第一次作业

请复习《机器学习实战》教材的第 2“端到端的机器学习项目”，仔细阅读该章节的配套源码“02_end_to_end_machine_learning_project.ipynb”，并仿照该源码，尝试写一段 python 程序，利用 SVM 来解决房价预测问题。并完成以下问答题。

1、 问题设定。通过沟通确定了业务目标，明确了要解决的问题属于回归问题，并选择了相应的性能衡量指标。本章中，所使用的性能衡量指标是什么？

均方根误差(RMSE, Root Mean Squared Error)

$$\mathrm{RMSE}(\mathbf{X},h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(\mathbf{x}^{(i)}) - y^{(i)}\right)^2}$$

2、 获取数据。假定数据集已下载到本地，

（1） 如何加载数据集？请写出相应的代码。

预先手动下载数据集“housing.csv”文件到项目文件的根目录，故拼接“.”和文件名“housing.csv”作为路径加载数据集。使用 pandas 的 read_csv 方法读取数据，返回 DataFrame 对象。

```
def load_housing_data(housing_path="."):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)

housing = load_housing_data()
```

（2） 数据集的总行数是？每个属性的类型是？哪个属性存在缺失值？

20640 行；除了 ocean_proximity 是 object 类型，剩下 9 种属性都是 64 位浮点数类型；total_bedrooms 属性存在 207 个缺失值。

```
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)
```

（3） 数据集中的 ocean_proximity 属性为非数值属性，其取值情况有哪几种?

5 种，分别为<1H OCEAN，9136 个；INLAND，6551 个；NEAR OCEAN，2658 个；NEAR BAY，2290 个；ISLAND，5 个。

（4） 如何查看数据集中所有数值型属性的平均值，最大值和最小值。

使用 pandas.DataFrame.describe 方法：mean 为平均值，max 为最大值，min 为最小值。效果如图：

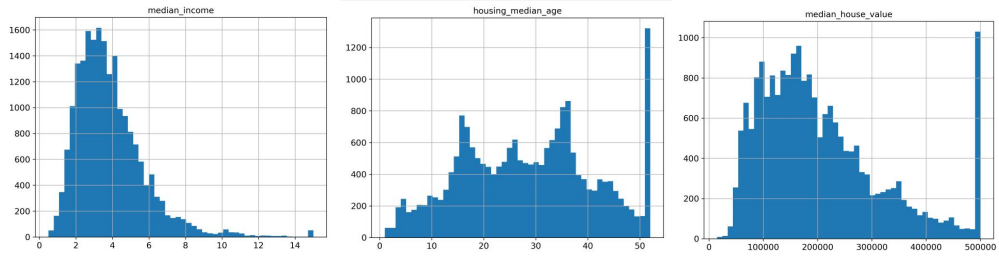
```
housing.describe()
```

	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	206855.816909
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

（5） 如何查看数据集中所有数值型属性的直方图？观察直方图，如何判断某个属性的取值被设定了上限？直方图中的“重尾”是指什么？发现重尾现象，需要进行处理吗？

使用 pandas.DataFrame.hist 方法查看直方图；

如果在边界处（最大值）有异常偏高的值（如下图）则可认为是设定了上限，median_income、housing_median_age 和 median_house_value 分别设定了 15、50 和 500000 的上限；



重尾指图形在中位数右侧的延伸比左侧要远得多；

需要进行处理，将这些属性转化为更偏向钟形的分布。

（6） 创建测试集时，通常有哪两类抽样方式？本章中哪种抽样方式更好？为什么？

随机抽样和分层抽样；分层抽样更好；可以确保在各种属性上，训练集和测试集能够更完整地代表整个数据集中各种数据的分布情况。

3、 研究数据。

（1） 我们需要对测试集中的样本进行研究吗？为什么？

不需要；测试集是为了验证模型的性能，如果对测试集进行研究，有可能因为数据透视

误差使得模型性能高于实际表现。

（2） 通过探寻属性之间的相关性，可以找出最重要的特征。有哪些函数可以用于探寻属性之间的相关性？

pandas.DataFrame.corr 方法可用于计算每对属性之间的标准相关系数，这里仅测量线性相关性（皮尔逊相关系数）；pandas.plotting.scatter_matrix 方法可绘制出每个数值属性相对于其他数值属性的相关性。

