέχεια στον κατασκευαστή pd.DataFrame για να δημιουργήσει ένα νέο DataFrame X_enc με τις μετασχηματισμένες τιμές. Οι στήλες του νέου DataFrame καθορίζονται από την παράμετρο columns, η οποία χρησιμοποιεί τη μεταβλητή oc_prox για να καθορίσει τα ονόματα των στηλών.

X_temp = X.drop(categorical, axis=1): Αυτή η γραμμή απορρίπτει τις κατηγορικές στήλες από το αρχικό X DataFrame και αποθηκεύει το

αποτέλεσμα σε ένα νέο DataFrame X_temp. Η μέθοδος drop χρησιμοποιείται για την αφαίρεση των στηλών που καθορίζονται από τη μεταβλητή categorical και η παράμετρος axis ορίζεται σε 1 ώστε η drop να εκτελεστεί σε στήλες και όχι σε γραμμές.

X = pd.concat([X_temp, X_enc], axis=1): Αυτή η γραμμή συνενώνει τα δύο DataFrame X_temp και X_enc κατά μήκος των στηλών (axis=1) για να δημιουργήσει ένα νέο DataFrame X. Το αποτέλεσμα είναι ένα DataFrame που περιέχει όλες τις αρχικές στήλες του X εκτός από τις κατηγορικές στήλες, οι οποίες έχουν αντικατασταθεί από τις κωδικοποιημένες με τις hot εκδοχές τους.

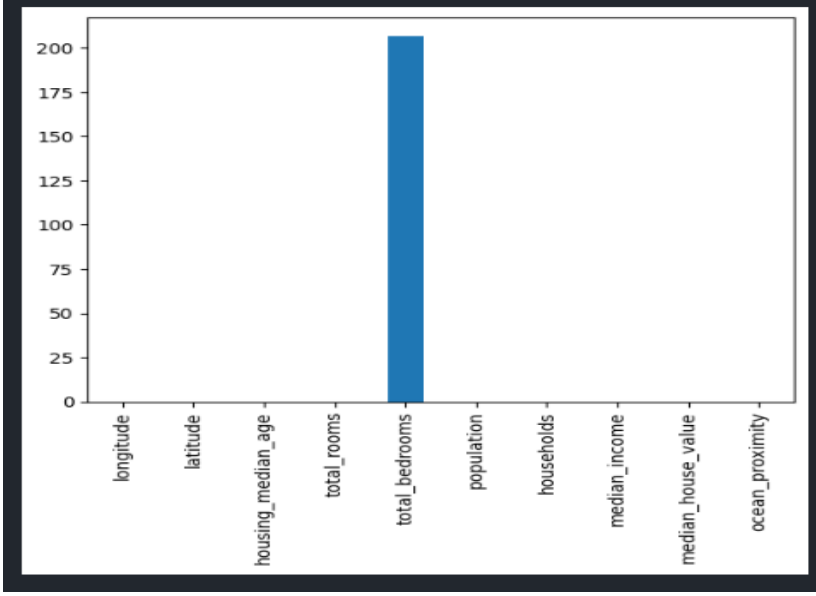
4) Ελλιπείς τιμές

Χρησιμοποιούμε την μέθοδο isnull() και παρατηρούμε πως το total bedrooms έχει 207 εγγραφές.

```
lost = dataset.isnull().sum()
print(lost)
lost.plot.bar()
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64

<AxesSubplot: >
```



Άρα με την βοηθεια της strategy median θα γεμισουμε τις ελλιπείς τιμές με τη διάμεση τιμή.

```
imputer = SimpleImputer(strategy="median")
X_imp = pd.DataFrame(imputer.fit_transform(X[numerical]),columns=numerical)
X_temp = X.drop(numerical,axis=1)
X = pd.concat([X_temp,X_imp],axis=1)
print(X.isnull().sum())
```

```
NEAR BAY            0
<1H OCEAN           0
INLAND              0
NEAR OCEAN          0
ISLAND              0
longitude           0
latitude            0
housing_median_age  0
total_rooms         0
total_bedrooms      0
population          0
households          0
median_income       0
dtype: int64
```

Αλγοριθμικη Περιγραφη

Ο παραπάνω κώδικας χρησιμοποιεί την κλάση SimpleImputer από τη βιβλιοθήκη sklearn για να διαχειριστεί τις ελλείπουσες τιμές. Ο imputer αρχικοποιείται με το όρισμα strategy="median", το οποίο σημαίνει ότι θα αντικαταστήσει τις ελλείπουσες τιμές στις αριθμητικές στήλες του συνόλου δεδομένων με τη διάμεσο των μη ελλιπών τιμών.

X_imp=pd.DataFrame(imputer.fit_transform(X[numerical]),columns=numerical) χρησιμοποιεί τη μέθοδο fit_transform για να αντικαταστήσει τις ελλείπουσες τιμές στις αριθμητικές στήλες του συνόλου δεδομένων και μετατρέπει το αποτέλεσμα σε ένα DataFrame της pandas. Οι αρχικές αριθμητικές στήλες του συνόλου δεδομένων καθορίζονται από τη μεταβλητή numerical.

Στη συνέχεια, ο κώδικας χρησιμοποιεί τη μέθοδο drop για να αφαιρέσει τις αρχικές αριθμητικές στήλες από το σύνολο δεδομένων και αποθηκεύει το αποτέλεσμα στην X_temp. Τέλος, ο κώδικας χρησιμοποιεί τη μέθοδο concat για να συνδέσει το πλαίσιο δεδομένων X_temp και το πλαίσιο δεδομένων X_imp, το οποίο περιέχει τις αριθμητικές στήλες με υπολογισμένες ελλείπουσες τιμές.

Η τελευταία γραμμή κώδικα print(X.isnull().sum()) εκτυπώνει τον αριθμό των ελλιπών τιμών σε κάθε στήλη του συνόλου δεδομένων που προκύπτει. Εάν δεν υπάρχουν άλλες ελλιπείς τιμές, το άθροισμα των ελλιπών τιμών σε κάθε στήλη θα πρέπει να είναι μηδέν.

Οπτικοποίηση Δεδομένων

1) **Ιστογράμματα συχνοτήτων χαρακτηριστικών**

Χρησιμοποιήθηκε παντού η ίδια εντολή.
Έστω ότι έχουμε την εντολή:

```
sns.histplot(data[data.columns[0]],bins=50,kde=True,lw=2)
```

Σημαίνει ότι:

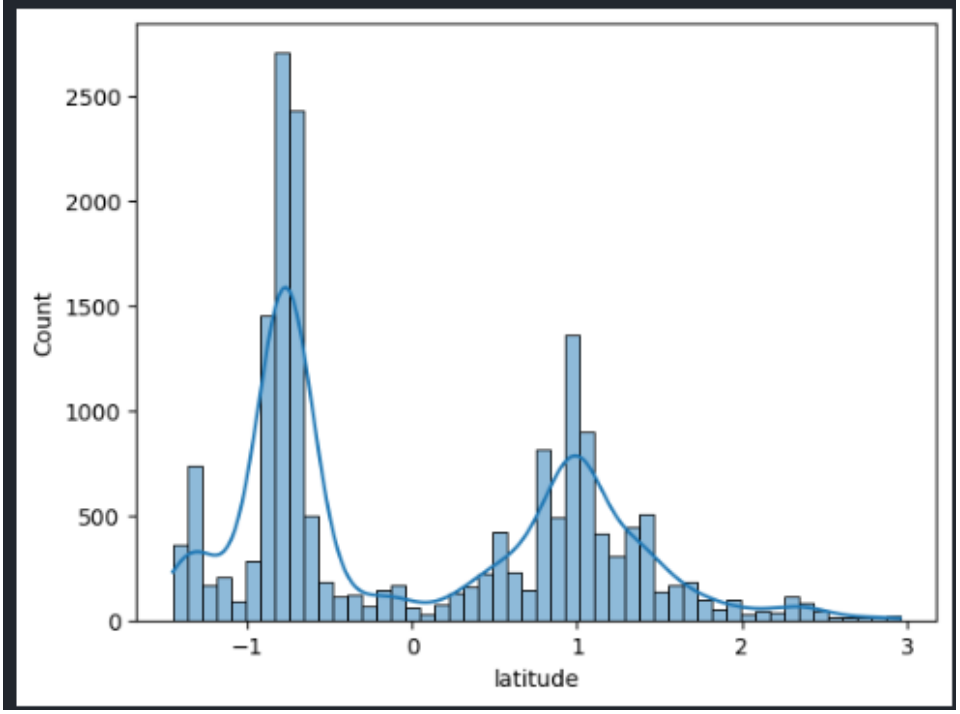
1) Χρησιμοποιούμε την βιβλιοθηκη Seaborn
2) Το data[data.columns[0]] χρησιμοποιείται για την εξαγωγή της πρώτης στήλης του πλαισίου δεδομένων και τη σχεδίασή της ως ιστόγραμμα.
3) Το bins=50 καθορίζει τον αριθμό των bins (δηλαδή των διαστημάτων) στα οποία θα διαιρεθούν τα δεδομένα για το ιστόγραμμα.
4) Το kde=True προσθέτει μια καμπύλη εκτίμησης (KDE) στη γραφική παράσταση, η οποία είναι μια εκτίμηση της υποκείμενης συνάρτησης πυκνότητας πιθανότητας των δεδομένων.
5) Το lw=2 θέτει το πλάτος της γραμμής της καμπύλης KDE σε 2.

Άρα με λιγα λογια αυτή η γραμμή κώδικα παράγει ένα ιστόγραμμα της πρώτης στήλης του πλαισίου δεδομένων, με 50 bins, μια εκτίμηση πυκνότητας πυρήνα και πλάτος γραμμής 2 για την καμπύλη KDE.

