0.PreparingDataOceanProximityClassification

This California Housing Prices dataset has been downloaded from StatLib repository (http://lib.stat.cmu.edu/datasets/). It is based on data from the 1990 California census. It is not recent, but this is not important for deep learning. The original dataset appeared in R. Kelley Pace and Ronald Barry, "Sparse Spatial Autoregressions," Statistics & Probability Letters 33, no. 3 (1997): 291–297.

Data for each instance (observation) is referred to a block group in California, which could be corresponded to a district, with a population of 600 to 3,000 people, and 1,425.5 on average.

OceanProximityPreparedCleanAttributes.csv The original dataset contained 20,640 instances, which is cleaned, preprocessed and prepared in this notebook. After this phase of data preparation, a final dataset of 20,433 instances are obtained with 9 attributes individually normalized with a min-max scaling, $\frac{x-min}{max-min}$: *longitude* and *latitude* (location), *median age*, *total rooms*, *total bedrooms*, *population*, *households*, *median income* and *median house value*.

From this data, the classification problem consists on estimating the location (label *ocean proximity*), categorized into five clases: "<1H OCEAN", "INLAND" "NEAR BAY", "NEAR OCEAN" and "ISLAND". We will see that label "ISLAND" only has 5 instances. Therefore, they are removed from the dataset. The remaining classes are labelled from 0 ("<1H OCEAN") to 3 ("NEAR OCEAN"), and one-hot encoded in OceanProximityOneHotEncodedClasses.csv file for supervised training models.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.preprocessing import LabelEncoder, OneHotEncoder, minmax_scale
        import matplotlib.pyplot as plt
```

```
In [2]: INPUT_FILE_NAME = "HousingRawDataset.csv"
        ATT_FILE_NAME = "OceanProximityPreparedCleanAttributes.csv"
        ONE_HOT_ENCODED_LABEL_FILE_NAME = "OceanProximityOneHotEncodedClasses.csv"
```

```
In [3]: dataset = pd.read_csv(INPUT_FILE_NAME)
```

```
In [4]: dataset.info()
```

```
<class 'pandas.core.frame.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
```

In [5]:
```
dataset.iloc[:10]
```

Out[5]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0
5    -122.25     37.85                52.0        919.0           213.0
6    -122.25     37.84                52.0       2535.0           489.0
7    -122.25     37.84                52.0       3104.0           687.0
8    -122.26     37.84                42.0       2555.0           665.0
9    -122.25     37.84                52.0       3549.0           707.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
5       413.0       193.0         4.0368            269700.0        NEAR BAY
6      1094.0       514.0         3.6591            299200.0        NEAR BAY
7      1157.0       647.0         3.1200            241400.0        NEAR BAY
8      1206.0       595.0         2.0804            226700.0        NEAR BAY
9      1551.0       714.0         3.6912            261100.0        NEAR BAY
```

In [6]:
```
dataset.iloc[-10:]
```

Out[6]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
20630    -121.32     39.29                11.0       2640.0           505.0
20631    -121.40     39.33                15.0       2655.0           493.0
20632    -121.45     39.26                15.0       2319.0           416.0
20633    -121.53     39.19                27.0       2080.0           412.0
20634    -121.56     39.27                28.0       2332.0           395.0
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
```

```
20639    -121.24     39.37                16.0       2785.0           616.0

          population  households  median_income  median_house_value  \
20630       1257.0       445.0         3.5673            112000.0
20631       1200.0       432.0         3.5179            107200.0
20632       1047.0       385.0         3.1250            115600.0
20633       1082.0       382.0         2.5495             98300.0
20634       1041.0       344.0         3.7125            116800.0
20635        845.0       330.0         1.5603             78100.0
20636        356.0       114.0         2.5568             77100.0
20637       1007.0       433.0         1.7000             92300.0
20638        741.0       349.0         1.8672             84700.0
20639       1387.0       530.0         2.3886             89400.0

         ocean_proximity
20630             INLAND
20631             INLAND
20632             INLAND
20633             INLAND
20634             INLAND
20635             INLAND
20636             INLAND
20637             INLAND
20638             INLAND
20639             INLAND
```

In [7]:
```
dataset.info()
```

