Tutorial 3 for Chapter 2

Case study 6: Real Estate Ownership Classification by Logistic Regression

Reference: Python数据挖掘实战

For the course AMA546 Statistical Data Mining
Lecturer: Dr. Catherine Liu
AMA, PolyU, HKSAR

Contents

In [1]:

```
import pandas
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib. pyplot as plt
```

executed in 1.80s, finished 09:36:25 2023-02-20

1 Objectives of the analysis

In this case study, we will use **logistic regression** to predict the **ownership of the real estate**.

The initiator of this case study was the government department of a city in China, who wants to know the ownership state of the real estate in the city. However, the cost of household census is extremely high. So the government wants to predict the ownership state of the real estate through the data related to the real estate.

The government has found that whether a real estate is owned by the occupant is related to the personal **circumstances of the household** and the **condition of the house**. This allows us to predict whether a real estate is owned by the occupant using housing data collected from the estate management office and the cable TV company.

2 Description of the data

In [2]:

```
### Load data
owner = pandas.read_csv('data/Ownership.csv')
owner.head()
```

executed in 20ms, finished 09:36:25 2023-02-20

Out[2]:

	Age	Education Level	Gender	Internet Connection	Marital Status	Movie Selector	Num Bathrooms	Num Bedrooms	Num Cars	C
0	33	Doctorate	Male	Dial-Up	Married	Spouse/Partner	2.5	3	1	
1	47	Doctorate	Male	DSL	Married	Spouse/Partner	2.0	2	2	
2	35	Bachelor's Degree	Male	Cable Modem	Married	Me	2.5	4	2	
3	32	Bachelor's Degree	Male	Cable Modem	Married	Me	3.5	5	2	
4	32	Bachelor's Degree	Male	No Internet Connection	Married	Me	2.5	4	2	

5 rows × 21 columns

The dataset contain 7 columns of numerical data and 14 column of categorical data. The column `Home Ownership` is the response variable we want to predict. There are 3085 rows (samples) in the dataset. The explanations and range of the features are attached below:

Variable	Explanation	Range
Age	Age of the house owner	[20, 62]
Education Level	Education level of the house owner	{Doctorate, Bachelor's Degree, Master's Degree, Associate's Degree, Some College, Post-Doc, Trade School, High School, Grade School}
Gender	Gender of the house owner	{Male, Female}
Home Ownership	House ownership	{Own, Rent}
Internet Connection	Internet connection type of the house owner	{Dial-Up, DSL, Cable Modem, No Internet Connection, Other, IDSN}
Marital Status	Marital status of the house owner	{Married, Divorced, Never Married, Separated, Other}

Variable	Explanation	Range
Movie Selector	The type of the movie selector	{Spouse/Partner, Me, Other, Children}
Num Bathrooms	Number of bathrooms in the house owner's home	[0.5, 5]
Num Bedrooms	Number of bedrooms in the house owner's home	[1, 10]
Num Cars	Number of cars owned by the house owner	[1, 6]
Num Children	Number of children of the house owner	[0, 6]
Num TVs	Number of televisions in the house owner's home	[0, 10]
PPV Freq	Frequency of Pay-Per-View of the house owner	{Rarely, Never, Monthly, Weekly, Daily}
Prerec Buying Freq	Frequency of pre-recorded movie buying	{Monthly, Rarely, Never, Weekly, Daily}
Prerec Format	Pre-recorded movie format	{DVD, VHS, Betamax, Laserdisk, Video CD, Other}
Prerec Renting Freq	Frequency of pre-recorded movie renting	{Rarely, Monthly, Weekly, Never, Daily}
Prerec Viewing Freq	Frequency of pre-recorded movie viewing	{Monthly, Weekly, Rarely, Daily, Never}
CustomerID	House owner identifier	[877687, 927818]
Theater Freq	Frequency of theater movie watching	{Monthly, Rarely, Weekly, Never, Daily}
TV Movie Freq	Frequency of TV movie watching	{Monthly, Weekly, Daily, Rarely, Never}
TV Signal	Type of TV signal	{Cable, Digital Satellite, Analog antennae, Don't watch TV, Analog Satellite}

3 Exploratory data analysis

3.1 Data quality (No need in our case)

3.1.1 Data Cleaning

We'll check for any missing values (**NA or Null**) in our dataset using the `isna()` method in Python. Fortunately, no such values were found, and data cleaning is not necessary in this regard. Note that .isnull() is equivalent to .isna(), so we can use either method to check for missing values.

In [3]:

```python
### There is no NA or Null data in the dataframe
print(owner[owner.isna().any(axis=1)])
```

executed in 7ms, finished 09:36:25 2023-02-20

```
Empty DataFrame
Columns: [Age, Education Level, Gender, Internet Connection, Marital S
tatus, Movie Selector, Num Bathrooms, Num Bedrooms, Num Cars, Num Chil
dren, Num TVs, PPV Freq, Prerec Buying Freq, Prerec Format, Prerec Ren
ting Freq, Prerec Viewing Freq, CustomerID, Theater Freq, TV Movie Fre
q, TV Signal, Home Ownership]
Index: []

[0 rows x 21 columns]
```

3.1.2 Data Validation

Through the analysis of continuous variables and categorical variables, their values are within the reasonable range. All of the data appears to be reasonable and **no inconsistencies** are immediately noticeable.