Ακολουθούν ScreenShots με τα διαγράμματα:

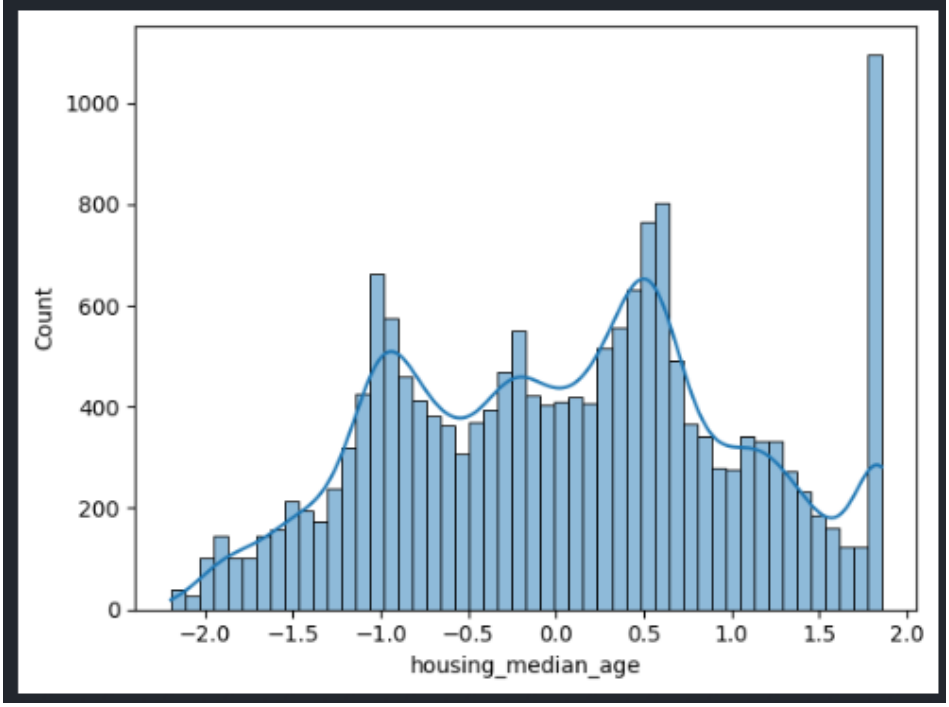
```
#latitude'
sns.histplot(data[data.columns[1]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='latitude', ylabel='Count'>
```



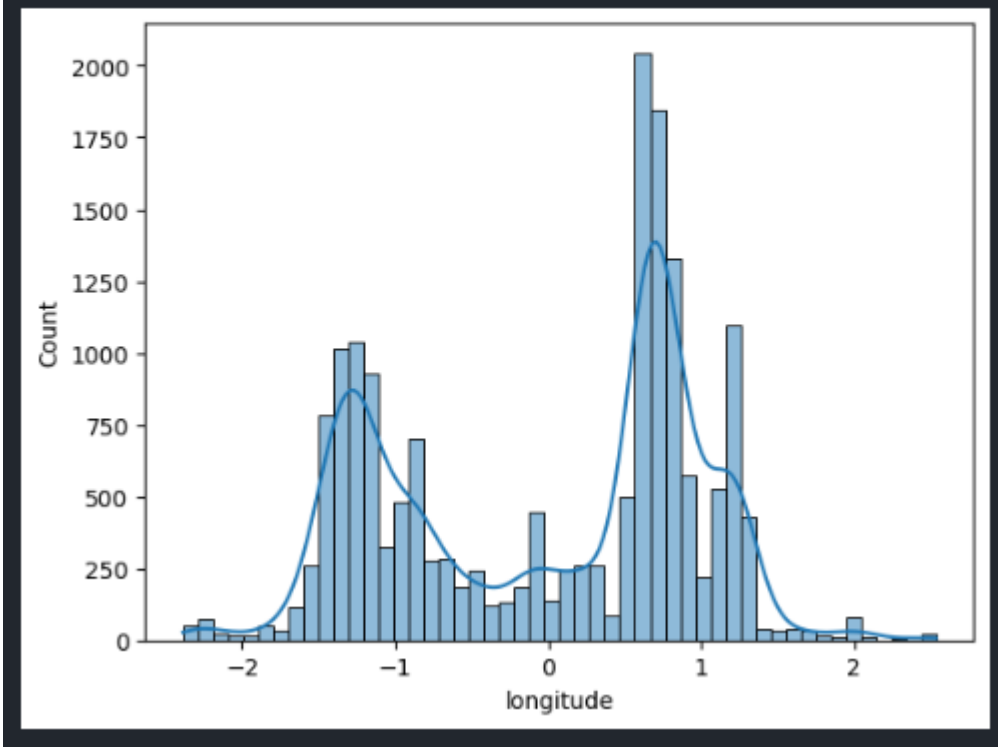

```
#'housing_median_age'
sns.histplot(data[data.columns[2]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='housing_median_age', ylabel='Count'>
```



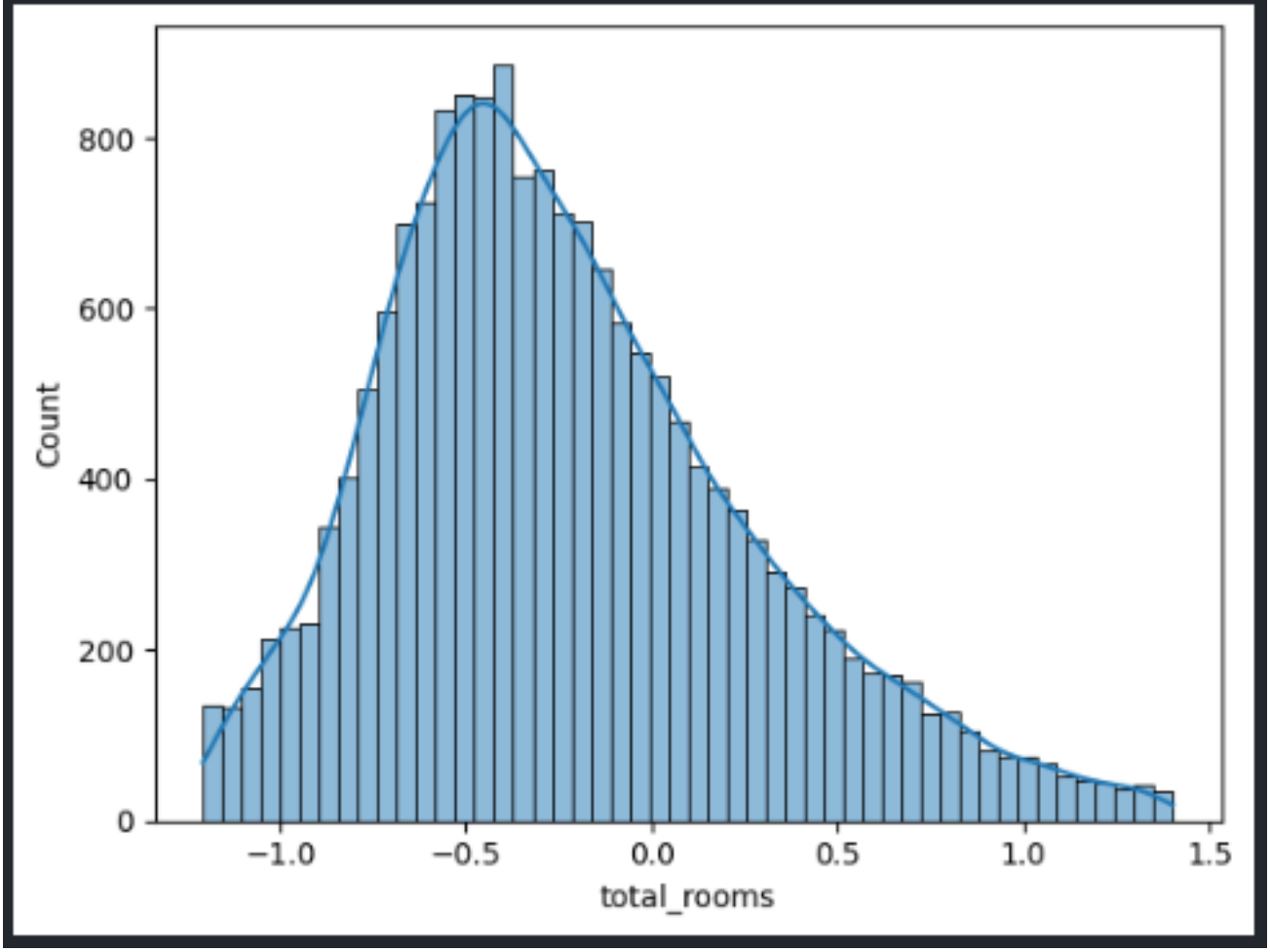
```
# longitude
sns.histplot(data[data.columns[0]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='longitude', ylabel='Count'>
```



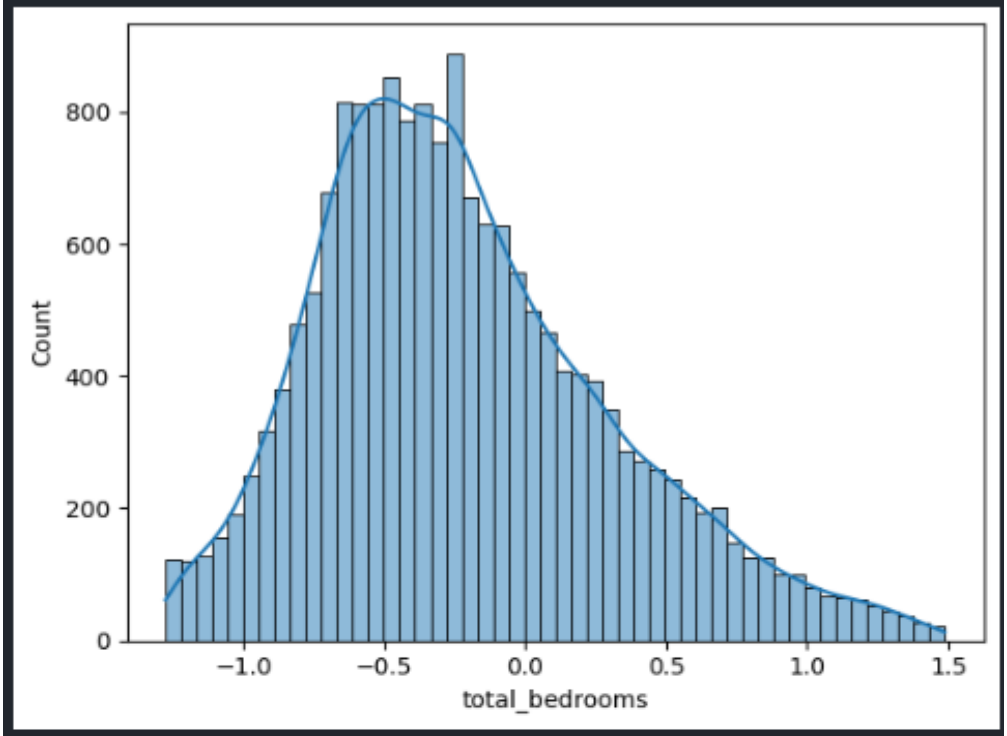
```
#'total_rooms'
sns.histplot(data[data.columns[3]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='total_rooms', ylabel='Count'>
```



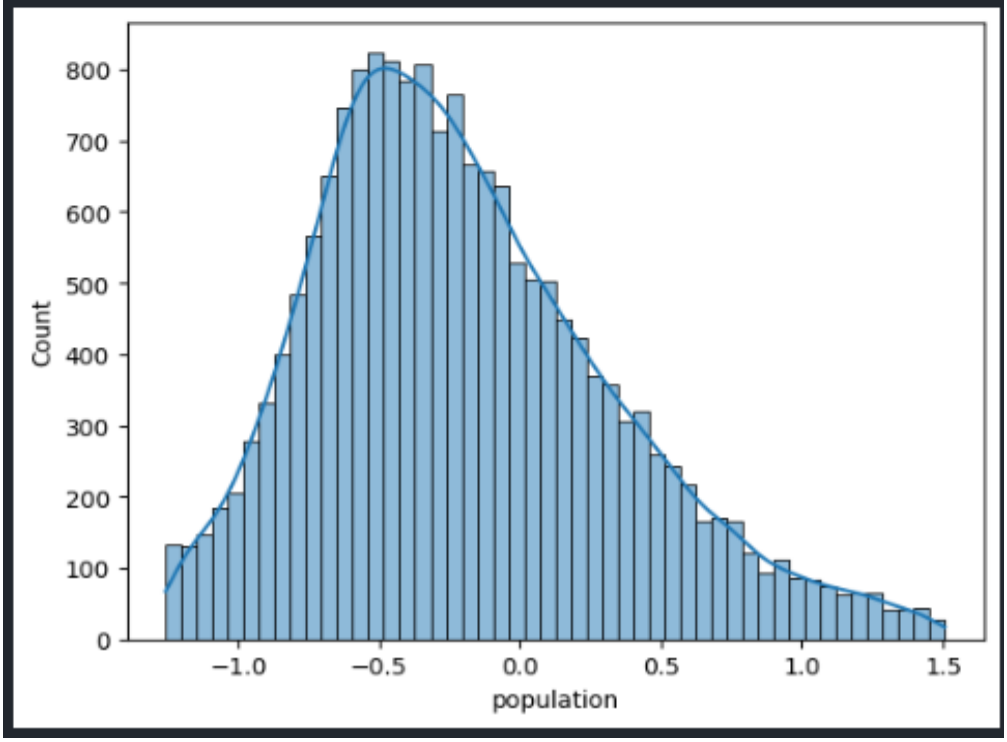
```
#'total_bedrooms'
sns.histplot(data[data.columns[4]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='total_bedrooms', ylabel='Count'>
```