```
<class 'pandas.core.frame.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
```

In [8]:
```
labels=list(set(dataset["ocean_proximity"]))
labels
```

Out[8]:
```
['ISLAND', 'NEAR OCEAN', 'NEAR BAY', '<1H OCEAN', 'INLAND']
```

First Step: find out wheather or not there are missing values and, in such case, remove them

```
In [9]: {att : dataset[dataset[att].isnull()].shape[0] for att in dataset.columns}
```

```
Out[9]: {'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}
```

total_bedrooms has 207 missing values. The corresponding rows are removed.

```
In [10]: dataset.dropna(inplace=True)
```

```
In [11]: {att : dataset[dataset[att].isnull()].shape[0] for att in dataset.columns}
```

```
Out[11]: {'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}
```

Second Step: check how many instances per labels there are

```
In [12]: dataset["ocean_proximity"].value_counts()
```

```
Out[12]: <1H OCEAN     9034
         INLAND        6496
         NEAR OCEAN    2628
         NEAR BAY      2270
         ISLAND           5
         Name: ocean_proximity, dtype: int64
```

Label *ISLAND* is removed since there are only 5 examples

```
In [13]: dataset[dataset["ocean_proximity"]=="ISLAND"]
```

```
Out[13]:       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
         8314    -118.32     33.35                27.0       1675.0           521.0
         8315    -118.33     33.34                52.0       2359.0           591.0
```

```
8316    -118.32    33.33                52.0       2127.0           512.0
8317    -118.32    33.34                52.0        996.0           264.0
8318    -118.48    33.43                29.0        716.0           214.0

      population  households  median_income  median_house_value  \
8314       744.0       331.0         2.1579            450000.0
8315      1100.0       431.0         2.8333            414700.0
8316       733.0       288.0         3.3906            300000.0
8317       341.0       160.0         2.7361            450000.0
8318       422.0       173.0         2.6042            287500.0

