##现在开始为机器学习算法准备数据，首先把x属性与y属性（标签）分开，注意drop（）会创建一个数据副本，但是不影响strat\_train\_set

housing = strat\_train\_set.drop("median\_house\_value", axis=1)

housing\_labels = strat\_train\_set["median\_house\_value"].copy()

##数据清理：1. 将缺失的值设置为中位数 2.将分类变量转换为独热向量 3.添加一些自定义的属性 4. 特征缩放（标准化）

##1. 先导入SimpleImputer方程，为缺失值插值做准备

from sklearn.impute import SimpleImputer

##2. 先导入OneHotEncoder方程，为将分类变量转换为独热变量做准备

from sklearn.preprocessing import OneHotEncoder

##3. 定义一个转换器，增加一些在代码详解2-第一部分提到的属性，这一部分可能比较难，学会套用即可

from sklearn.base import BaseEstimator, TransformerMixin

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]

## 4. 先导入StandardScaler方程，为属性标准化做准备

from sklearn.preprocessing import StandardScaler

##定义一个流水线：首先加载方程Pipeline，然后定义对于数值变量的流水线，要按照操作的步骤，即步骤1、3、4。

from sklearn.pipeline import Pipeline

num\_pipeline = Pipeline([

('imputer', SimpleImputer(strategy="median")),

('attribs\_adder', CombinedAttributesAdder()),

('std\_scaler', StandardScaler()),

])

##加载方程ColumnTransformer，定义新的流水线，分别对数值型和分类型的变量做不同处理，数值型的用以上的流水线，而分类型的采用步骤2（即转换为独热向量）

from sklearn.compose import ColumnTransformer

num\_attribs = list(housing)

num\_attribs.remove('ocean\_proximity')

cat\_attribs = ["ocean\_proximity"]

full\_pipeline = ColumnTransformer([

("num", num\_pipeline, num\_attribs),

("cat", OneHotEncoder(), cat\_attribs),

])

##将流水线用于训练数集的x属性上，进行4个步骤的处理，将处理好的数据放在housing\_prepared数据中。注意y变量数据不进行处理，仍旧保存在housing\_labels中

housing\_prepared = full\_pipeline.fit\_transform(housing)

##选择和训练模型，分别对1）线性回归2）决策树3）随机森林进行拟合和参数估计。

##首先，采用线性回归模型，从sklearn加载LinearRegression方程

from sklearn.linear\_model import LinearRegression

##在训练数据上拟合线性回归模型

lin\_reg = LinearRegression()

lin\_reg.fit(housing\_prepared, housing\_labels)

##在训练数据上计算均方根误差RMSE

##首先加载mean\_squared\_error方程

from sklearn.metrics import mean\_squared\_error

##在训练集上对y变量做预测

housing\_predictions = lin\_reg.predict(housing\_prepared)

##对比y变量的预测值和真值，计算RMSE，注意mean\_squared\_error计算的是RMSE的平方，即均方误差MSE

lin\_mse = mean\_squared\_error(housing\_labels, housing\_predictions)

##对均方误差MSE开方计算RMSE

lin\_rmse = np.sqrt(lin\_mse)

lin\_rmse

##在训练数据上拟合决策树模型

##首先从sklearn加载DecisionTreeRegressor方程

from sklearn.tree import DecisionTreeRegressor

##在训练数据上拟合决策树模型

tree\_reg = DecisionTreeRegressor(random\_state=42)

tree\_reg.fit(housing\_prepared, housing\_labels)

##在训练数据上计算均方根误差RMSE

housing\_predictions = tree\_reg.predict(housing\_prepared)

tree\_mse = mean\_squared\_error(housing\_labels, housing\_predictions)

tree\_rmse = np.sqrt(tree\_mse)

tree\_rmse

##注意以上的预测是在训练数据上，由于模型参数的估计是用训练数据，而预测用同一套数据是in-sample error，in-sample error小的模型并不一定在测试数集上有好的表现。我们需要用k交叉验证来选择模型，评价哪个模型更好

##首先加载cross\_val\_score方程

from sklearn.model\_selection import cross\_val\_score

##计算10-折交叉验证下的10个负MSE

scores = cross\_val\_score(tree\_reg, housing\_prepared, housing\_labels,

scoring="neg\_mean\_squared\_error", cv=10)

##计算10-折交叉验证下的10个RMSE

tree\_rmse\_scores = np.sqrt(-scores)

##定义一个方程，将k-折交叉验证算出来的RMSE及RMSE的均值的标准差打印出来，用来以后做比较

def display\_scores(scores):

print("Scores:", scores)

print("Mean:", scores.mean())

print("Standard deviation:", scores.std())

display\_scores(tree\_rmse\_scores)

##计算线性回归的10-交叉验证下的RMSE

lin\_scores = cross\_val\_score(lin\_reg, housing\_prepared, housing\_labels,

scoring="neg\_mean\_squared\_error", cv=10)

lin\_rmse\_scores = np.sqrt(-lin\_scores)

display\_scores(lin\_rmse\_scores)

##按照上面的流程，再试一下随机森林

from sklearn.ensemble import RandomForestRegressor

##训练模型

forest\_reg = RandomForestRegressor(max\_features =8, n\_estimators=30, random\_state=42)

forest\_reg.fit(housing\_prepared, housing\_labels)

##在训练数据上做预测，计算RMSE

housing\_predictions = forest\_reg.predict(housing\_prepared)

forest\_mse = mean\_squared\_error(housing\_labels, housing\_predictions)

forest\_rmse = np.sqrt(forest\_mse)

forest\_rmse

##计算10-交叉验证下的RMSE

from sklearn.model\_selection import cross\_val\_score

forest\_scores = cross\_val\_score(forest\_reg, housing\_prepared, housing\_labels,

scoring="neg\_mean\_squared\_error", cv=10)

forest\_rmse\_scores = np.sqrt(-forest\_scores)

display\_scores(forest\_rmse\_scores)

##最后，我们在比较不同模型在k-折交叉验证下的误差之后，选择了随机森林。然后，在测试数据上预测Y值，并计算out-of-sample，即在训练数集上的预测误差

##首先，将训练数集的x和y属性分开

X\_test = strat\_test\_set.drop("median\_house\_value", axis=1)

y\_test = strat\_test\_set["median\_house\_value"].copy()

##将x属性进行之前的4个步骤的转换

X\_test\_prepared = full\_pipeline.transform(X\_test)

##拟合随机森林模型

final\_predictions = forest\_reg.predict(X\_test\_prepared)

##计算预测值的RMSE

final\_mse = mean\_squared\_error(y\_test, final\_predictions)

final\_rmse = np.sqrt(final\_mse)

final\_rmse