In [4]:

```python
### Descriptive statistics for continous variables
owner.describe()
```

executed in 11ms, finished 09:36:25 2023-02-20

Out[4]:

	Age	Num Bathrooms	Num Bedrooms	Num Cars	Num Children	Num TVs	Custom
count	3085.000000	3085.000000	3085.000000	3085.000000	3085.000000	3085.000000	3085.00
mean	33.156240	2.096759	2.878444	1.682658	0.806483	2.098865	897854.73
std	7.239346	0.812988	1.150117	0.802138	1.111995	1.230140	12692.68
min	20.000000	0.500000	1.000000	0.000000	0.000000	0.000000	877687.00
25%	28.000000	1.500000	2.000000	1.000000	0.000000	1.000000	888787.00
50%	32.000000	2.000000	3.000000	2.000000	0.000000	2.000000	891654.00
75%	38.000000	2.500000	4.000000	2.000000	2.000000	3.000000	912515.00
max	62.000000	5.000000	10.000000	6.000000	6.000000	10.000000	927818.00

In [5]:

```
### Possiable values for discrete variables
# set display option
pandas.set_option('max_colwidth', None)
# categorical features
oneHotColumns = [
    'Gender', 'Internet Connection', 'Marital Status',
    'Movie Selector', 'Prerec Format', 'TV Signal',
    'Education Level', 'PPV Freq', 'Theater Freq',
    'TV Movie Freq', 'Prerec Buying Freq',
    'Prerec Renting Freq', 'Prerec Viewing Freq'
]

# store the categorical feature
unique_values = {}
for column in oneHotColumns:
    unique_values[column] = owner[column].unique()
result_df = pandas.DataFrame(list(unique_values.items()), columns=["Column", "Ur
result_df
```

executed in 10ms, finished 09:36:25 2023-02-20

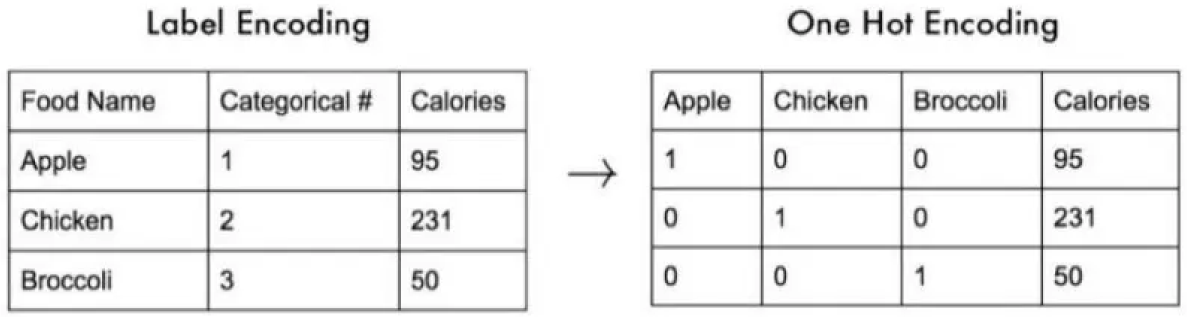
Out[5]:

	Column	Unique Values
0	Gender	[Male, Female]
1	Internet Connection	[Dial-Up, DSL, Cable Modem, No Internet Connection, Other, IDSN]
2	Marital Status	[Married, Divorced, Never Married, Separated, Other]
3	Movie Selector	[Spouse/Partner, Me, Other, Children]
4	Prerec Format	[DVD, VHS, Betamax, Laserdisk, Video CD, Other]
5	TV Signal	[Cable, Digital Satellite, Analog antennae, Don't watch TV, Analog Satellite]
6	Education Level	[Doctorate, Bachelor's Degree, Master's Degree, Associate's Degree, Some College, Post-Doc, Trade School, High School, Grade School]
7	PPV Freq	[Rarely, Never, Monthly, Weekly, Daily]
8	Theater Freq	[Monthly, Rarely, Weekly, Never, Daily]
9	TV Movie Freq	[Monthly, Weekly, Daily, Rarely, Never]
10	Prerec Buying Freq	[Monthly, Rarely, Never, Weekly, Daily]
11	Prerec Renting Freq	[Rarely, Monthly, Weekly, Never, Daily]
12	Prerec Viewing Freq	[Monthly, Weekly, Rarely, Daily, Never]

3.1.3 Data Transformation

Since there are many **string variables** in the dataset, we need to transform them for later processing.

- One transformation method is the **label encoding.** It is a **direct numeric representation of a string variable.** The values for the different values of the variable will be determined in the order in which they appear in the dataset. **We will use this coding method for our exploratory data analysis.**



Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

- Another way to convert is the **one-hot encoding** we discussed in the early tutorial. We first **separate** the string variables from the numeric variable, and then **train** the one-hot encoder with the string variable to get the transformation rules and **encode** them. We then **combine** the encoded data with the numeric variable to create our explanatory dataframe. The response variable remains in string format, as this does not affect our analysis. **We will use this encoding for logistic regression.**