（3） 尝试增加“每个家庭的房间数量”、“每个家庭的人口数”、“卧室/房间比例”三个属性。与房价中位数相关性最高的 4 个属性分别是？

增加属性代码如下：

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"]=housing["population"]/housing["households"]
```

相关性最高的四个属性为：median_income、rooms_per_household、total_rooms、housing_median_age；

```
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
median_house_value          1.000000
median_income               0.687160
rooms_per_household         0.146285
total_rooms                 0.135097
housing_median_age          0.114110
households                  0.064506
total_bedrooms              0.047689
population_per_household   -0.021985
population                 -0.026920
longitude                  -0.047432
latitude                   -0.142724
bedrooms_per_room          -0.259984
Name: median_house_value, dtype: float64
```

4、 准备数据。

（1） 在此环节，需要对测试集中的数据做相关处理吗？

在训练验证集时不需要，在测试集上验证模型结果时需要做与验证集相同的处理。

（2） 你是如何处理缺失值的？

可以删除带有缺失值的数据行、删除含有缺失值的属性或使用中位数填充缺失值，我选择使用中位数填充缺失值。使用 SimpleImputer 进行处理，代码如下：

```
#处理缺失值，使用中位数填充
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num) #算均值，方差，最大值，最小值供transform使用
X = imputer.transform(housing_num)#在fit的基础上，进行imputer变换操作
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
                          index=housing_num.index)
```

（3） 为什么必须将 ocean_proximity 属性的文本类型转换为数值类型的?

在这个回归任务里，SVM 模型无法直观地理解文本类型所代表的含义，模型的输入应统一是数值类型。

（4） 你是如何将 ocean_proximity 属性的文本类型转换为数值类型的?

使用 sklearn.preprocessing.OneHotEncoder 方法，将 object 里的各文本值依次转换为 0/1 的二进制数值类型属性。代码如下:

```
#使用OneHotEncoder方法处理object类型
housing_cat = housing[["ocean_proximity"]]

from sklearn.preprocessing import OneHotEncoder
cat_encoder = OneHotEncoder()
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
housing_cat_1hot.toarray()
```

（5） 尝试增加 3 个新的重要特征：“每个家庭的房间数量”、“每个家庭的人口数”、“卧室/房间比例”

实现一个 CombinedAttributesAdder 类用于转换和添加属性。代码如下:

```python
#添加三个新属性
from sklearn.base import BaseEstimator, TransformerMixin
# column index
rooms_ix, bedrooms_ix, population_ix, households_ix = 3, 4, 5, 6

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_bedrooms_per_room = True): # no *args or **kargs
        self.add_bedrooms_per_room = add_bedrooms_per_room
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
        population_per_household = X[:, population_ix] / X[:, households_ix]
        if self.add_bedrooms_per_room:
            bedrooms_per_room = X[:, bedrooms_ix] / X[:, rooms_ix]
            return np.c_[X, rooms_per_household, population_per_household,
                         bedrooms_per_room]
        else:
            return np.c_[X, rooms_per_household, population_per_household]

attr_adder = CombinedAttributesAdder(add_bedrooms_per_room=False)
housing_extra_attribs = attr_adder.transform(housing.values)

housing_extra_attribs = pd.DataFrame(
    housing_extra_attribs,
    columns=list(housing.columns)+["rooms_per_household", "population_per_household"],
    index=housing.index)
```

（6） 你用到了特征缩放吗？你是如何实现特征缩放的？

用到了，使用 sklearn.preprocessing.StandardScaler 方法进行标准化。代码如下：