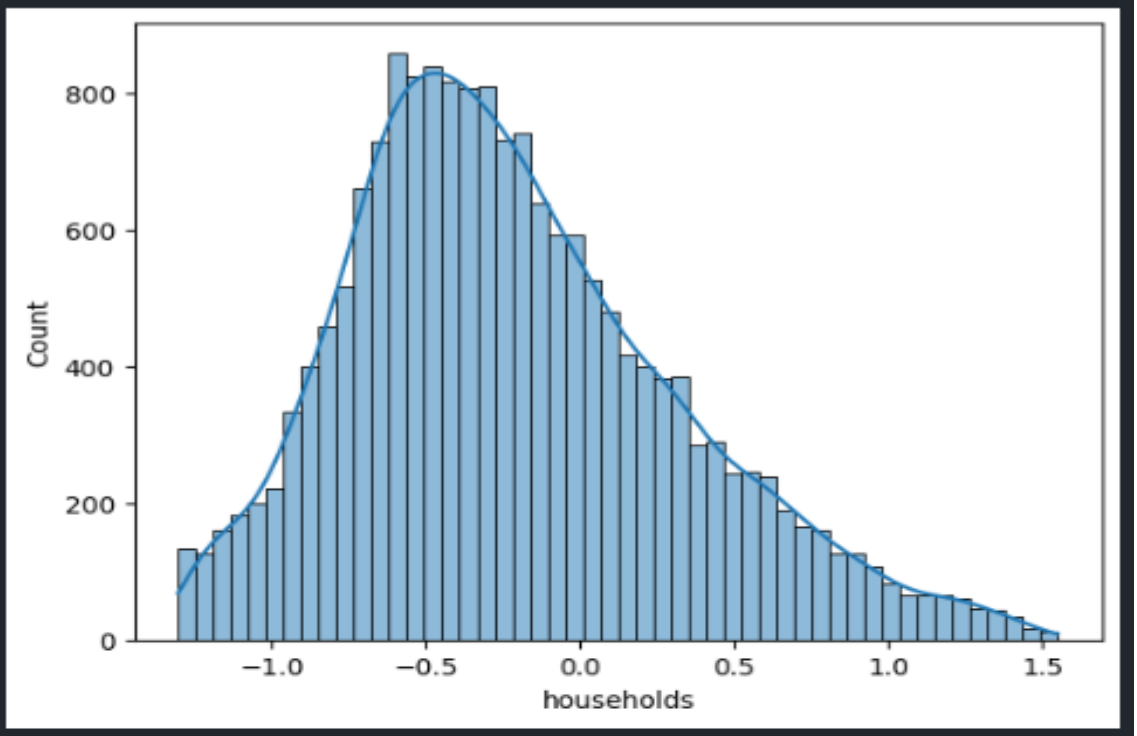

```
#'population'
sns.histplot(data[data.columns[5]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='population', ylabel='Count'>
```



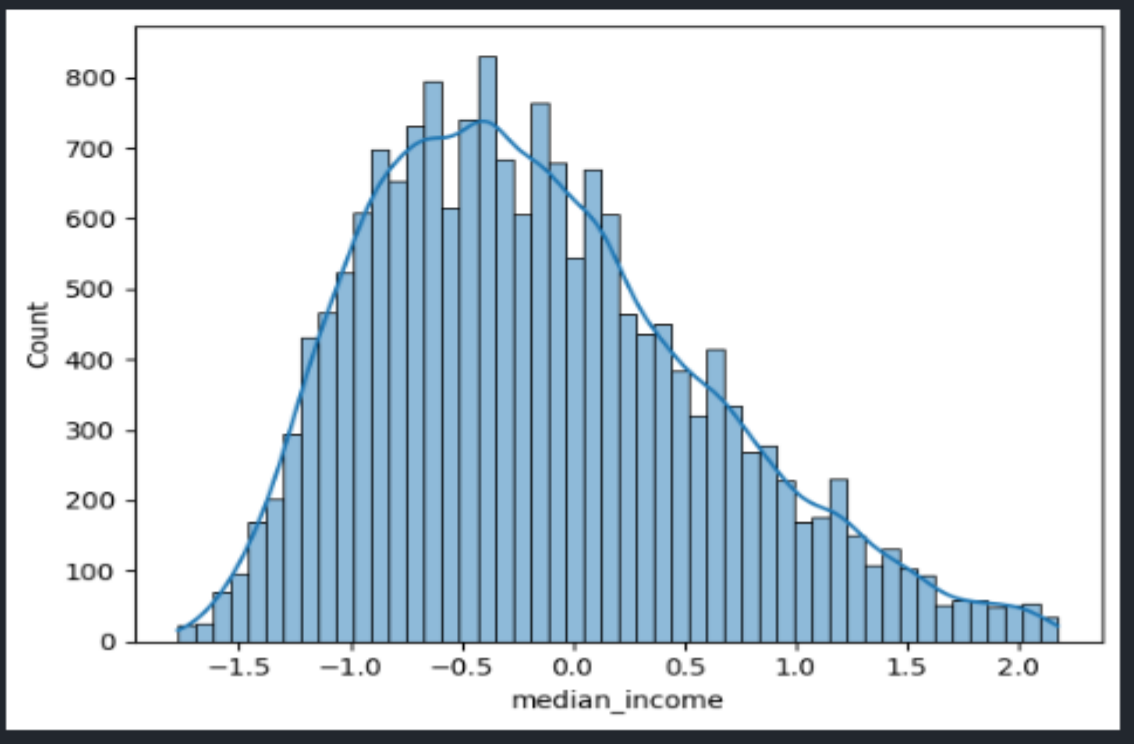
```
#'households'
sns.histplot(data[data.columns[6]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='households', ylabel='Count'>
```



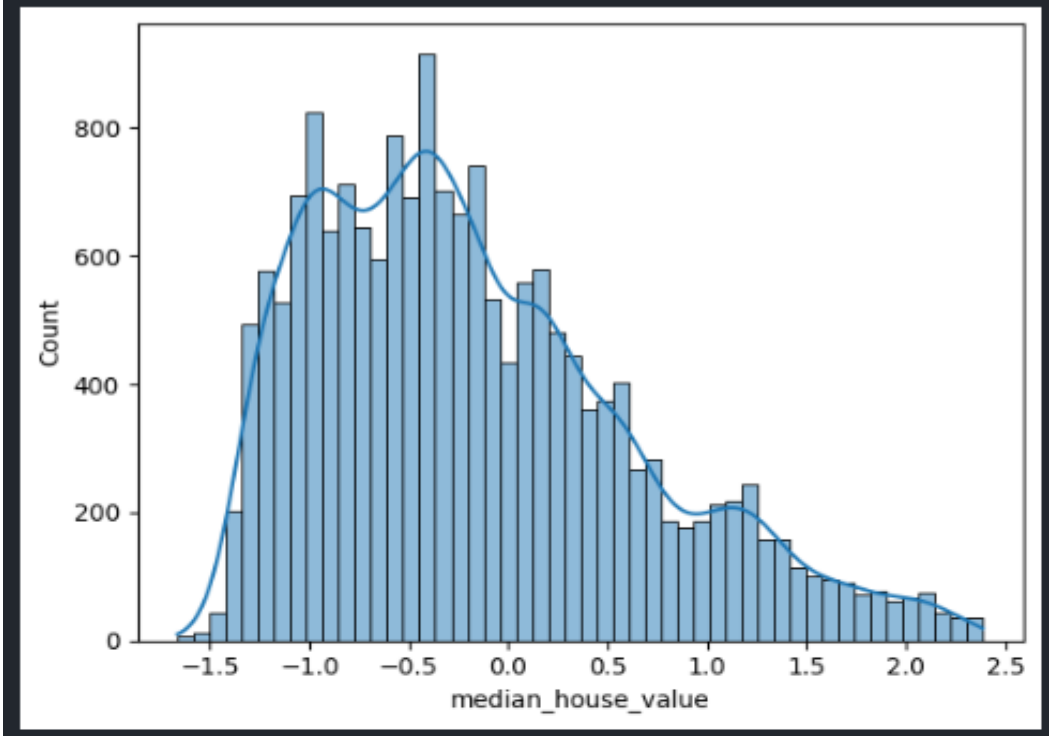

```
#'median_income'
sns.histplot(data[data.columns[7]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='median_income', ylabel='Count'>
```



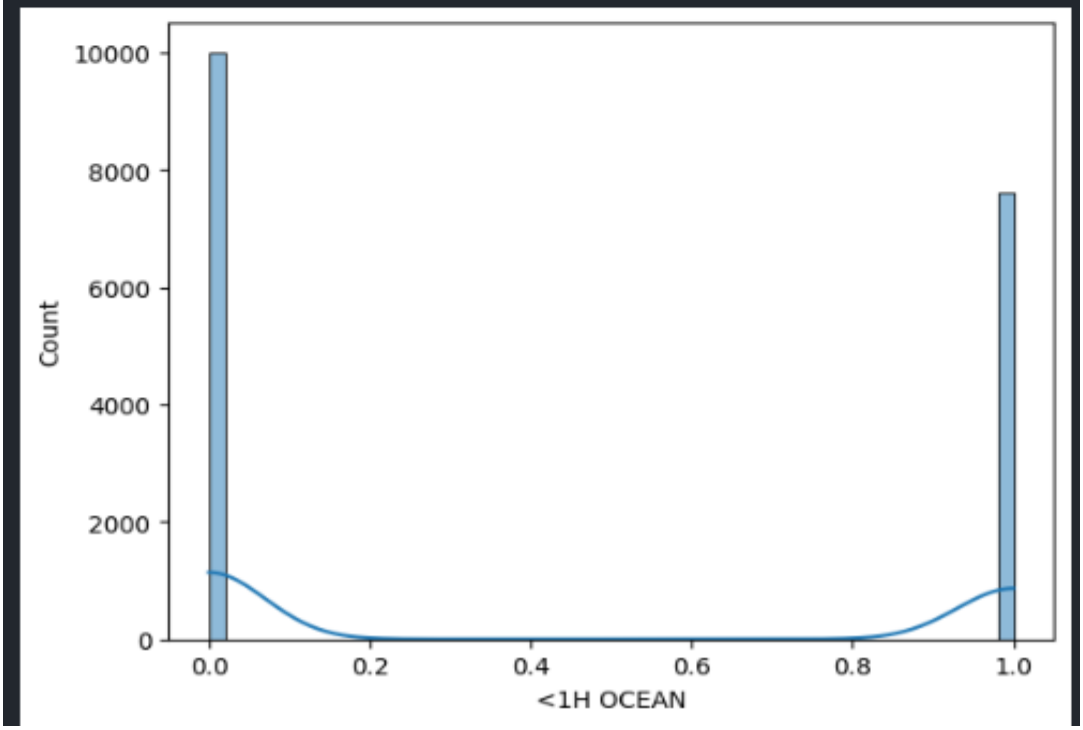
```
#'median_house_value'
sns.histplot(data[data.columns[8]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='median_house_value', ylabel='Count'>
```