In [6]:

```
### One-hot encoding
# categorical features
oneHotColumns = [
    'Gender', 'Internet Connection', 'Marital Status',
    'Movie Selector', 'Prerec Format', 'TV Signal',
    'Education Level', 'PPV Freq', 'Theater Freq',
    'TV Movie Freq', 'Prerec Buying Freq',
    'Prerec Renting Freq', 'Prerec Viewing Freq'
]
# numerical features
numericColumns = [
    'Age', 'Num Bathrooms',
    'Num Bedrooms', 'Num Cars',
    'Num Children', 'Num TVs'
]

# import OneHotEncoder module
from sklearn.preprocessing import OneHotEncoder
# build a OneHotEncoder object
oneHotEncoder = OneHotEncoder()
# called on the encoder object to learn the encoding rules
oneHotEncoder.fit(owner[oneHotColumns])
# produce a one-hot encoded representation of the categorical features
oneHotData = oneHotEncoder.transform(owner[oneHotColumns])

from scipy.sparse import hstack
# horizontally stack the one-hot encoded data with the numerical data
x = hstack([
    oneHotData,
    owner[numericColumns].astype(float).values
])
# response variable
y = owner['Home Ownership']
```

executed in 14ms, finished 09:36:25 2023-02-20

In [7]:

```python
### Label encoder
# import label encoder
from sklearn import preprocessing
# make an instance of Label Encoder
label_encoder = preprocessing.LabelEncoder()
# make a copy of the original dataframe
owner_ecd = owner.copy()
# perform label encoding
for i in oneHotColumns:
    owner_ecd[i] = label_encoder.fit_transform(owner[i])
# preview
owner_ecd.head()
```

executed in 16ms, finished 09:36:25 2023-02-20

Out[7]:

	Age	Education Level	Gender	Internet Connection	Marital Status	Movie Selector	Num Bathrooms	Num Bedrooms	Num Cars	Nun Childrer
0	33	2	1	2	1	3	2.5	3	1	(
1	47	2	1	1	1	3	2.0	2	2	-
2	35	1	1	0	1	1	2.5	4	2	2
3	32	1	1	0	1	1	3.5	5	2	2
4	32	1	1	4	1	1	2.5	4	2	(

5 rows × 21 columns

As you can see, the columns except for the response variable are encoded by 0, 1, 2, ...

3.2 Marginal variable analysis

3.2.1 Home Ownership

We begin with an analysis of the possible **`Home Ownership`** . In the data set we obtained, there are a total of two possible `Home Ownership` , which is `Own` and `Rent` . Among them, 64.7% of the observations own houses, and the remaining 35.3% of the observations rent houses. This suggests that **our observed data is somewhat imbalanced**, and we need to be careful when building our model and constructing the confusion matrix.

In [8]:

```
### Use crosstab to get the proportion of label
renewal_counts = pandas.crosstab(index=owner["Home Ownership"],
                                 columns="Proportion",
                                 normalize='all') *100
round(renewal_counts,1).astype(str).apply(lambda x:x + '%')
```

executed in 10ms, finished 09:36:25 2023-02-20

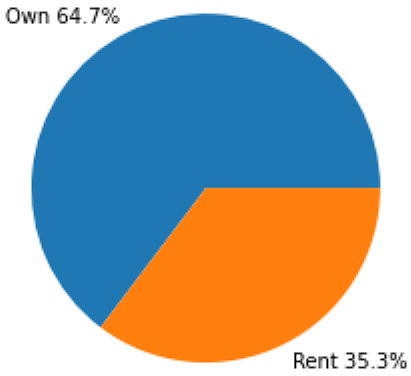
Out[8]:

col_0	Proportion
Home Ownership	
Own	64.7%
Rent	35.3%

In [9]:

```
### In pie plot
plt.pie(renewal_counts['Proportion'], labels = ['Own 64.7%', 'Rent 35.3%']); plt
```

executed in 39ms, finished 09:36:25 2023-02-20



The same as the former case study, we will **analyze all the explanatory variables together**, and **select a few variables of interest to explore in more detail.** The goal is to classify the home ownership, so we need to **pay more attention to the relationship between each explanatory variable and the `Home Ownership`.**

3.2.2 Scatter plot matrix

In a scatter plot, the **scatter points overlap** when both the X-axis and Y-axis are categorical variables. In this case, the scatter plot loses its meaning. Here, we **only focus on the distribution plots on the diagonal**.

In a scatter plot below:

- For feature `Age` , **homeowners are significantly older** than renters.
- For feature `martial status` , those with **value 1 (married) have a higher proportion of home ownership** and those with value 2 (unmarried) have a higher proportion of renting.

- For feature `Num Bathroom` , `Num Bedroom` , `Num Cars` , and `Num Children` , **homeowners with larger values (larger numbers)** have a higher proportion of home ownership. These phenomenon are also consistent with the common sense.

In [10]:

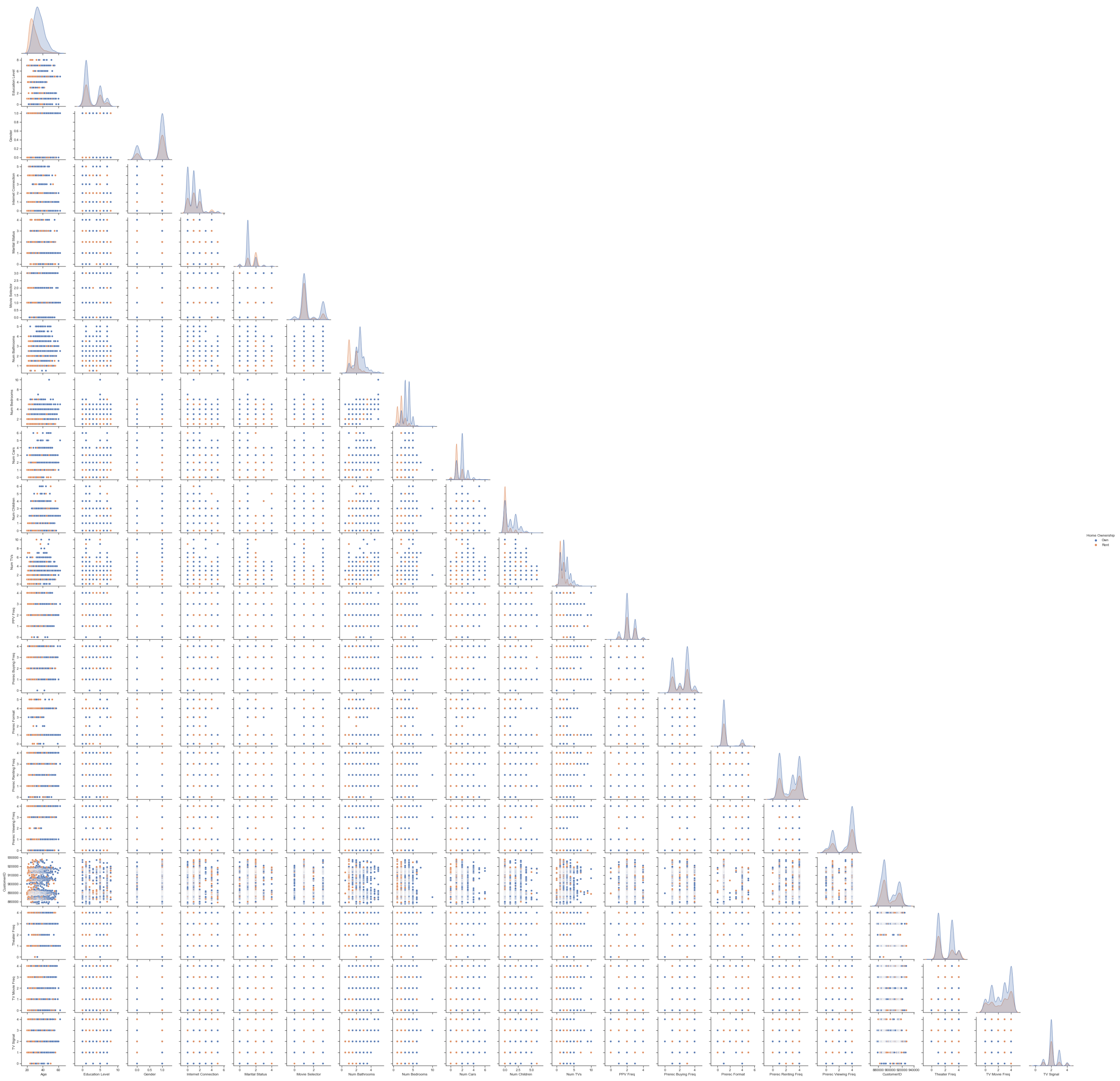
```python
# Scatter plot matrices of explanatory variables
# it takes about 55 secounds to run
sns.set_theme(style="ticks")
plt.figure(figsize=(32, 32), dpi=1800)
sns.pairplot(owner_ecd, hue="Home Ownership", corner=True)
```

executed in 36.9s, finished 09:37:02 2023-02-20

Out[10]:

```
<seaborn.axisgrid.PairGrid at 0x7fd049a8d2e0>

<Figure size 57600x57600 with 0 Axes>
```



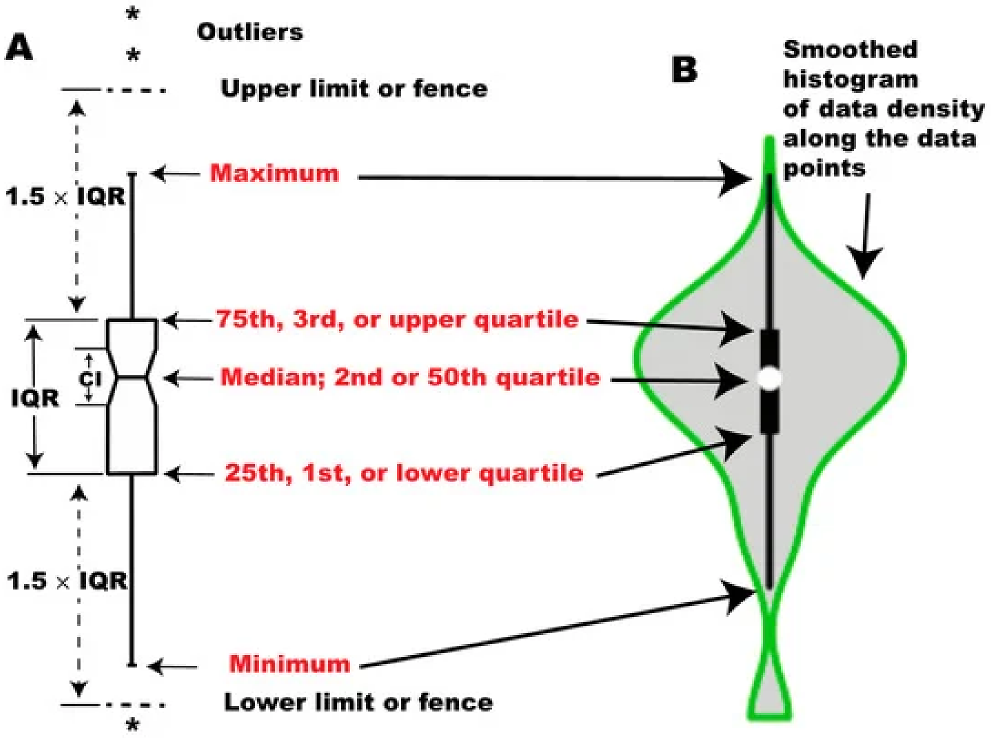
3.2.3 Violin plot of Age, Num Bathrooms, Num Bedrooms, Num Cars and Num Children

For the five variables have significant discriminative effects in the above analysis, we draw violin plots to explore their classification effects.