```python
#pipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', CombinedAttributesAdder()),
        ('std_scaler', StandardScaler()),
    ])

housing_num_tr = num_pipeline.fit_transform(housing_num)
```

（7） 尝试使用流水线，来对数据做相关处理，包括缺失值处理，文本类型转换为数值类型，特征缩放，增加新的重要属性。（这是对提交的 Python 代码的要求）

流水线代码如上图（同提交的 Python 代码）。

5、 研究模型。

（1） 训练一个 SVM（使用线性核函数）。并在训练集上评估其性能。在训练集上的

RMSE 是?

如图 111094.6308539982

```
111  #模型训练
112  from sklearn.metrics import mean_squared_error
113  from sklearn.svm import SVR
114
115  svm_reg = SVR(kernel="linear")
116  svm_reg.fit(housing_prepared, housing_labels)
117  housing_predictions = svm_reg.predict(housing_prepared)
118  svm_mse = mean_squared_error(housing_labels, housing_predictions)
119  svm_rmse = np.sqrt(svm_mse)
120  print(svm_rmse)
121
```

```
问题  输出  调试控制台  终端                    2: Python
PS F:\ai\hw1> conda activate base
PS F:\ai\hw1> & C:/Users/Harry/Anaconda3/python.exe f:/ai/hw1/main.py
111094.6308539982
```

（2） 利用 10 折交叉验证来评估其泛化性能。在验证集上的 RMSE 均值是?

如图 111809.84009600841

```
127  #交叉验证
128  from sklearn.model_selection import cross_val_score
129
130  scores = cross_val_score(svm_reg, housing_prepared, housing_labels,
131                           scoring="neg_mean_squared_error", cv=10)
132  svm_rmse_scores = np.sqrt(-scores)
133
134  print("svm_rmse CV mean = ")
135  print(svm_rmse_scores.mean())
136
```

```
问题  输出  调试控制台  终端                    2: Python
svm_rmse CV mean =
111809.84009600841
```

6、微调模型。

（1） 选择最佳超参。

a) 利用网格搜索，从以下超参组合中选取最佳超参。其中，网格搜索的配置为“cv=5, scoring='neg_mean_squared_error', verbose=2”。最后得到的最佳超参是？最佳超参时，验证集上的 RMSE 为多少?

```
param_grid = [
        {'kernel': ['linear'], 'C': [10., 30., 100., 300., 1000., 3000., 10000., 30000.0]},
        {'kernel': ['rbf'], 'C': [1.0, 3.0, 10., 30., 100., 300., 1000.0],
         'gamma': [0.01, 0.03, 0.1, 0.3, 1.0, 3.0]},
    ]
```

网格搜索得到的最佳超参是：{'C': 30000.0, 'kernel': 'linear'}
RMSE 是 70363.90313964167

b) 利用随机搜索，寻找超参。

随机搜索的配置如下：“param_distributions=param_distribs, n_iter=50, cv=5, scoring='neg_mean_squared_error', verbose=2, random_state=42) ”，其中：param_distribs = {

```
'kernel': ['linear', 'rbf'],
'C': reciprocal(20, 200000),
'gamma': expon(scale=1.0),
}
```

随机搜索得到的最佳超参是：{'C': 157055.10989448498, 'gamma': 0.26497040005002437, 'kernel': 'rbf'}
RMSE 是 54767.99053704408

(2)在测试集上评估系统。其在测试集上的 RMSE 是？（提示：利用流水线的 transform() 方法对测试集做相关处理后，再进行评估）

使用随机搜索的最佳超参数在测试集上的得到的 RMSE 是 53539.045409938415

```
#使用随机搜索的最佳参数在测试集上的rmse
params_ = {'C': 157055.10989448498, 'gamma': 0.26497040005002437, 'kernel'
svm_reg_new = SVR(**params_)
svm_reg_new.fit(housing_prepared, housing_labels)
housingt_predictions = svm_reg_new.predict(housingt_prepared)
svm_mse_final = mean_squared_error(housingt_labels, housingt_predictions)
svm_rmse_final = np.sqrt(svm_mse_final)
print("svm_rmse_final = ")
print(svm_rmse_final)
```

```
问题  输出  调试控制台  终端  2: Python

svm_rmse =
111094.6308539982
svm_rmse CV mean =
111809.84009600841
svm_rmse_final =
53539.045409938415
```