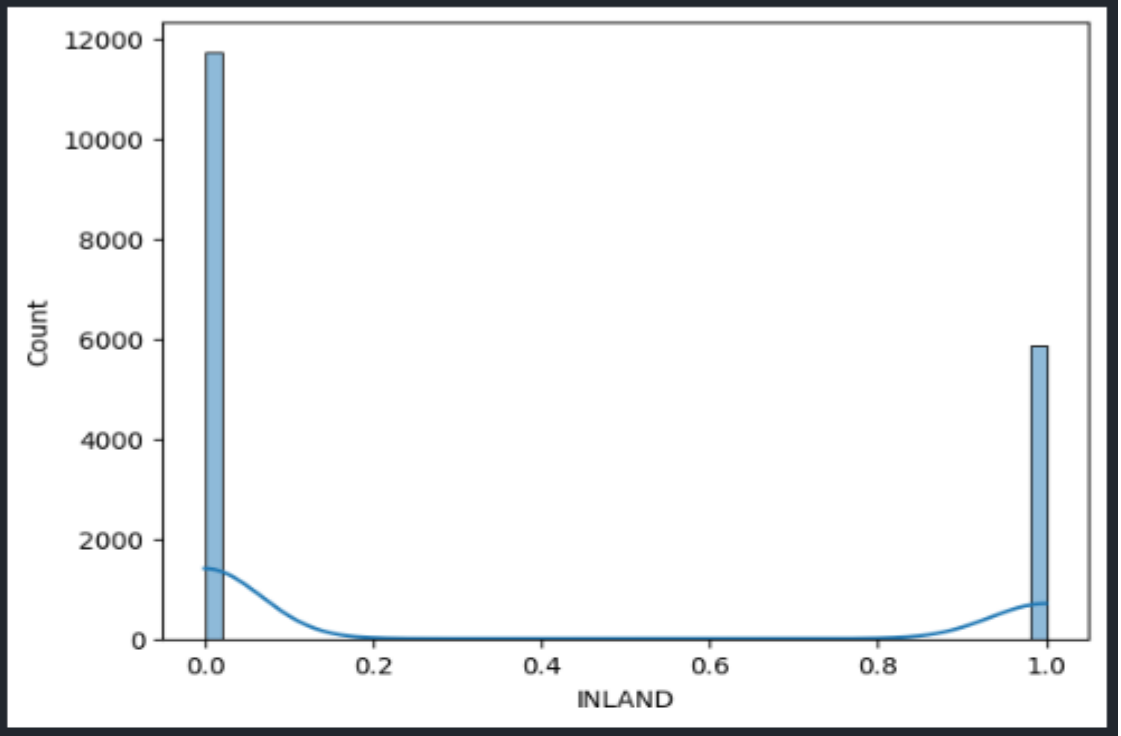

```
#'<1H OCEAN'
sns.histplot(data[data.columns[9]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='<1H OCEAN', ylabel='Count'>
```



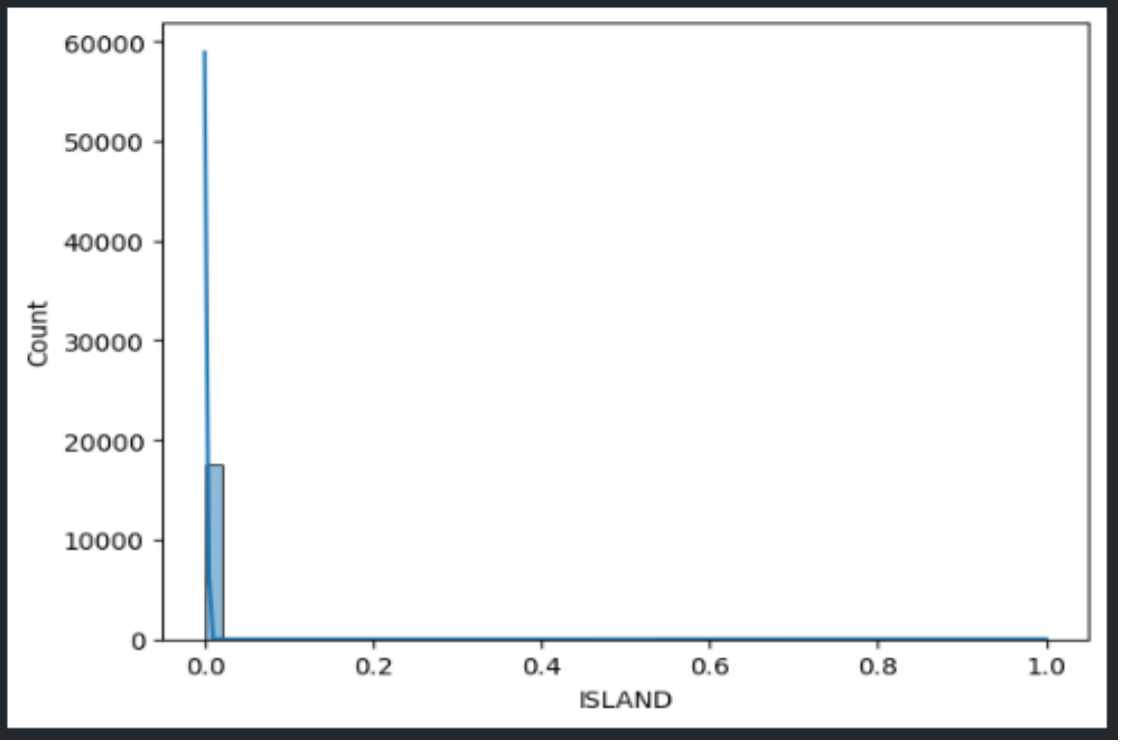
```
#'INLAND'
sns.histplot(data[data.columns[10]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='INLAND', ylabel='Count'>
```



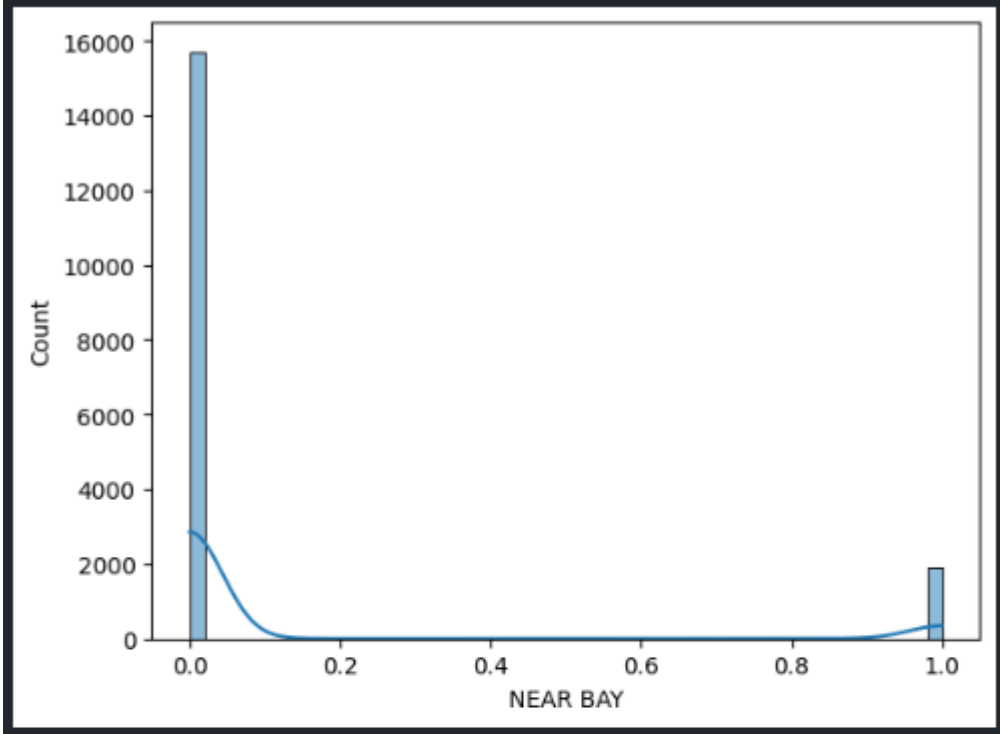

```
#'ISLAND'
sns.histplot(data[data.columns[11]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='ISLAND', ylabel='Count'>
```



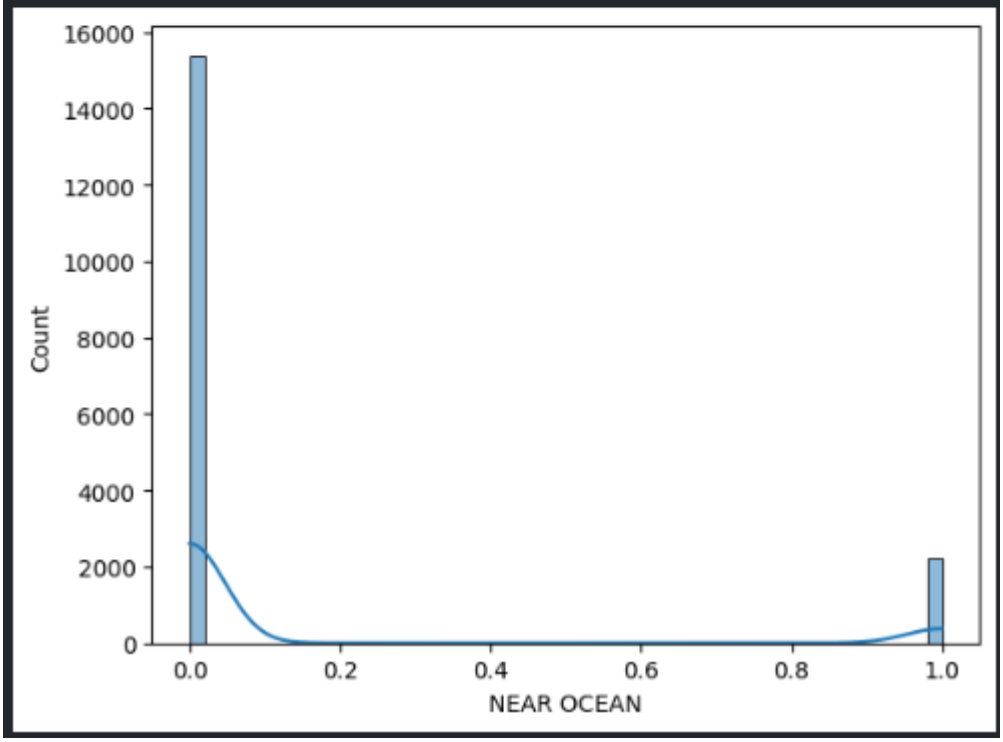
```
#'NEAR BAY'
sns.histplot(data[data.columns[12]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='NEAR BAY', ylabel='Count'>
```




```
#'NEAR OCEAN'
sns.histplot(data[data.columns[13]],bins=50,kde=True,lw=2)
```

```
<AxesSubplot: xlabel='NEAR OCEAN', ylabel='Count'>
```



2) Δισδιάστατα γραφήματα

Εδώ χρησιμοποιήσαμε δύο συναρτήσεις.