     ocean_proximity
8314          ISLAND
8315          ISLAND
8316          ISLAND
8317          ISLAND
8318          ISLAND
```

In [14]:
```
dataset.drop(index=range(8314,8319),inplace=True)
```

In [15]:
```
dataset["ocean_proximity"].value_counts()
```

Out[15]:
```
<1H OCEAN     9034
INLAND        6496
NEAR OCEAN    2628
NEAR BAY      2270
Name: ocean_proximity, dtype: int64
```

In [16]:
```
labels=list(set(dataset["ocean_proximity"]))
labels
```

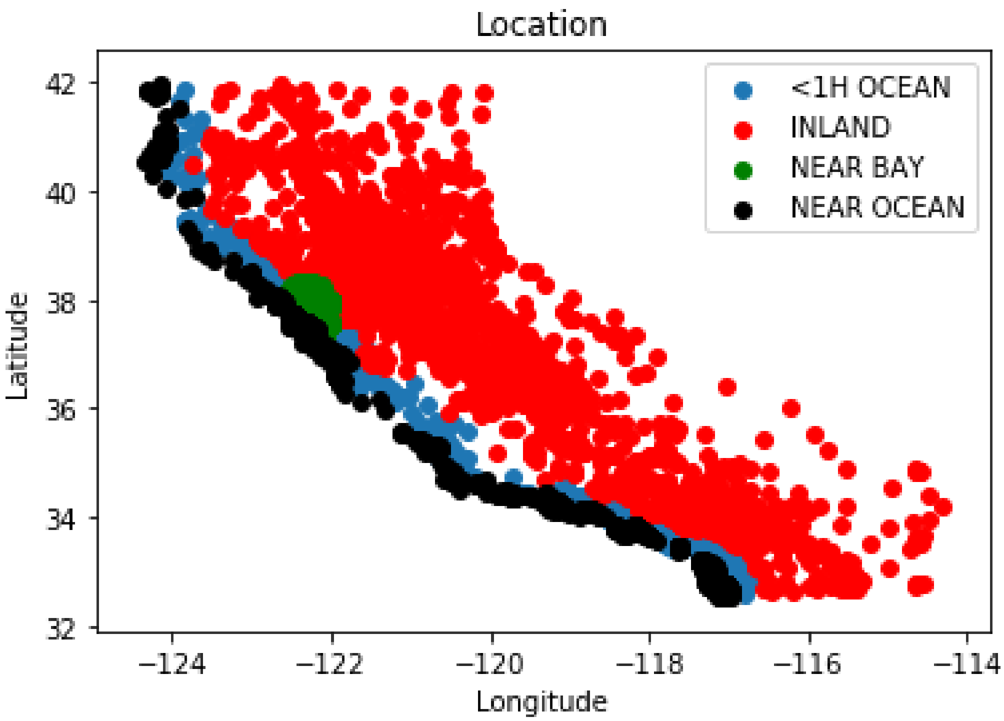
Out[16]:
```
['NEAR OCEAN', '<1H OCEAN', 'INLAND', 'NEAR BAY']
```

In [17]:
```
dataset.shape
```

Out[17]:
```
(20428, 10)
```

Location of the block goups (houses) included in the clean dataset:

In [18]:
```
plt.title("Location")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.scatter(dataset[dataset["ocean_proximity"]=="<1H OCEAN"]["longitude"],
            dataset[dataset["ocean_proximity"]=="<1H OCEAN"]["latitude"])
plt.scatter(dataset[dataset["ocean_proximity"]=="INLAND"]["longitude"],
            dataset[dataset["ocean_proximity"]=="INLAND"]["latitude"],c="r")
plt.scatter(dataset[dataset["ocean_proximity"]=="NEAR BAY"]["longitude"],
            dataset[dataset["ocean_proximity"]=="NEAR BAY"]["latitude"],c="g")
plt.scatter(dataset[dataset["ocean_proximity"]=="NEAR OCEAN"]["longitude"],
            dataset[dataset["ocean_proximity"]=="NEAR OCEAN"]["latitude"],c="k")
plt.legend(["<1H OCEAN","INLAND","NEAR BAY","NEAR OCEAN"])
plt.show()
```



Third Step: suffle the dataset (three times)

In [19]:
```
dataset=dataset.sample(frac=1) #frac is the fraction of axis items to return. 1 means a
dataset=dataset.sample(frac=1)
dataset=dataset.sample(frac=1).reset_index(drop=True) #Reset index and drop the old one
dataset.head()
```

Out[19]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms \
0	-124.05	40.85	31.0	2414.0	428.0
1	-122.19	39.50	23.0	462.0	97.0
2	-117.20	33.70	23.0	6323.0	1196.0
3	-119.55	36.71	32.0	1963.0	508.0
4	-119.22	34.18	17.0	3332.0	762.0

	population	households	median_income	median_house_value	ocean_proximity
0	1005.0	401.0	3.5156	143000.0	NEAR OCEAN
1	261.0	90.0	2.1705	53000.0	INLAND
2	1984.0	1124.0	2.3276	92400.0	<1H OCEAN
3	2052.0	518.0	1.9076	55800.0	INLAND
4	1797.0	673.0	4.4292	231200.0	NEAR OCEAN

Fourth Step: split dataset vertically into attributes x and label t for supervised learning

In [20]:
```
t = dataset["ocean_proximity"]
t[:10]
```

```
Out[20]: 0    NEAR OCEAN
         1        INLAND
         2     <1H OCEAN
         3        INLAND
         4    NEAR OCEAN
         5     <1H OCEAN
         6    NEAR OCEAN
         7     <1H OCEAN
         8     <1H OCEAN
         9        INLAND
         Name: ocean_proximity, dtype: object
```

```
In [21]: x = dataset.drop (columns="ocean_proximity")
```

```
In [22]: x.head()
```

```
Out[22]:    longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
         0    -124.05     40.85                31.0       2414.0           428.0
         1    -122.19     39.50                23.0        462.0            97.0
         2    -117.20     33.70                23.0       6323.0          1196.0
         3    -119.55     36.71                32.0       1963.0           508.0
         4    -119.22     34.18                17.0       3332.0           762.0

            population  households  median_income  median_house_value
         0      1005.0       401.0         3.5156            143000.0
         1       261.0        90.0         2.1705             53000.0
         2      1984.0      1124.0         2.3276             92400.0
         3      2052.0       518.0         1.9076             55800.0
         4      1797.0       673.0         4.4292            231200.0
```

Fifth Step: discretizing and one-hot encoding of labels (target values)