The **violin plot** is a popular visualization tool used to display the **distribution** of a dataset. It is **a combination of a box plot and a kernel density plot**, which resembles a violin.

In a violin plot, the **distribution** of the data is **displayed as a kernel density estimate**, with the width of the plot at each point reflecting the density of the data at that point. The **plot also includes a box plot**, which provides information about the quartiles, median, and outliers of the data. The box plot itself is often drawn as a thin line within the thicker violin plot. The resulting plot provides a useful and informative summary of the distribution of the data.

Graph A is the boxplot, Graph B is the violin plot:

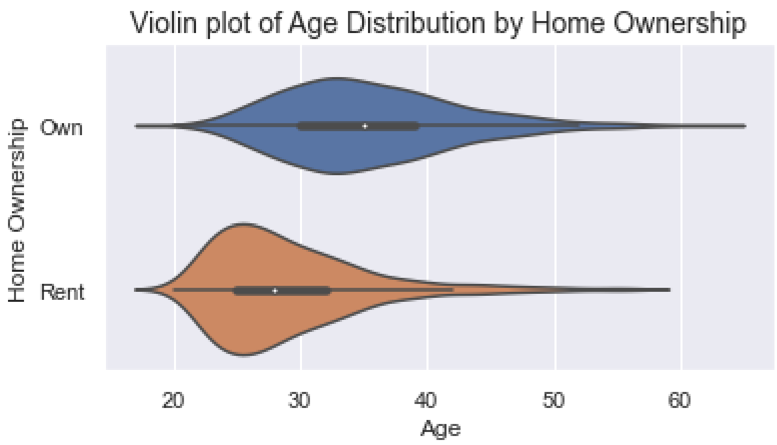

A
*
*
Outliers
Upper limit or fence
B
Smoothed histogram of data density along the data points
1.5 × IQR
Maximum
75th, 3rd, or upper quartile
CI
IQR
Median; 2nd or 50th quartile
25th, 1st, or lower quartile
1.5 × IQR
Minimum
Lower limit or fence
*

Age:
As can be seen in the figure, the distribution of Rent is concentrated between **20 and 30 years old**, while the distribution of Own is concentrated between **27 and 40 years old.** The distributions of different classes is different, but there is also **some overlap**.

In [11]:

```python
# Set the style
sns.set(style="ticks")
sns.set_style("darkgrid")
# plot the violinplot
fig, ax = plt.subplots(1, 1, figsize=(6, 3))
sns.violinplot(y="Home Ownership", x="Age", data=owner, ax=ax, orient="h")
ax.set_title("Violin plot of Age Distribution by Home Ownership", fontsize=14)
plt.show()
```

executed in 80ms, finished 09:37:02 2023-02-20



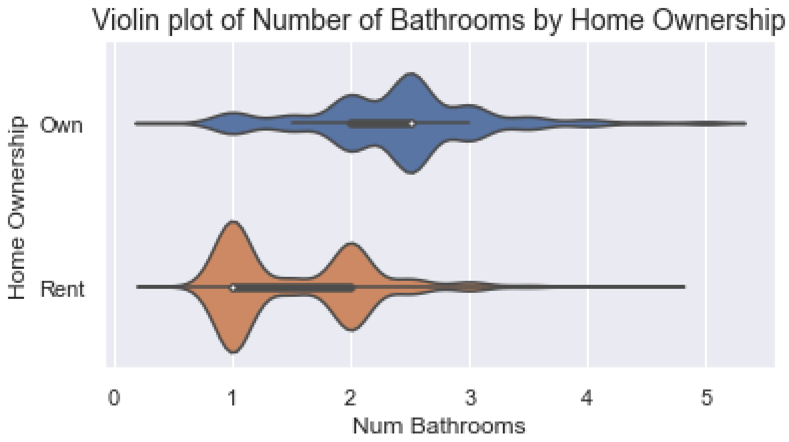
Num Bathrooms:

`Rent` usually only has 1-2 bathrooms, while `Own` usually has 2-4 bathrooms. The distinguishing effect of Bathrooms is relatively obvious, and we will also further observe this variable in the following analysis.

In [12]:

```python
fig, ax = plt.subplots(1, 1, figsize=(6, 3))
sns.violinplot(y="Home Ownership", x="Num Bathrooms", data=owner, ax=ax, orient=
ax.set_title("Violin plot of Number of Bathrooms by Home Ownership", fontsize=14
plt.show()
```

executed in 79ms, finished 09:37:02 2023-02-20



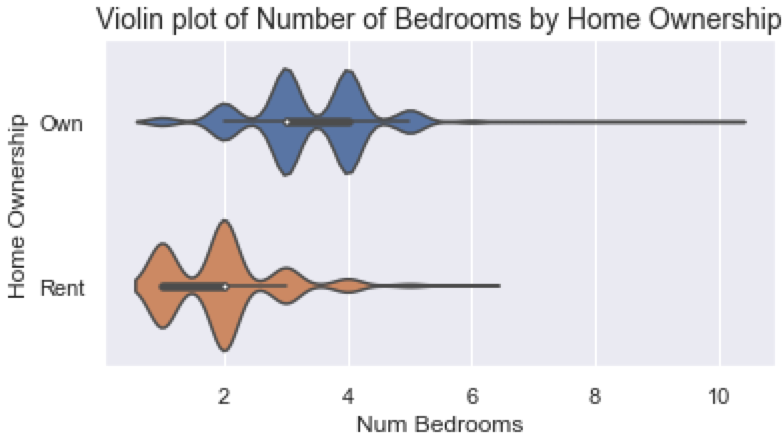
Num Bedrooms:

`Rent` usually has 1-3 bedrooms, and `Own` has 2-4 bathrooms. The distinguishing effect of the `Num Bedrooms` is not as obvious as the `Num Bathrooms`. Notice that `Own` has some outliers on the `Num Bedrooms`.