1) Πρώτη Συνάρτηση.
2) Έστω ότι έχουμε :

```
dataset.plot(kind="scatter",x="longitude",y="median_house_value")
```

Σημαίνει ότι :

1) Χρησιμοποιούμε την βιβλιοθήκη Pandas
2) Η kind="scatter" καθορίζει ότι πρέπει να δημιουργηθεί ένα διάγραμμα διασποράς.
3) Τα x και y ορίζουν τις μεταβλητές x και y, αντίστοιχα, για κάθε διάγραμμα. Για παράδειγμα, στο πρώτο διάγραμμα, οι x="longitude" και y="median_house_value" καθορίζουν ότι η μεταβλητή longitude θα πρέπει να απεικονιστεί στον άξονα x και η διάμεση τιμή κατοικίας θα πρέπει να απεικονιστεί στον άξονα y.

2) Δεύτερη Συνάρτηση.

Έστω ότι έχουμε :

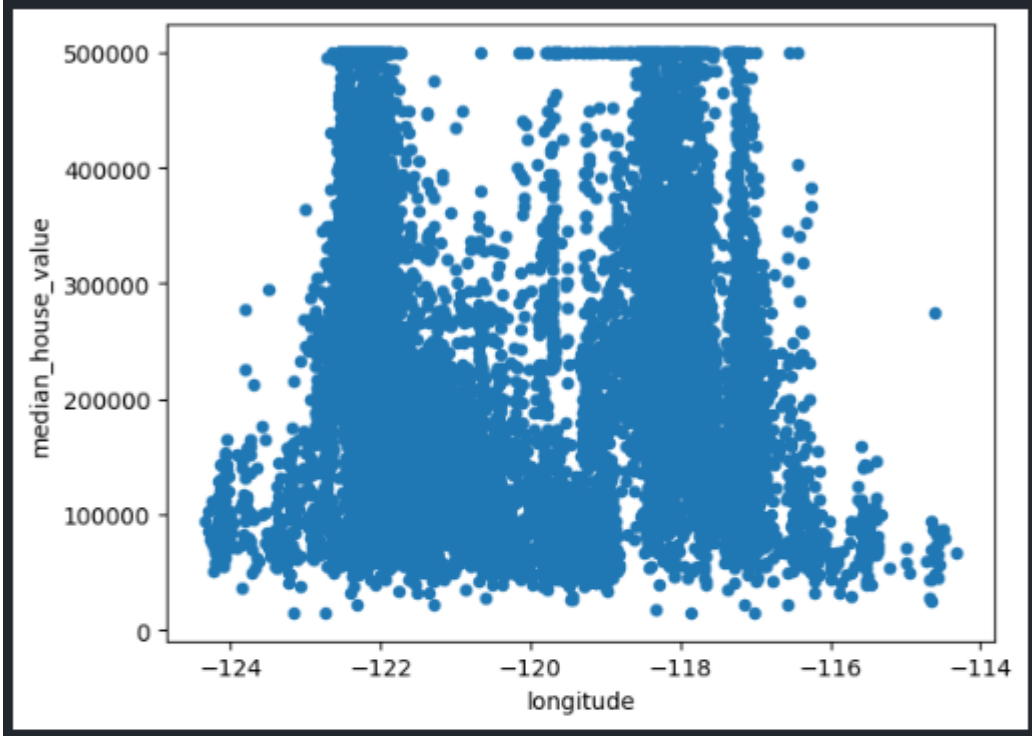
```
sns.scatterplot(x=data['median_income'],y=data['median_house_value'],hue=data[
NEAR OCEAN'])
```

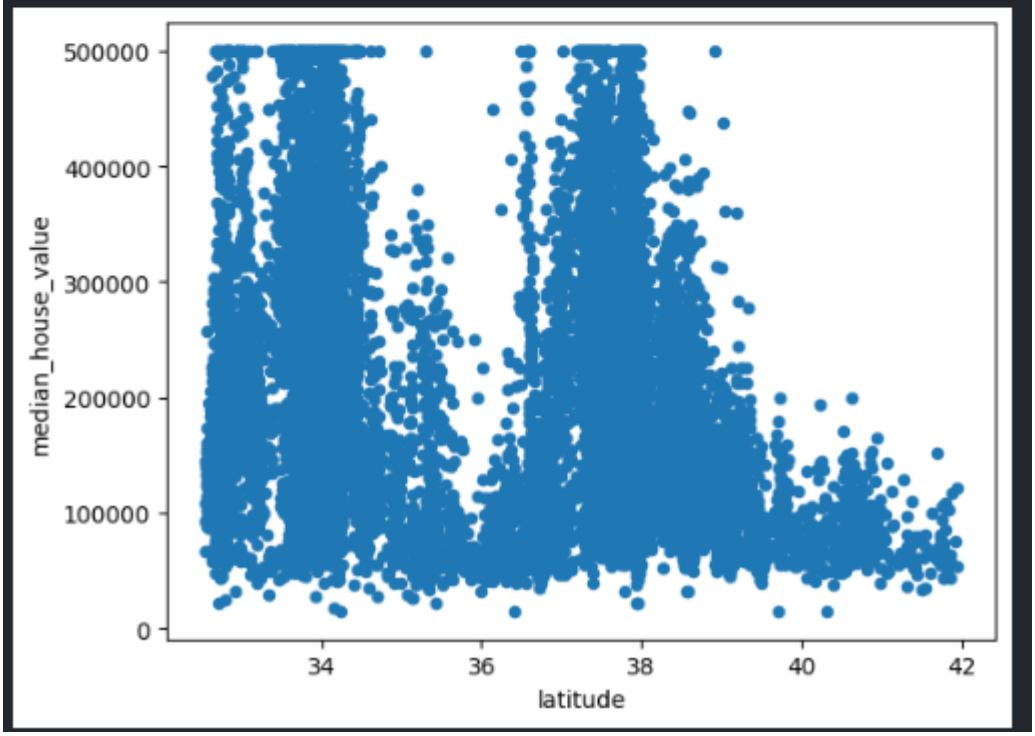
Σημαίνει ότι :

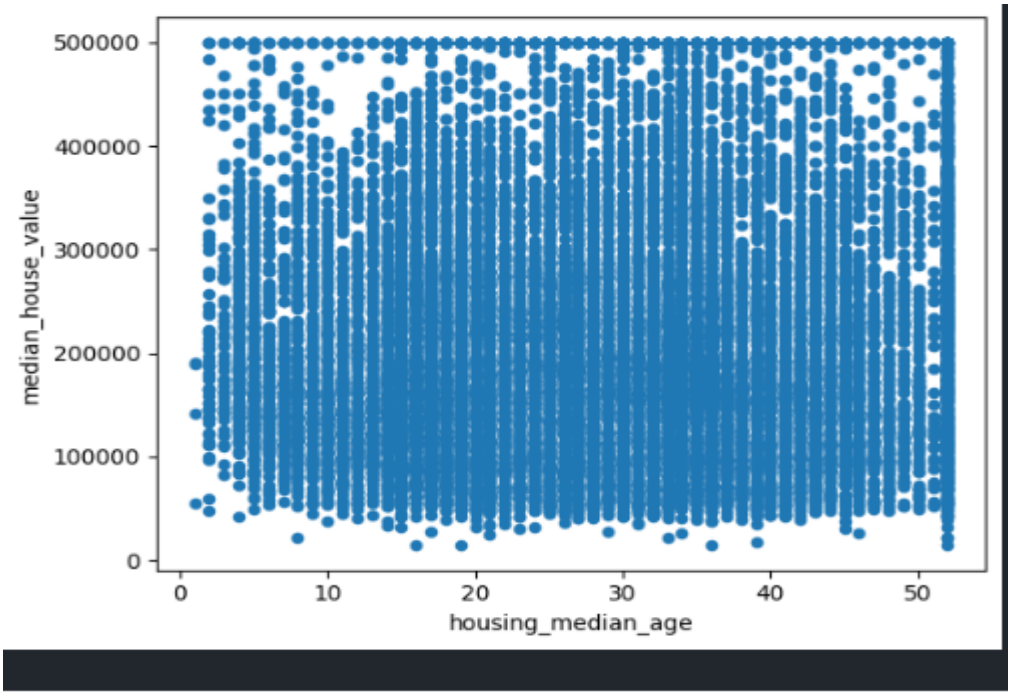
1) Χρησιμοποιούμε την βιβλιοθήκη Seaborn
2) Η x=data['median_income'] καθορίζει ότι η στήλη median_income του πλαισίου δεδομένων θα πρέπει να απεικονιστεί στον άξονα x.
3) Η y=data['median_house_value'] καθορίζει ότι η στήλη median_house_value του πλαισίου δεδομένων θα πρέπει να απεικονίζεται στον άξονα y.
4) Η hue=data['NEAR OCEAN'] χρωματίζει τα σημεία διασποράς με βάση τις τιμές της στήλης NEAR OCEAN του πλαισίου δεδομένων. Αυτό μας επιτρέπει να απεικονίσουμε πώς επηρεάζεται η σχέση μεταξύ median_income και median_house_value από το NEAR OCEAN.

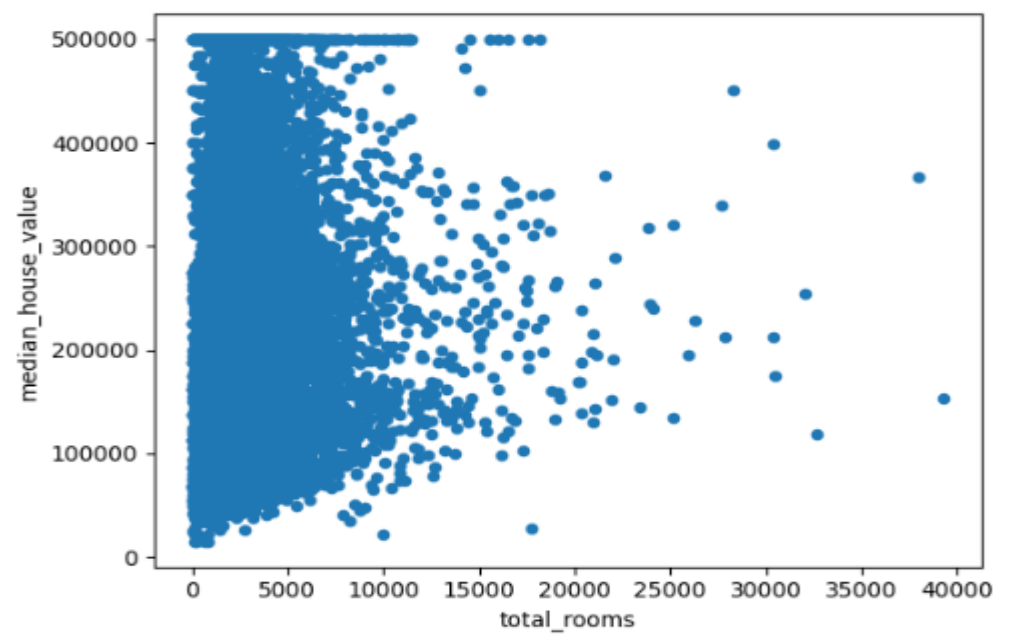
Ακολουθούν ScreenShots με τα διαγράμματα:

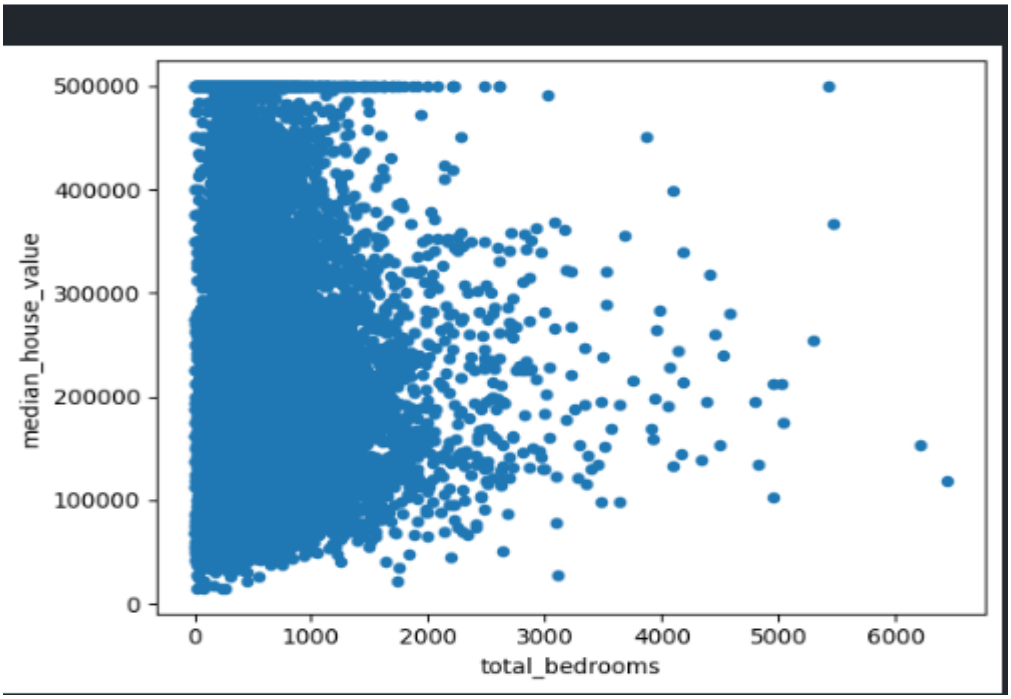
```
dataset.plot(kind="scatter",x="longitude",y="median_house_value")
dataset.plot(kind="scatter",x="latitude",y="median_house_value")
dataset.plot(kind="scatter",x="housing_median_age",y="median_house_value")
dataset.plot(kind="scatter",x="total_rooms",y="median_house_value")
dataset.plot(kind="scatter",x="total_bedrooms",y="median_house_value")
dataset.plot(kind="scatter",x="population",y="median_house_value")
dataset.plot(kind="scatter",x="households",y="median_house_value")
dataset.plot(kind="scatter",x="median_income",y="median_house_value")
plt.show()
```

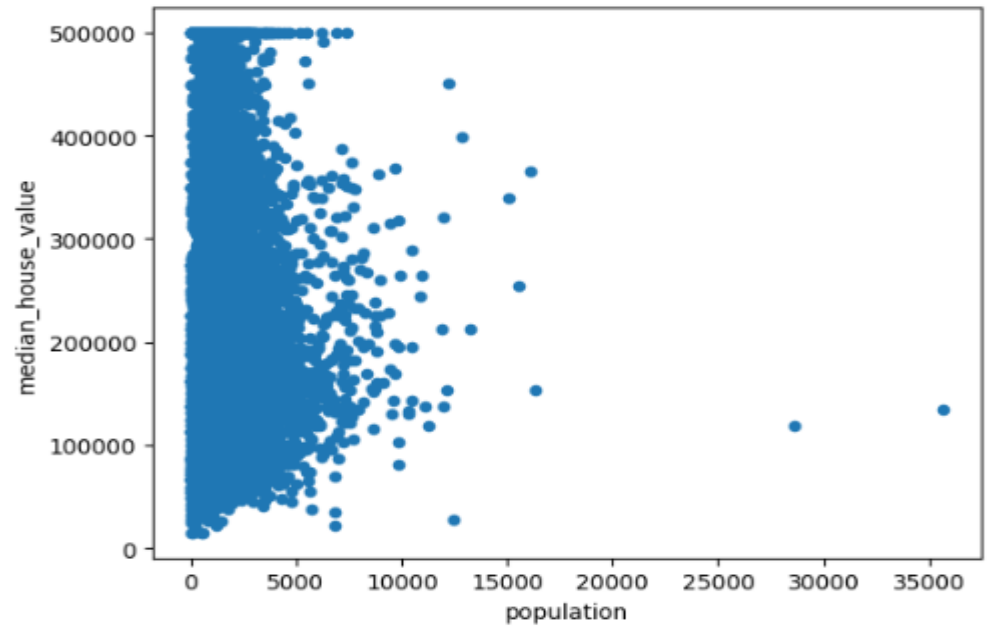


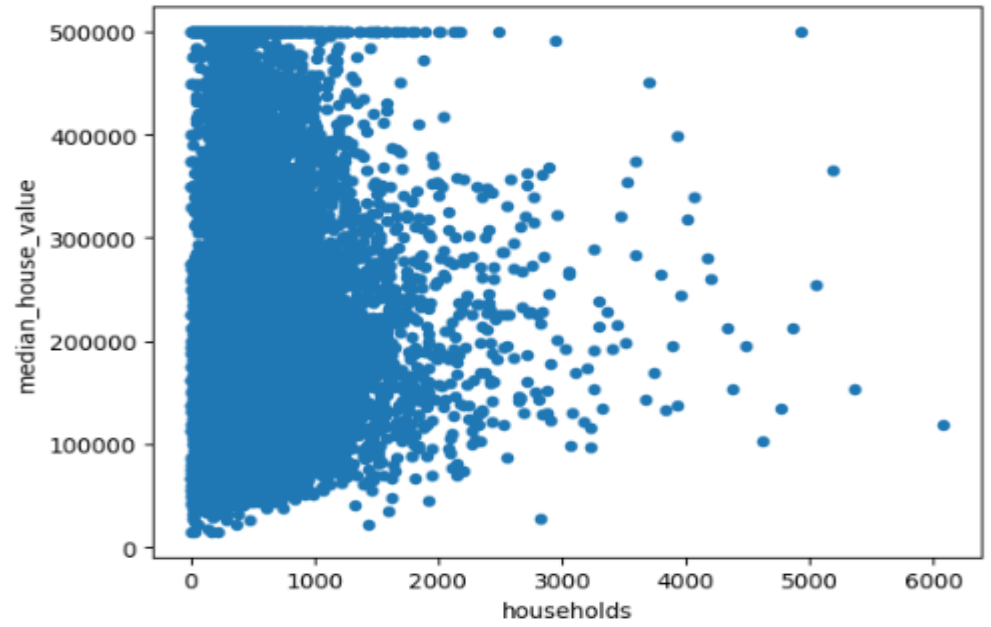


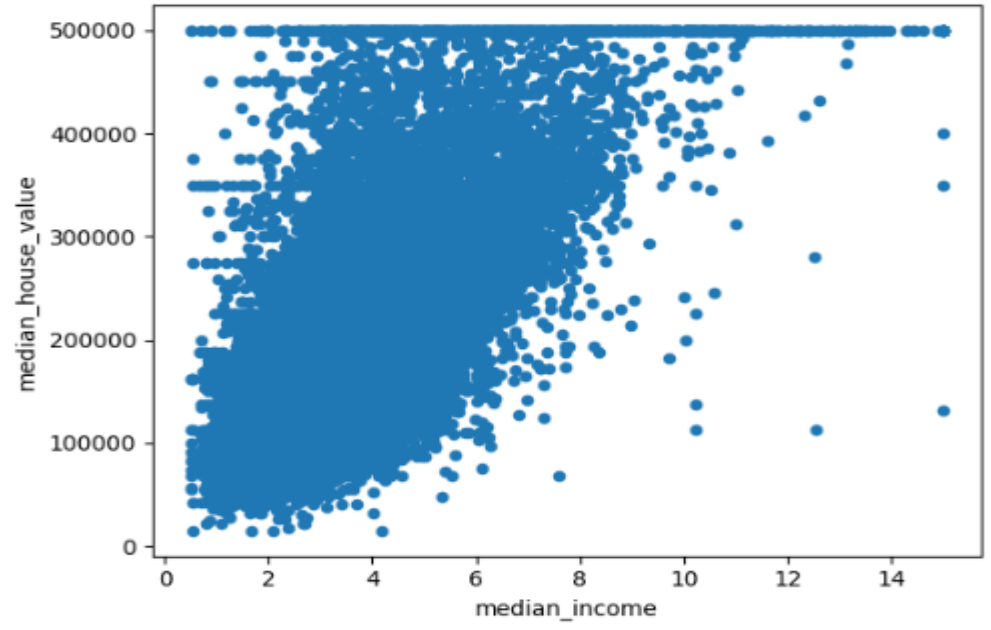

median_house_value
housing_median_age
0
100000
200000
300000
400000
500000
0
10
20
30
40
50

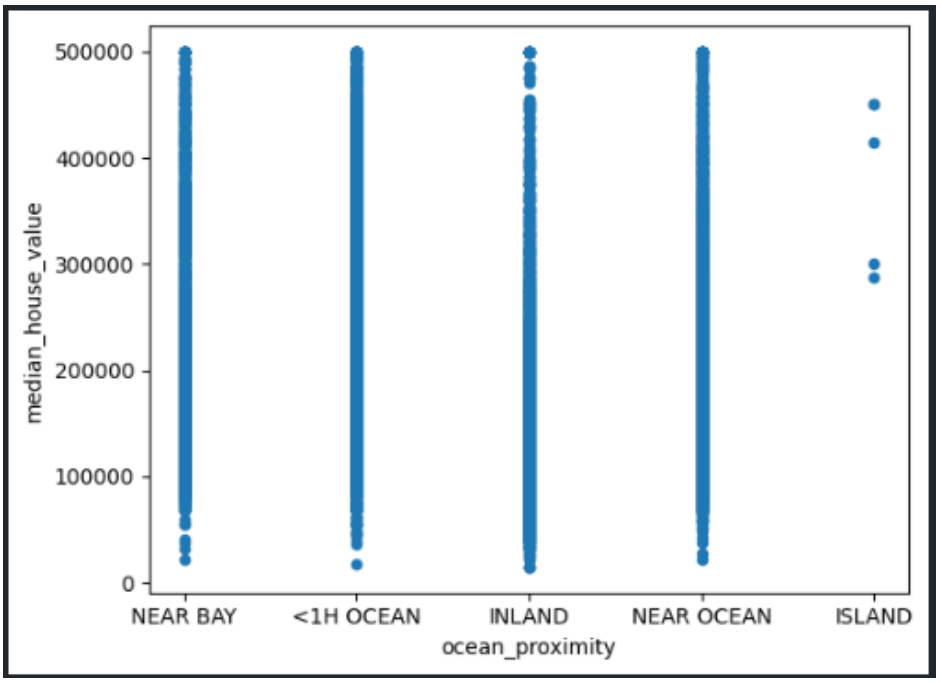

median_house_value
total_rooms
0
100000
200000
300000
400000
500000
0
5000
10000
15000
20000
25000
30000
35000
40000


median_house_value
total_bedrooms
0
100000
200000
300000
400000
500000
0
1000
2000
3000
4000
5000
6000

