Πρόβλεψη τιμών ακινήτων

Load τα βιβλιοθήκες.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.linear_regression import LinearRegression, RidgeCV, LassoCV, ElasticNetCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import cross_val_score, KFold
```

```
pd.set_option('display.max_columns', 20)
pd.set_option('display.width', 120)
```

```
RANDOM_STATE = 42
```

Λίγοι φακελάκια: η βιβλιοθήκη sklearn απαιτεί τα αριθμητικά δεδομένα και τα κενά. Θα χρειαστούμε τις τιμές βιβλιοθηκών τώρα, ώστε η ανάλυση παρακάτω να μη σκάει στη μέση.

```
df = pd.read_csv('housing.csv')
df.shape
```

```
(20640, 10)
```

Κοιτάμε φορτώνουμε το δεδομένο από το αρχείο σε ένα pandas dataframe και με τη συνήθεια: head() βλέπουμε τις πρώτες 5 γραμμές.

```
df.head()
```

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

Η ανάλυση: βλέπουμε μερικές τιμές μερικές στατιστικές, πλήρεις μετρήσεις και τα δεδομένα

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816279
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Είναι ο κανονικός κόσμος μας; Ναι, οι τιμές δίνουν μόνο ενδιαφέρον.

```
df.info()
```

```
<class_name>.DataFrame RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 longitude           20640 non-null  float64
 latitude            20640 non-null  float64
 housing_median_age  20640 non-null  float64
 total_rooms         20640 non-null  float64
 total_bedrooms      20433 non-null  float64
 population          20640 non-null  float64
 households          20640 non-null  float64
 median_income       20640 non-null  float64
 median_house_value  20640 non-null  float64
 ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory_usage: 1.6+ MB
```

Ξαναμετράμε τα χαρακτηριστικά στις κατηγορίες, 8 και 9 στην μετάβαση 9 θα μου πει τον χαρακτηριστικό στήλης median_house_value.

```
df.select_dtypes(include=['object', 'category', 'bool', 'int', 'float', 'datetime', 'timedelta'].shape[1]
```

```
9
```

Ξαναμετράμε τα αριθμητικά χαρακτηριστικά από τα κατηγορικά

```
len(df.select_dtypes(include=['object', 'category', 'bool'].shape[1]
```

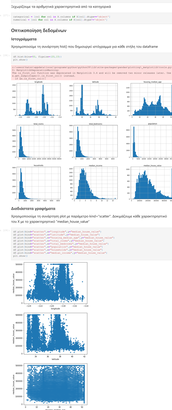
```
1
```

Οπτικοποίηση δεδομένων

Ιστογράμματα

Χρησιμοποιούμε τη συνάρτηση hist() που δημιουργεί ιστογράμμα για κάθε στήλη του dataframe

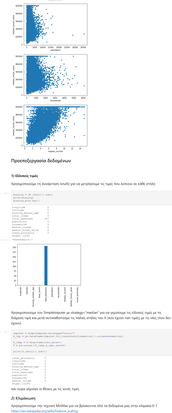
```
df.hist(bins=50, figsize=(20,15))
plt.show()
```



Διαδραστικό χαρτογράφο

Χρησιμοποιούμε τη συνάρτηση plot με παράμετρο kind='scatter'. Δημιουργίας κάθε χαρακτήρα median_house_value με τη χαρακτηριστικό "median_house_value"

```
df.plot(kind='scatter', x='longitude', y='latitude', c='median_house_value')
df.plot(kind='scatter', x='longitude', y='latitude', c='median_house_value', cmap='jet')
df.plot(kind='scatter', x='longitude', y='latitude', c='median_house_value', cmap='jet', alpha=0.1)
df.plot(kind='scatter', x='longitude', y='latitude', c='median_house_value', cmap='jet', alpha=0.1, figsize=(10,7))
df.plot(kind='scatter', x='longitude', y='latitude', c='median_house_value', cmap='jet', alpha=0.1, figsize=(10,7), colorbar=False)
```



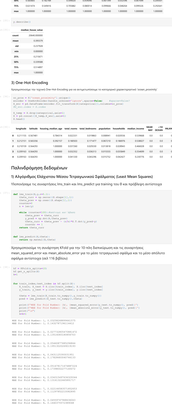
Προεπεξεργασία δεδομένων

Ύπαιθρος κωδικός

Χρησιμοποιούμε τη συνάρτηση isnull() για να μετρήσουμε τις τιμές που λείπουν σε κάθε στήλη:

```
df.isnull().sum
df.isnull().sum()
```

```
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
```



Χρησιμοποιούμε τον ξεπεταμένο με σύνταξη "median" για να μετρήσουμε τις ελλείπες τιμές με τις δείχνει τιμές και μετά αντικαθιστώνται τις κενές στήλες τιμές τα ίδιου τμήματα τιμών τιμές με τις νέες που δεν ήρθανε.

```
median = df.loc[df['total_bedrooms'].notnull()]['total_bedrooms'].median()
df.loc[df['total_bedrooms'].isnull(), 'total_bedrooms'] = median
df.loc[df['total_bedrooms'].isnull(), 'total_bedrooms'] = df.loc[df['total_bedrooms'].isnull(), 'total_bedrooms'].fillna(median)
```

```
df[df['total_bedrooms'].isnull()]
```

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity

```
df.isnull().sum()
```

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

Χρησιμοποιούμε τη συνάρτηση Κλειδί για τον 10 κάτω δοκιμαζόμενα και τις ονομασίες τους ενώνουμε μαζί και κοιτάμε μόνο μία από τα μέρη τη χρησιμοποιούν κομμάτια και τα μέρη αποτελούν κομμάτια κατάστημα κατά 19 βαθμίδες.

```
df['ocean_proximity'].unique()
```

```
array(['NEAR BAY', 'INLAND', 'NEAR OCEAN', 'ISLAND', '<1H OCEAN'],
      dtype=object)
```

Αν κάνουμε dummies τώρα θα δούμε πως οι μετρητές και οι ιδιόχρωτες τιμές είναι πολύ πιο κοντά ή και 1 κατάστημα.

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	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	537.870553	1425.476744	499.539680	3.870671	206855.816279
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	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	537.870553	1425.476744	499.539680	3.870671	206855.816279
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

One-Hot Encoding

Κλιμάκωμα Ελάχιστου Μέγιστου Τετραγωνικού Σφάλματος (Least Square)

Υπολογίζουμε τις συναρτήσεις και, έτσι και τώρα, μπορεί μα θαστις τις 4 και πρόβλεψης κατάστημα

```
def get_train(df):
    '''Extract SklearnDataset from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']]
```

```
def get_target(df):
    '''Extract SklearnDataset target from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df['median_house_value']
```

```
X = get_train(df)
y = get_target(df)
```

```
X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']] = \
    StandardScaler().fit_transform(X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']])
```

```
X.shape, y.shape
```

```
((20640, 9), (20640,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=RANDOM_STATE)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((16512, 9), (4128, 9), (16512,), (4128,))
```

Δίκτυο Κλιμάκωμα Τετραγωνικού Σφάλματος (Least Squares)

Υπολογίζουμε τις συναρτήσεις, έτσι και τώρα, μπορεί μα θαστις τις 4 και πρόβλεψης κατάστημα και τα μέρη αποτελούν κομμάτια κατάστημα κατά 19 βαθμίδες.

```
def get_train(df):
    '''Extract SklearnDataset from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']]
```

```
def get_target(df):
    '''Extract SklearnDataset target from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df['median_house_value']
```

```
X = get_train(df)
y = get_target(df)
```

```
X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']] = \
    MinMaxScaler().fit_transform(X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']]
```

```
X.shape, y.shape
```

```
((20640, 9), (20640,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=RANDOM_STATE)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((16512, 9), (4128, 9), (16512,), (4128,))
```

Σε κάθε iteration τώρα θαφ κάνουμε train στα training set και predict στα test set