```
In [23]: encoder = LabelEncoder() # Function that transform non-numeral labels into integers.
         discretized_t = encoder.fit_transform(t.values)
         labels =  [encoder.inverse_transform([value]) for value in range(4)] #hold the label fo
         discretized_t[:10]
```

```
Out[23]: array([3, 1, 0, 1, 3, 0, 3, 0, 0, 1])
```

```
In [24]: discretized_t = pd.DataFrame(data=discretized_t,columns=["ocean_proximity"])
         discretized_t[:10]
```

```
Out[24]:    ocean_proximity
         0                3
         1                1
         2                0
         3                1
         4                3
         5                0
```

```
6              3
7              0
8              0
9              1
```

In [25]:
```
labels = np.array(labels).reshape(1,-1)[0]
labels
```

Out[25]:
```
array(['<1H OCEAN', 'INLAND', 'NEAR BAY', 'NEAR OCEAN'], dtype=object)
```

In [26]:
```
{ value: encoder.inverse_transform([value]) for value in range(4)}
```

Out[26]:
```
{0: array(['<1H OCEAN'], dtype=object),
 1: array(['INLAND'], dtype=object),
 2: array(['NEAR BAY'], dtype=object),
 3: array(['NEAR OCEAN'], dtype=object)}
```

In [27]:
```
encoder = OneHotEncoder(sparse=False) # Function that one-hot encoders integers
#one_hot_t = encoder.fit_transform(discretized_t.reshape(-1,1)).toarray()
one_hot_t = encoder.fit_transform (t.values.reshape(-1,1))
one_hot_t[:10]
```

Out[27]:
```
array([[0., 0., 0., 1.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.]])
```

In [28]:
```
one_hot_t = pd.DataFrame(data=one_hot_t,columns=labels)
one_hot_t[:10]
```

Out[28]:

	<1H OCEAN	INLAND	NEAR BAY	NEAR OCEAN
0	0.0	0.0	0.0	1.0
1	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	1.0
5	1.0	0.0	0.0	0.0
6	0.0	0.0	0.0	1.0
7	1.0	0.0	0.0	0.0
8	1.0	0.0	0.0	0.0
9	0.0	1.0	0.0	0.0

Sixth Step: Normalization of the input dataset within the range [-1,1]

In [29]: x = pd.DataFrame (minmax_scale (x, feature_range=(-1,1)),columns=x.columns)

In [30]: x[:10]

Out[30]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
0	-0.940239	0.766206	0.176471	-0.877308	-0.867474
1	-0.569721	0.479277	-0.137255	-0.976601	-0.970205
2	0.424303	-0.753454	-0.137255	-0.678468	-0.629112
3	-0.043825	-0.113709	0.215686	-0.900249	-0.842644
4	0.021912	-0.651435	-0.372549	-0.830612	-0.763811
5	-0.679283	0.336876	-0.568627	-0.770436	-0.718187
6	0.235060	-0.738576	0.019608	-0.845313	-0.756052
7	0.217131	-0.647184	0.568627	-0.894959	-0.860025
8	0.207171	-0.670563	1.000000	-0.935399	-0.900372
9	-0.057769	-0.190223	-0.647059	-0.813215	-0.772502

	population	households	median_income	median_house_value
0	-0.943833	-0.868443	-0.584047	-0.472163
1	-0.985538	-0.970728	-0.769576	-0.843295
2	-0.888954	-0.630653	-0.747907	-0.680822
3	-0.885143	-0.829962	-0.805837	-0.831749
4	-0.899437	-0.778984	-0.458035	-0.108453
5	-0.913058	-0.826673	-0.606461	-0.687420
6	-0.921803	-0.754646	-0.578916	0.056082
7	-0.928305	-0.850682	-0.299913	0.216082
8	-0.945010	-0.884230	-0.806016	0.016907
9	-0.895681	-0.764184	-0.708335	-0.730719

Some descriptive statistics on the attributes to confirm the max and min values

In [31]: x.describe()

Out[31]:

	longitude	latitude	housing_median_age	total_rooms
count	20428.000000	20428.000000	20428.000000	20428.000000
mean	-0.048005	-0.342449	0.083519	-0.865977
std	0.399150	0.454052	0.493732	0.111168
min	-1.000000	-1.000000	-1.000000	-1.000000
25%	-0.492032	-0.704570	-0.333333	-0.926344
50%	0.165339	-0.634431	0.098039	-0.891907
75%	0.262948	0.100956	0.411765	-0.840213
max	1.000000	1.000000	1.000000	1.000000

	total_bedrooms	population	households	median_income
count	20428.000000	20428.000000	20428.000000	20428.000000
mean	-0.833365	-0.920282	-0.836051	-0.534967
std	0.130796	0.063526	0.125745	0.261987
min	-1.000000	-1.000000	-1.000000	-1.000000
25%	-0.908442	-0.955997	-0.908239	-0.715383
50%	-0.865301	-0.934808	-0.865812	-0.581026