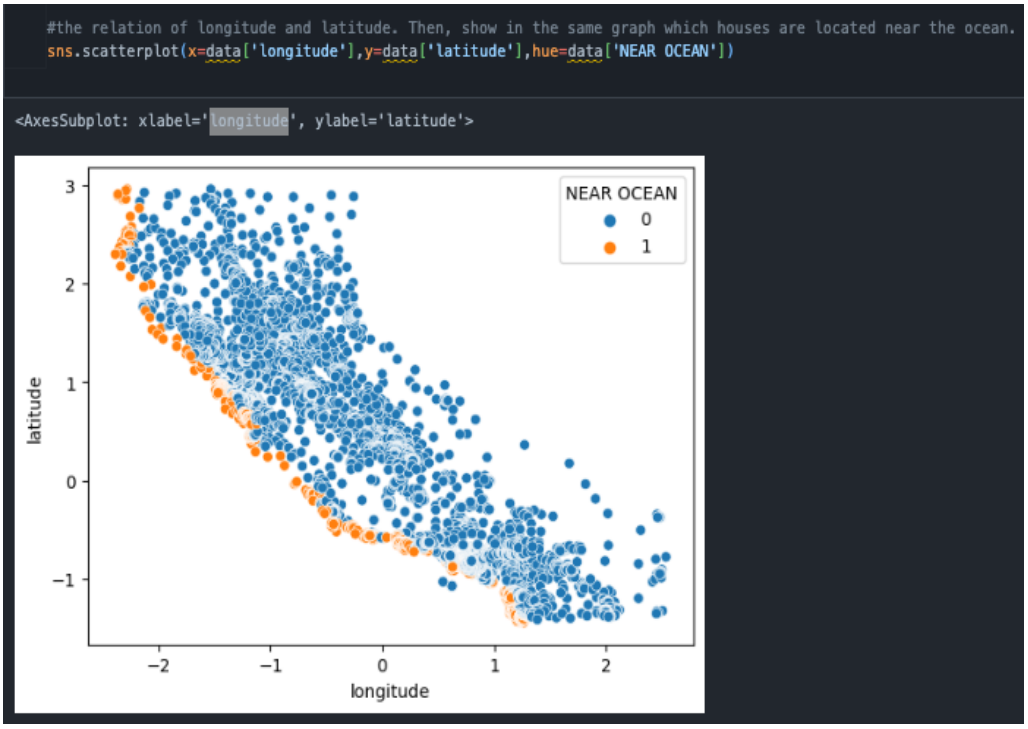

median_house_value
500000
400000
300000
200000
100000
0
0
5000
10000
15000
20000
25000
30000
35000
population


median_house_value
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
6000
households


median_house_value
500000
400000
300000
200000
100000
0
0
2
4
6
8
10
12
14
median_income



Παρατηρούμε μια γραμμική συσχέτιση ανάμεσα σε median_income & median_house_value
Επίσης σχεδιάζοντας το longitude & latitude & NEAR OCEAN παρατηρούμε ότι:

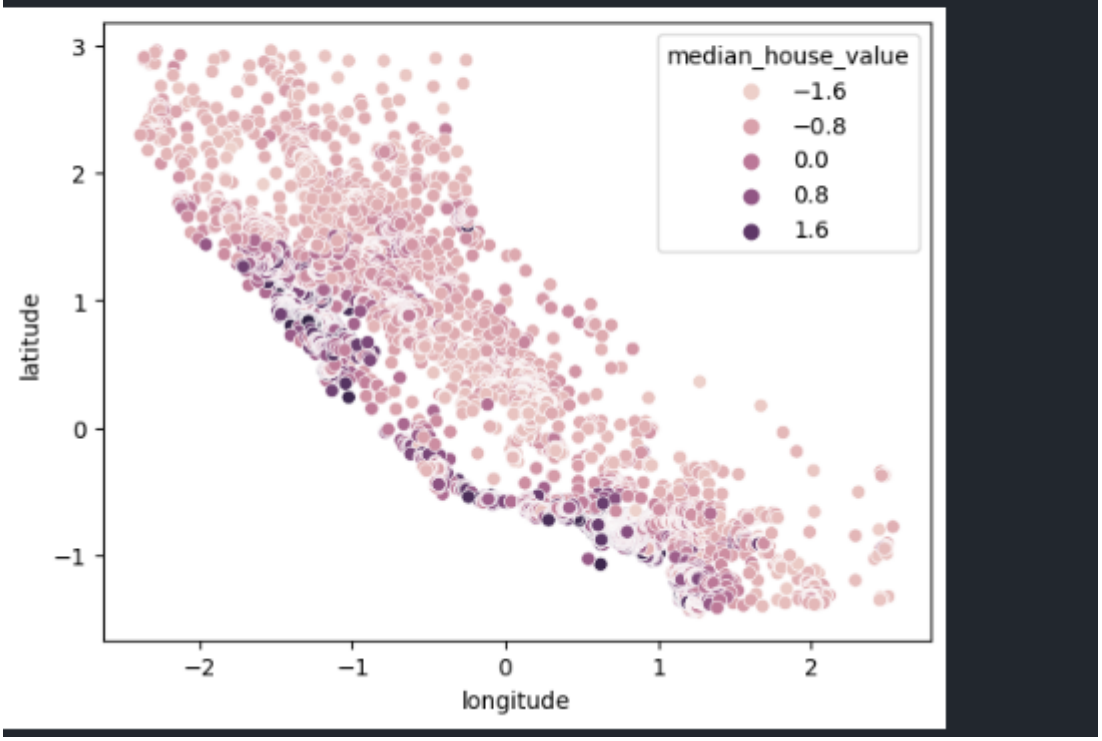


Μοιάζουν με την μορφή της πολιτείας:

Άλλο ένα παράδειγμα:

```
#houses cost along each area of Callifornia.
sns.scatterplot(x=data['longitude'],y=data['latitude'],hue=data['median_house_value'])
```

```
<AxesSubplot: xlabel='longitude', ylabel='latitude'>
```



Παλινδρόμηση Δεδομένων

1) Αλγόριθμος perceptron

Το ερώτημα αυτο δεν υλοποιήθηκε.

2) Least squares

Ο αλγόριθμος παλινδρόμησης ελαχίστων τετραγώνων χρησιμοποιείται για την εύρεση της γραμμής καλύτερης προσαρμογής που μοντελοποιεί τη σχέση μεταξύ των χαρακτηριστικών και των ετικετών στα δεδομένα. Η γραμμή καλύτερης προσαρμογής αναπαρίσταται από ένα σύνολο συντελεστών και οι συντελεστές αυτοί μπορούν να χρησιμοποιηθούν για να γίνουν προβλέψεις σε νέα δεδομένα.

```
def least_squares_train(X, y):
  mul1 = X.T.dot(X)
  inv1 = np.linalg.pinv(mul1)
  mul2 = X.T.dot(y)
  weight = np.matmul(inv1, mul2)
  return weight

def least_squares_predict(X, w):
  return np.matmul(X, w)
```

Εδώ υλοποιήθηκαν 2 συναρτήσεις όπου:
Η least_squares_train δέχεται δύο εισόδους, X και y, οι οποίες είναι τα χαρακτηριστικά και οι ετικέτες των δεδομένων που μοντελοποιούνται, αντίστοιχα. Η συνάρτηση υπολογίζει πρώτα το γινόμενο της αντιμετάθεσης των X και X και το αποθηκεύει στο mul1. Στη συνέχεια, υπολογίζει το ψευδοαντιστροφο του mul1 και το αποθηκεύει στο inv1. Στη συνέχεια, η συνάρτηση υπολογίζει το γινόμενο του transpose των X και y και το αποθηκεύει στο mul2. Τέλος, η συνάρτηση υπολογίζει το γινόμενο των inv1 και mul2 και το αποθηκεύει στο weight. Η μεταβλητή weight παίρνει τις τιμές των συντελεστών της γραμμής παλινδρόμησης που μοντελοποιεί τα δεδομένα εισόδου.

Η least_squares_predict δέχεται δύο εισόδους, X και w, οι οποίες είναι τα χαρακτηριστικά των δεδομένων και οι συντελεστές της γραμμής παλινδρόμησης, αντίστοιχα. Η συνάρτηση επιστρέφει το γινόμενο των X και w, το οποίο είναι η προβλεπόμενη τιμή της με βάση τα δεδομένα χαρακτηριστικά και τους συντελεστές παλινδρόμησης.

```
kf = KFold(n_splits=10)
for k, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    w = least_squares_train(X_train.to_numpy(), y_train.to_numpy())
    pred = least_squares_predict(X_test.to_numpy(), w)
    mse = mean_squared_error(y_test.to_numpy(), pred)
    mae = mean_absolute_error(y_test.to_numpy(), pred)
    print(f"Fold {k + 1} - MSE: {mse}")
    print(f"Fold {k + 1} - MAE: {mae}")
    print("\n")
```

Ο αλγόριθμος KFold(n_splits=10) δημιουργεί ένα αντικείμενο της κλάσης KFold από τη βιβλιοθήκη scikit-learn, όπου τα δεδομένα χωρίζονται σε 10 folds.

Η επανάληψη for περνάει από κάθε fold, όπου k είναι ο δείκτης του τρέχοντος fold (ξεκινώντας από το 0), train_index και test_index είναι οι δείκτες των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση και τη δοκιμή, αντίστοιχα.

X_train και y_train είναι τα δεδομένα εκπαίδευσης για τις μεταβλητές πρόβλεψης και τις μεταβλητές στόχου, αντίστοιχα. X_test και y_test είναι τα δεδομένα δοκιμής για τις μεταβλητές πρόβλεψης και τις μεταβλητές στόχου, αντίστοιχα.

Η συνάρτηση least_squares_train καλείται για να εκπαιδεύσει το μοντέλο χρησιμοποιώντας τα δεδομένα εκπαίδευσης και να υπολογίσει το βάρος ή τους συντελεστές (w) για το μοντέλο. Η συνάρτηση least_squares_predict χρησιμοποιείται στη συνέχεια για να γίνουν προβλέψεις στα δεδομένα δοκιμής χρησιμοποιώντας το βάρος.

Το μέσο τετραγωνικό σφάλμα (MSE) και το μέσο απόλυτο σφάλμα (MAE) υπολογίζονται χρησιμοποιώντας τις συναρτήσεις mean_squared_error και

mean_absolute_error της scikit-learn, αντίστοιχα. Οι τιμές αυτές αντιπροσωπεύουν τη διαφορά μεταξύ των πραγματικών τιμών και των προβλεπόμενων τιμών.

Τέλος, οι τιμές MSE και MAE εκτυπώνονται για κάθε fold μαζί με τον αριθμό του fold.