In [13]:

```
fig, ax = plt.subplots(1, 1, figsize=(6, 3))
sns.violinplot(y="Home Ownership", x="Num Bedrooms", data=owner, ax=ax, orient='
ax.set_title("Violin plot of Number of Bedrooms by Home Ownership", fontsize=14)
plt.show()
```

executed in 76ms, finished 09:37:02 2023-02-20

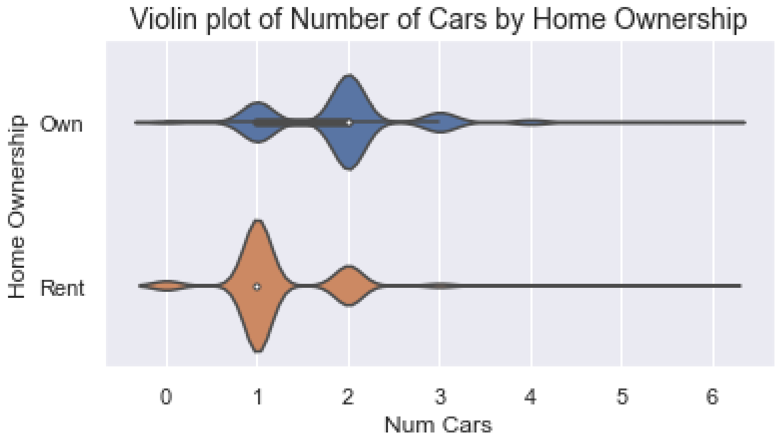


Num Cars:
As for the number of cars owned, `Rent` range between 0-2 cars, while `Own` will typically have 1-3 cars and is unlikely to have no cars. Both `Rent` and `Own` both have some outliers.

In [14]:

```
fig, ax = plt.subplots(1, 1, figsize=(6, 3))
sns.violinplot(y="Home Ownership", x="Num Cars", data=owner, ax=ax, orient="h")
ax.set_title("Violin plot of Number of Cars by Home Ownership", fontsize=14)
plt.show()
```

executed in 81ms, finished 09:37:03 2023-02-20

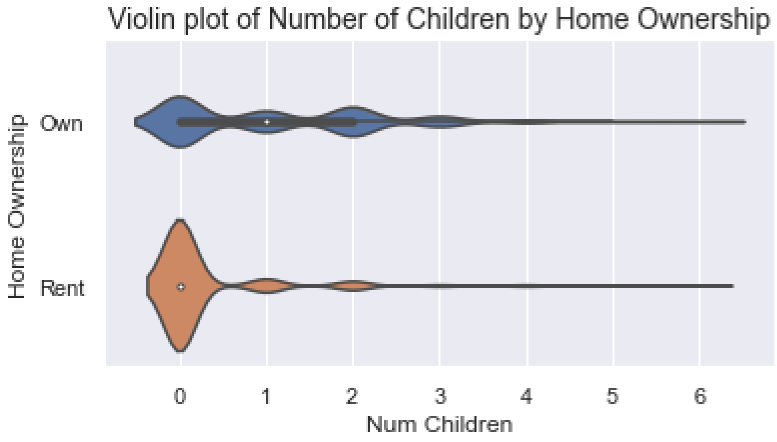


Num Children:
Most `Rent` has no children, and only a few have 1-2 children. While `Own` usually has 0-3 children.

In [15]:

```
fig, ax = plt.subplots(1, 1, figsize=(6, 3))
sns.violinplot(y="Home Ownership", x="Num Children", data=owner, ax=ax, orient='
ax.set_title("Violin plot of Number of Children by Home Ownership", fontsize=14)
plt.show()
```

executed in 81ms, finished 09:37:03 2023-02-20



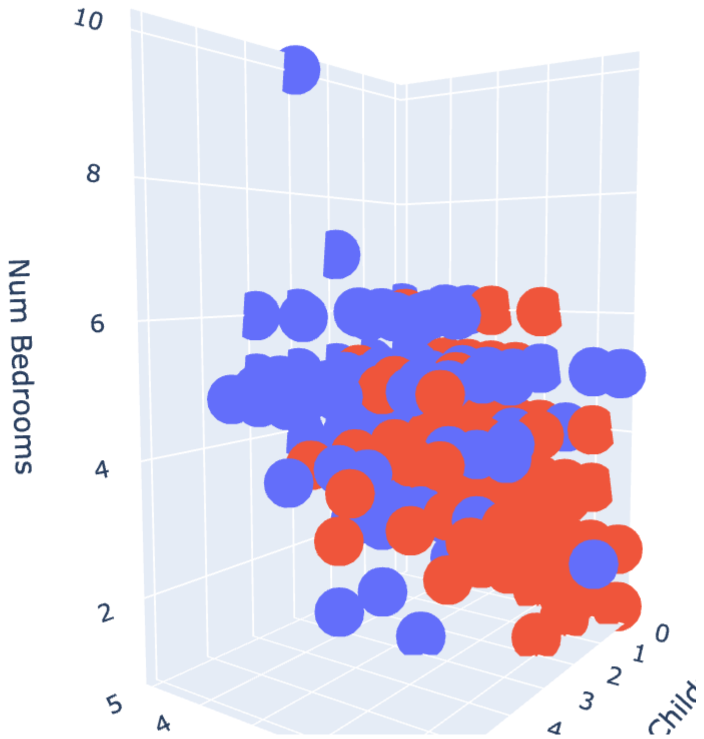
3.2.4 Exploring the Relationships using 3D Scatter Plot