```
75%        -0.799503     -0.903585     -0.801677      -0.414605
max         1.000000      1.000000      1.000000       1.000000

       median_house_value
count        20428.000000
mean            -0.208981
std              0.475925
min             -1.000000
25%             -0.569173
50%             -0.320823
75%              0.029691
max              1.000000
```

The correlation matrix permits to visualize dependencies between pairs of attributes: values close to -1 or +1 indicate high correlation. A negative correlation value means than when the value of an attribute gets high the value of the other attribute decreases, and viceversa. Positive correlation values means that both attributes increase or deacrese simultaneously. For example *population* and *households* have high positive correlation, while high negative correlations do not occur. *population* and *median_income* are highly uncorrelated.

```
In [32]: x.corr()
```

```
Out[32]:                     longitude  latitude  housing_median_age  total_rooms  \
         longitude            1.000000 -0.924628           -0.109556     0.045555
         latitude            -0.924628  1.000000            0.012192    -0.036799
         housing_median_age  -0.109556  0.012192            1.000000    -0.360634
         total_rooms          0.045555 -0.036799           -0.360634     1.000000
         total_bedrooms       0.069653 -0.067066           -0.320486     0.930382
         population           0.100380 -0.109193           -0.295715     0.857273
         households           0.056604 -0.071939           -0.302714     0.918987
         median_income       -0.015464 -0.079796           -0.118191     0.197822
         median_house_value  -0.045642 -0.144312            0.106077     0.133516

                             total_bedrooms  population  households  median_income  \
         longitude                 0.069653    0.100380    0.056604      -0.015464
         latitude                 -0.067066   -0.109193   -0.071939      -0.079796
         housing_median_age       -0.320486   -0.295715   -0.302714      -0.118191
         total_rooms               0.930382    0.857273    0.918987       0.197822
         total_bedrooms            1.000000    0.877758    0.979740      -0.007767
         population                0.877758    1.000000    0.907177       0.004989
         households                0.979740    0.907177    1.000000       0.013350
         median_income            -0.007767    0.004989    0.013350       1.000000
         median_house_value        0.049792   -0.025069    0.065122       0.688848

                             median_house_value
         longitude                    -0.045642
         latitude                     -0.144312
         housing_median_age            0.106077
```

```
total_rooms                  0.133516
total_bedrooms               0.049792
population                  -0.025069
households                   0.065122
median_income                0.688848
median_house_value           1.000000
```

Finally, both attributes matrix x and target labels t are saved to csv files.

```
In [33]: x.to_csv(ATT_FILE_NAME, index=False)
         #discretized_t.to_csv(DISCRETIZED_LABEL_FILE_NAME, index=False)
         one_hot_t.to_csv(ONE_HOT_ENCODED_LABEL_FILE_NAME, index=False)
```