Αποτελεσμα:

```
Fold 1 - MSE: 7306445819.762907
Fold 1 - MAE: 69807.61393105896


Fold 2 - MSE: 3565413241.0120573
Fold 2 - MAE: 47373.14549299285


Fold 3 - MSE: 7755345331.922094
Fold 3 - MAE: 63224.759467397795


Fold 4 - MSE: 3940472038.5251455
Fold 4 - MAE: 47209.609502319436


Fold 5 - MSE: 6881773189.406185
Fold 5 - MAE: 61138.517272118814


Fold 6 - MSE: 4852123553.404929
Fold 6 - MAE: 47989.6854522758


Fold 7 - MSE: 2753517854.138328
Fold 7 - MAE: 37693.35666406169


Fold 8 - MSE: 8381066339.682972
Fold 8 - MAE: 68436.3153441601


Fold 9 - MSE: 6111689780.08459
Fold 9 - MAE: 57807.44509566541


Fold 10 - MSE: 2866911167.783155
Fold 10 - MAE: 38886.504697485354
```

3) Πολυστρωματικό νευρωνικό δίκτυο

```
for fold, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    z_train, z_test = z.iloc[train_index], z.iloc[test_index]

    model = Sequential()
    model.add(Dense(13, activation = 'relu', input_dim = 13))
    model.add(Dense(units = 13, activation = 'relu')) # Hidden layer 1
    model.add(Dens                                    # Hidden layer 2
    model.add(Dens  optimizer: Any
    model.compile(optimizer = 'adam', loss = 'mean_squared_error')

    model.fit(X_train, z_train, batch_size = 10, epochs = 5)
    z_pred = model.predict(X_test)

    mse = mean_squared_error(y_pred, y_test)
    mae = mean_absolute_error(y_pred, y_test)

    print(f'MSE for Fold Number: {fold + 1}, {mse}')
    print(f'MAE for Fold Number: {fold + 1}, {mae}')
    print("\n")
```

Ο παραπάνω κώδικας χρησιμοποιεί μια επανάληψη for για να εκτελέσει την K-Fold Cross Validation στα δεδομένα εισόδου, X και στα δεδομένα στόχου, z. Η K-Fold Cross Validation είναι μια τεχνική που χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός μοντέλου μηχανικής μάθησης, διαιρώντας τα δεδομένα σε k ίσα μέρη (ή Folds), εκπαιδεύοντας το μοντέλο σε k-1 μέρη και αξιολογώντας το στο υπόλοιπο. Αυτή η διαδικασία επαναλαμβάνεται k φορές με κάθε μέρος να χρησιμεύει ως σύνολο δοκιμής ακριβώς μία φορά.

Η μέθοδος kf.split(X) επιστρέφει τους δείκτες των δειγμάτων δεδομένων που θα πρέπει να χρησιμοποιηθούν για εκπαίδευση και δοκιμή σε κάθε fold, οι οποίοι στη συνέχεια χρησιμοποιούνται για το διαχωρισμό των δεδομένων εισόδου και στόχου σε σύνολα εκπαίδευσης και δοκιμής. Η συνάρτηση enumerate χρησιμοποιείται για την παρακολούθηση του τρέχοντος αριθμού fold.

Δημιουργείται ένα μοντέλο νευρωνικού δικτύου χρησιμοποιώντας την βιβλιοθήκη Keras Sequential. Το μοντέλο έχει τέσσερα πυκνά στρώματα, με το πρώτο στρώμα να έχει 13 νευρώνες, μια συνάρτηση ενεργοποίησης 'relu' και το σχήμα εισόδου 13. Τα επόμενα τρία στρώματα έχουν 13 νευρώνες το καθένα και η συνάρτηση ενεργοποίησης έχει επίσης οριστεί σε 'relu'. Το τελευταίο στρώμα έχει 1 νευρώνα και η συνάρτηση ενεργοποίησης δεν καθορίζεται, πράγμα που σημαίνει ότι θα χρησιμοποιηθεί η προεπιλεγμένη συνάρτηση

ενεργοποίησης 'linear'. Το μοντέλο μεταγλωττίζεται χρησιμοποιώντας τον βελτιστοποιηθεί 'adam' και η συνάρτηση απώλειας ορίζεται σε 'mean_squared_error'.

Στη συνέχεια, το μοντέλο εκπαιδεύεται χρησιμοποιώντας τα δεδομένα εκπαίδευσης για 5 εκδοχές με μέγεθος fold 10 . Στη συνέχεια γίνονται οι προβλέψεις για τα δεδομένα δοκιμής χρησιμοποιώντας τη μέθοδο predict και υπολογίζονται το μέσο τετραγωνικό σφάλμα (MSE) και το μέσο απόλυτο σφάλμα (MAE) μεταξύ των προβλέψεων και των πραγματικών τιμών χρησιμοποιώντας τις συναρτήσεις mean_squared_error και mean_absolute_error από τη βιβλιοθήκη scikit-learn αντίστοιχα. Τα MSE και MAE για κάθε Fold εκτυπώνονται στην κονσόλα.

Αποτελεσμα

```
Epoch 1/5
1858/1858 [==============================] - 3s 845us/step - loss: 0.0290
Epoch 2/5
1858/1858 [==============================] - 2s 845us/step - loss: 0.0202
Epoch 3/5
1858/1858 [==============================] - 2s 849us/step - loss: 0.0187
Epoch 4/5
1858/1858 [==============================] - 2s 852us/step - loss: 0.0177
Epoch 5/5
1858/1858 [==============================] - 2s 928us/step - loss: 0.0171
65/65 [==============================] - 0s 929us/step
MSE for Fold Number: 1, 2730345081.84659
MAE for Fold Number: 1, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 3s 1ms/step - loss: 0.0309
Epoch 2/5
1858/1858 [==============================] - 2s 864us/step - loss: 0.0211
Epoch 3/5
1858/1858 [==============================] - 2s 836us/step - loss: 0.0197
Epoch 4/5
1858/1858 [==============================] - 2s 835us/step - loss: 0.0191
Epoch 5/5
1858/1858 [==============================] - 2s 850us/step - loss: 0.0186
65/65 [==============================] - 0s 822us/step
MSE for Fold Number: 2, 2730345081.84659
MAE for Fold Number: 2, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 3s 955us/step - loss: 0.0249
Epoch 2/5
1858/1858 [==============================] - 2s 889us/step - loss: 0.0185
Epoch 3/5
1858/1858 [==============================] - 2s 840us/step - loss: 0.0177
Epoch 4/5
1858/1858 [==============================] - 2s 852us/step - loss: 0.0172
Epoch 5/5
1858/1858 [==============================] - 2s 845us/step - loss: 0.0166
65/65 [==============================] - 0s 706us/step
MSE for Fold Number: 3, 2730345081.84659
MAE for Fold Number: 3, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 2s 865us/step - loss: 0.0273
Epoch 2/5
1858/1858 [==============================] - 2s 878us/step - loss: 0.0200
Epoch 3/5
1858/1858 [==============================] - 2s 1ms/step - loss: 0.0190
Epoch 4/5
1858/1858 [==============================] - 2s 905us/step - loss: 0.0181
Epoch 5/5
1858/1858 [==============================] - 2s 896us/step - loss: 0.0172
65/65 [==============================] - 0s 731us/step
MSE for Fold Number: 4, 2730345081.84659
MAE for Fold Number: 4, 37496.21326440619
```

```
Epoch 1/5
1858/1858 [==============================] - 3s 1ms/step - loss: 0.0259
Epoch 2/5
1858/1858 [==============================] - 2s 977us/step - loss: 0.0194
Epoch 3/5
1858/1858 [==============================] - 2s 1ms/step - loss: 0.0184
Epoch 4/5
1858/1858 [==============================] - 2s 995us/step - loss: 0.0179
Epoch 5/5
1858/1858 [==============================] - 2s 918us/step - loss: 0.0173
65/65 [==============================] - 0s 741us/step
MSE for Fold Number: 5, 2730345081.84659
MAE for Fold Number: 5, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 2s 907us/step - loss: 0.0306
Epoch 2/5
1858/1858 [==============================] - 2s 922us/step - loss: 0.0197
Epoch 3/5
1858/1858 [==============================] - 2s 856us/step - loss: 0.0184
Epoch 4/5
1858/1858 [==============================] - 2s 844us/step - loss: 0.0175
Epoch 5/5
1858/1858 [==============================] - 2s 839us/step - loss: 0.0169
65/65 [==============================] - 0s 720us/step
MSE for Fold Number: 6, 2730345081.84659
MAE for Fold Number: 6, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 2s 874us/step - loss: 0.0317
Epoch 2/5
1858/1858 [==============================] - 2s 879us/step - loss: 0.0219
Epoch 3/5
1858/1858 [==============================] - 2s 847us/step - loss: 0.0208
Epoch 4/5
1858/1858 [==============================] - 2s 1ms/step - loss: 0.0199
Epoch 5/5
1858/1858 [==============================] - 2s 1ms/step - loss: 0.0193
65/65 [==============================] - 0s 893us/step
MSE for Fold Number: 7, 2730345081.84659
MAE for Fold Number: 7, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 3s 1ms/step - loss: 0.0280
Epoch 2/5
1858/1858 [==============================] - 2s 1ms/step - loss: 0.0186
Epoch 3/5
1858/1858 [==============================] - 2s 854us/step - loss: 0.0176
Epoch 4/5
1858/1858 [==============================] - 2s 880us/step - loss: 0.0169
Epoch 5/5
1858/1858 [==============================] - 2s 850us/step - loss: 0.0163
65/65 [==============================] - 0s 711us/step
MSE for Fold Number: 8, 2730345081.84659
MAE for Fold Number: 8, 37496.21326440619
```

```
Epoch 1/5
1858/1858 [==============================] - 2s 884us/step - loss: 0.0244
Epoch 2/5
1858/1858 [==============================] - 2s 871us/step - loss: 0.0187
Epoch 3/5
1858/1858 [==============================] - 2s 861us/step - loss: 0.0176
Epoch 4/5
1858/1858 [==============================] - 2s 860us/step - loss: 0.0170
Epoch 5/5
1858/1858 [==============================] - 2s 845us/step - loss: 0.0163
65/65 [==============================] - 0s 705us/step
MSE for Fold Number: 9, 2730345081.84659
MAE for Fold Number: 9, 37496.21326440619


Epoch 1/5
1858/1858 [==============================] - 2s 855us/step - loss: 0.0299
Epoch 2/5
1858/1858 [==============================] - 2s 920us/step - loss: 0.0205
Epoch 3/5
1858/1858 [==============================] - 2s 860us/step - loss: 0.0192
Epoch 4/5
1858/1858 [==============================] - 2s 872us/step - loss: 0.0182
Epoch 5/5
1858/1858 [==============================] - 2s 871us/step - loss: 0.0179
65/65 [==============================] - 0s 704us/step
MSE for Fold Number: 10, 2730345081.84659
MAE for Fold Number: 10, 37496.21326440619
```

Βιβλιογραφία

1) pandas
   https://www.scaler.com/topics/pandas/how-to-install-pandas-in-python/

2) matplotlib
   https://www.scaler.com/topics/matplotlib/install-matplotlib/

3) sklearn
   https://scikit-learn.org/stable/install.html

4) keras
   https://www.activestate.com/resources/quick-reads/how-to-install-keras-and-tensorflow/

5) seaborn
   https://seaborn.pydata.org/installing.html

6) numpy
   https://www.edureka.co/blog/install-numpy/

7) ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ,S.Theodoridis, K. Koutroumbas
8) Οι σημειωσεις του μαθηματος στο gunet
9) Min-Max scaling:
   https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
10) ONE-HOT encode :
    https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Τέλος,
Σας ευχαριστούμε πολύ