The following plot is **interactive**, allowing you to explore the internal structure of the 3D scatter plot by zooming or rotating it with your mouse. We created this plot to investigate the classification effects of the variables `Num Bathrooms` , `Num Children` , and `Num Bedrooms` as an example. By plotting these three dimensions in a 3D scatter plot, we aim to gain insight into the relationship between these variables and potentially identify any patterns or trends that may exist. *You may explore the relationship of other variable pairs.*

In [16]:

```
import plotly.express as px
fig = px.scatter_3d(owner, x='Num Bathrooms', y='Num Children', z='Num Bedrooms'
              color='Home Ownership')
fig.show()
```

executed in 686ms, finished 09:37:03 2023-02-20



It can be seen that although there is a certain degree of differentiation between `Rent` and `Own` in space, the overlap between the clusters is serious. This suggests that `Rent` and `Own` are **difficult to distinguish in low-dimensional space using only a few variables**.

Conclusion:

- In the **Data quality** section, we mainly encoded the original dataset.
  - We first use the **Label encoder** to transform string variables into numerical variables to facilitate our exploratory data analysis.
  - After that we use **One-hot encoder** to encode the string variable into one-hot form, which is convenient for us to perform logistic regression.
- In **Marginal variable analysis**:
  - for the response variable Home Ownership, we find that there are more samples labeled `Own` than `Rent` ,
  - for the explanatory variables, we find that `Age` , `Num Bathroom` , `Num Bedroom` , `Num Cars` , and `Num Children` have a good discriminative effect on Home Ownership. We further analyzed the discriminative effect of these variables with **violinplot**. Finally, we used **3D Scatter Plot** to analyze the

classification effect of the combination of different variables. We find that **a few variables do not distinguish `Home Ownership` well**.

3.3 Data preprocessing

For the dataset after one-hot encoding, we only need to split the training set and the test set.

3.3.1 Split the training and testing set

In [17]:

```
# Split the dataset into training set (70%) and testing set (30%), the random_st
from sklearn.model_selection import train_test_split # Import the training and t
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
```

executed in 5ms, finished 09:37:03 2023-02-20

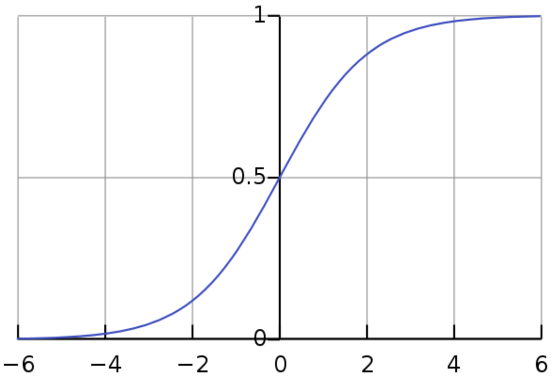
4 Model building

In the following, we will use **logistic regression** to classify the `Home Ownership`.

Logistic Regression is a statistical method for analyzing a dataset in which there are **one or more explanatory variables** that determine an outcome. It is used for **binary classification problems** where the dependent variable only has two possible outcomes, such as yes or no, pass or fail, true or false, etc.

Basic idea: Use a logistic function to **model the probability** of the dependent variable being a certain class. Mathematically, the logistic function can be represented as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$



where $z = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in}$, x_i is the vector of independent variables for the i^{th} observation and $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients.

Coefficients estimation: The coefficients are estimated through the **maximum likelihood estimation (MLE) method**. The MLE method is based on the assumption that the observed data follows a certain probability distribution. In logistic regression, the response variable is binary and **follows a Bernoulli distribution**. The **conditional probability of the response variable** given the predictor variables x_i and the coefficients β can be represented as:

$$p(y_i|x_i, \beta) = \sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in})^{y_i}(1 - \sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in}))^{1-y_i} \tag{2}$$

where y_i is the response variable for the i^{th} observation and $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic function. $\sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in})$ is the logistic function, can be interpreted as the probability of y_i being class 1: $P(y_i = 1 | x_i, \beta)$.

- $(\sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in}))^{y_i}$ is equal to 1 if y_i is class 1, and 0 otherwise.
- $(1 - \sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in}))^{1-y_i}$ is equal to 1 if y_i is class 0, and 0 otherwise.

The **likelihood function** is the product of the conditional probabilities of all observations:

$$L(\beta) = \prod_{i=1}^{N} p(y_i | x_i, \beta) \tag{3}$$

where N is the number of observations.

The **log-likelihood function** is the logarithm of the likelihood function:

$$l(\beta) = \ln L(\beta) = \sum_{i=1}^{N} [y_i \ln(\sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in})) + (1 - y_i) \ln(1 - \sigma(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$$

The coefficients are estimated by finding the values of $\beta_0, \beta_1, ..., \beta_n$ that maximize the log-likelihood function using gradient descent.

Prediction: Once the coefficients are estimated, the prediction of the dependent variable can be made using the following equation:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq \tau \\ 0 & \text{if } \sigma(z) < \tau \end{cases} \tag{5}$$

The **logistic regression model provides a probability score for each observation**, indicating the **likelihood of it belonging to a certain class**. This score can then be transformed into a binary outcome using a threshold value τ, usually $\tau = 0.5$.