```
X_train, X_test
```

```
(array([[-0.13307, -0.84188,  0.34933, ...,  0.12157,  0.16731,  0.56420],
        [-0.35780, -0.12494,  0.24756, ...,  0.01419,  0.26294,  0.565  ],
        [ 0.14053, -0.75771,  1.856  , ...,  0.04884,  0.05735,  0.565  ],
        [-0.69929,  1.00946,  0.02919, ...,  0.03444,  0.03444,  0.565  ],
        [-0.81723,  0.77283,  0.43933, ...,  0.05202,  0.05202,  0.565  ],
        [-0.77213,  0.74283,  0.34933, ...,  0.04884,  0.04884,  0.565  ]],
       dtype=float64),
array[[-0.13307, -0.84188,  0.34933, ...,  0.12157,  0.16731,  0.56420],
        [-0.35780, -0.12494,  0.24756, ...,  0.01419,  0.26294,  0.565  ],
        [ 0.14053, -0.75771,  1.856  , ...,  0.04884,  0.05735,  0.565  ],
        [-0.69929,  1.00946,  0.02919, ...,  0.03444,  0.03444,  0.565  ],
        [-0.81723,  0.77283,  0.43933, ...,  0.05202,  0.05202,  0.565  ],
        [-0.77213,  0.74283,  0.34933, ...,  0.04884,  0.04884,  0.565  ]],
       dtype=float64)
```

Δίκτυο Κλιμάκωμα Τετραγωνικού Σφάλματος (Least Squares)

Υπολογίζουμε τις συναρτήσεις, έτσι και τώρα, μπορεί μα θαστις τις 4 και πρόβλεψης κατάστημα και τα μέρη αποτελούν κομμάτια κατάστημα κατά 19 βαθμίδες.

```
def get_train(df):
    '''Extract SklearnDataset from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']]
```

```
def get_target(df):
    '''Extract SklearnDataset target from df.
    :param df: the dataset to extract.
    :return: the new dataset.
    '''
    return df['median_house_value']
```

```
X = get_train(df)
y = get_target(df)
```

```
X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']] = \
    StandardScaler().fit_transform(X[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity']]
```

```
X.shape, y.shape
```

```
((20640, 9), (20640,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=RANDOM_STATE)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((16512, 9), (4128, 9), (16512,), (4128,))
```

Σε κάθε iteration τώρα θαφ κάνουμε train στα training set και predict στα test set

```
X_train, X_test
```

```
(array([[-0.13307, -0.84188,  0.34933, ...,  0.12157,  0.16731,  0.56420],
        [-0.35780, -0.12494,  0.24756, ...,  0.01419,  0.26294,  0.565  ],
        [ 0.14053, -0.75771,  1.856  , ...,  0.04884,  0.05735,  0.565  ],
        [-0.69929,  1.00946,  0.02919, ...,  0.03444,  0.03444,  0.565  ],
        [-0.81723,  0.77283,  0.43933, ...,  0.05202,  0.05202,  0.565  ],
        [-0.77213,  0.74283,  0.34933, ...,  0.04884,  0.04884,  0.565  ]],
       dtype=float64),
array[[-0.13307, -0.84188,  0.34933, ...,  0.12157,  0.16731,  0.56420],
        [-0.35780, -0.12494,  0.24756, ...,  0.01419,  0.26294,  0.565  ],
        [ 0.14053, -0.75771,  1.856  , ...,  0.04884,  0.05735,  0.565  ],
        [-0.69929,  1.00946,  0.02919, ...,  0.03444,  0.03444,  0.565  ],
        [-0.81723,  0.77283,  0.43933, ...,  0.05202,  0.05202,  0.565  ],
        [-0.77213,  0.74283,  0.34933, ...,  0.04884,  0.04884,  0.565  ]],
       dtype=float64)
```

Σ: Νευρωνικό δίκτυο

Υπολογίζουμε τον νευρωνικό δίκτυο με τη βιβλιοθήκη keras. Το ίδιο ήμαι ίσα πομα ίμαι. Σ'πλάνο ήμαι μεγάλος, 13 και ένα ένα ομαγα ίμαι. Χρησιμοποιούμε μόνο 5 φορές μα λόγους περίσσεια. Σε ομάμαγα ίσια μικρό

```
epochs = 5
batch_size = 128
model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)
```

```
Epoch 1/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.5000
Epoch 2/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.4000
Epoch 3/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3000
Epoch 4/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2000
Epoch 5/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1000
```

```
model.evaluate(X_test, y_test)
```

```
Epoch 1/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.5000
Epoch 2/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.4000
Epoch 3/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3000
Epoch 4/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2000
Epoch 5/5
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1000
```