In conclusion, logistic regression is a powerful and widely used machine learning technique for binary classification problems. It is simple to implement, can handle a large number of independent variables, and provides probabilities for each observation, making it easy to interpret and understand the results.

In the following analysis, we will build a logistic regression model directly on the training set.

In [18]:

```python
from sklearn.linear_model import LogisticRegression  # import the LogisticRegres
from sklearn.model_selection import cross_val_score  # import the cross validati
# build a logistic regression object
lrModel = LogisticRegression(solver='newton-cg')
# perform the cross validation
lrModel.fit(x_train, y_train)
# get the Accuracy of the ogistic regression on training data
lrModel.score(x_train, y_train)
```

executed in 67ms, finished 09:37:03 2023-02-20

Out[18]:

```
0.8499305233904585
```

Logistic regression achieves an **Accuracy** of 0.8499 on the training set.

5 Model evaluation (based on testing data)

5.1 Accuracy

In [19]:

```
lrModel.score(x_test, y_test)
```

executed in 4ms, finished 09:37:03 2023-02-20

Out[19]:

```
0.8369330453563715
```

On the **testing set**, the logistic regression **Accuracy is 0.8369**. The Accuracy vary only a little compared to the training set. This indicates that the **generalization ability** of our logistic regression is relatively **strong**.

5.2 Confusion matrices

The confusion matrix is a table used to evaluate the performance of a classifier. It shows the number or the proportion of correct and incorrect predictions made by the classifier for each class. The matrix is usually presented as a 2x2 table for binary classification problems, where one class represents `Rent` (1) class and the other class represents `Own` (0) class. The **rows** in the table represent the **actual** class, while the **columns** represent the **predicted** class.

In [20]:

```
# Define a function to calculate the confusion matrix
from sklearn.metrics import confusion_matrix # import confusion matrix module
def calculate_confusion_matrix(y_true, y_pred, labels):
    confu_mat = pandas.DataFrame(confusion_matrix(y_true, y_pred,
                                                  normalize=None,  # calculate t
                                                  labels=labels))  # pos label =
    return round(confu_mat, 2).astype(str)  # formatting the output
# Confusion matrix of KNN mdel with k=17
calculate_confusion_matrix(y_test, lrModel.predict(x_test), labels=['Own', 'Rent
```

executed in 6ms, finished 09:37:03 2023-02-20

Out[20]:

	0	1
0	525	64
1	87	250

For the following confusion matrix, 0 means label=Own and 1 means label=Rent. The logistic regression classifier misclassified 64 out of 589 samples (10.87%) with label=Own to label=Rent and 87 out of 337 samples (25.82%) with label=Rent to label=Own. In general, the logistic regression classifier has a **strong ability to identify samples with label=Own**.

6 Summary report

6.1 Context

The impetus for this case study originated from a city government department in China, which sought to determine the ownership status of real estate within the city. However, conducting a household census can be prohibitively expensive. Therefore, the government aimed to **predict real estate ownership** status using **data related to the properties** instead.

6.2 Objectives

This case study aims to **predict the home ownership of real estate** using **logistic regression** based on the data related to the house.

6.3 Organisation of the data

The dataset consists of 21 columns including the response variable, Home Ownership. It has 7 numerical columns and 14 categorical columns. The dataset contains 3085 rows (samples) with each row representing a different house owner. The features include age, education level, gender, internet connection, marital status, number of bathrooms, number of bedrooms, number of cars, number of children, number of televisions, PPV frequency, pre-recorded movie buying frequency, pre-recorded movie format, pre-recorded movie renting frequency, pre-recorded movie viewing frequency, customer ID, theater frequency, TV movie frequency, and TV signal type. The range of each feature is also given in the table provided. We will use this dataset to predict the home ownership of the house owners.

6.4 Exploratory data analysis

During the exploratory data analysis, we assessed the quality of the data by performing various operations such as data cleaning, data validation, and data transformation. We then conducted marginal variable analysis to investigate the impact of different variables on the classification of home ownership.

- In the **Data quality** section, we mainly encoded the original dataset.
    - We first use the **Label encoder** to transform string variables into numerical variables to facilitate our exploratory data analysis.
    - After that we use **One-hot encoder** to encode the string variable into one-hot form, which is convenient for us to perform logistic regression.
- In **Marginal variable analysis**:
    - for the response variable Home Ownership, we find that there are more samples labeled `Own` than `Rent` ,
    - for the explanatory variables, we find that `Age` , `Num Bathroom` , `Num Bedroom` , `Num Cars` , and `Num Children` have a good discriminative effect on Home Ownership. We further analyzed the discriminative effect of these variables with **violinplot**. Finally, we used **3D Scatter Plot** to analyze the classification effect of the combination of different variables. We find that **a few variables do not distinguish `Home Ownership` well**.

6.5 Model specification

The aim of our study is to devise a **binary** classification model capable of predicting whether a house is owned by its resident. The features are encoded in **one-hot format**, making **logistic regression** a suitable method for classification

6.6 Model evaluation

30% of the observations are reserved for model evaluation. In the present study, the logistic regression classifier was evaluated using both **training and test sets**, and the corresponding **Accuracy** scores were found to be **0.8499 and 0.8369**, respectively. The results indicate that the accuracy of the classifier is not significantly reduced by the dataset transformation, thereby reflecting its **robust generalization ability**. The confusion matrix analysis revealed that 10.87% of the negative samples were misclassified, while 25.82% of the positive samples were misclassified. These findings suggest that the logistic regression classifier is **proficient at identifying samples with a `Own` label**.