任务介绍

使用纽约州房地产数据来预测房屋的销售价格。主要包括以下几部分：

1. 数据下载、存储、读取
2. 数据分析
3. 拆分数据集
4. 数据清洗
5. 创建管道与模型训练
6. 模型评估
7. 核心代码

数据下载、存储、读取

- 设置下载地址与保存地址
- 下载tar文件并解压
- 数据读取为DataFrame

```
import os
import tarfile
import urllib.request
# download data setting
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"
```

```
# 下载tar文件并解压
def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    # stores data on local disk
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
#fetch_housing_data()
```

```
# 数据读取
import pandas as pd
import numpy as np
# load data
def load_hossing_data(path = HOUSING_PATH):
    csv_ptah = os.path.join(path,'housing.csv')
    return pd.read_csv(csv_ptah)
```

数据分析

- 数据整体情况
  - 通过housing.info()查看数据字段情况
  - 查看数据样例
  - 查看非数字字段“ocean_proximity”的情况
  - housing.describe()查看数据的分布情况

```
housing = load_hossing_data()
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	m
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	45
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	35
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	35
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	34
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	34

```
housing["ocean_proximity"].value_counts()
```

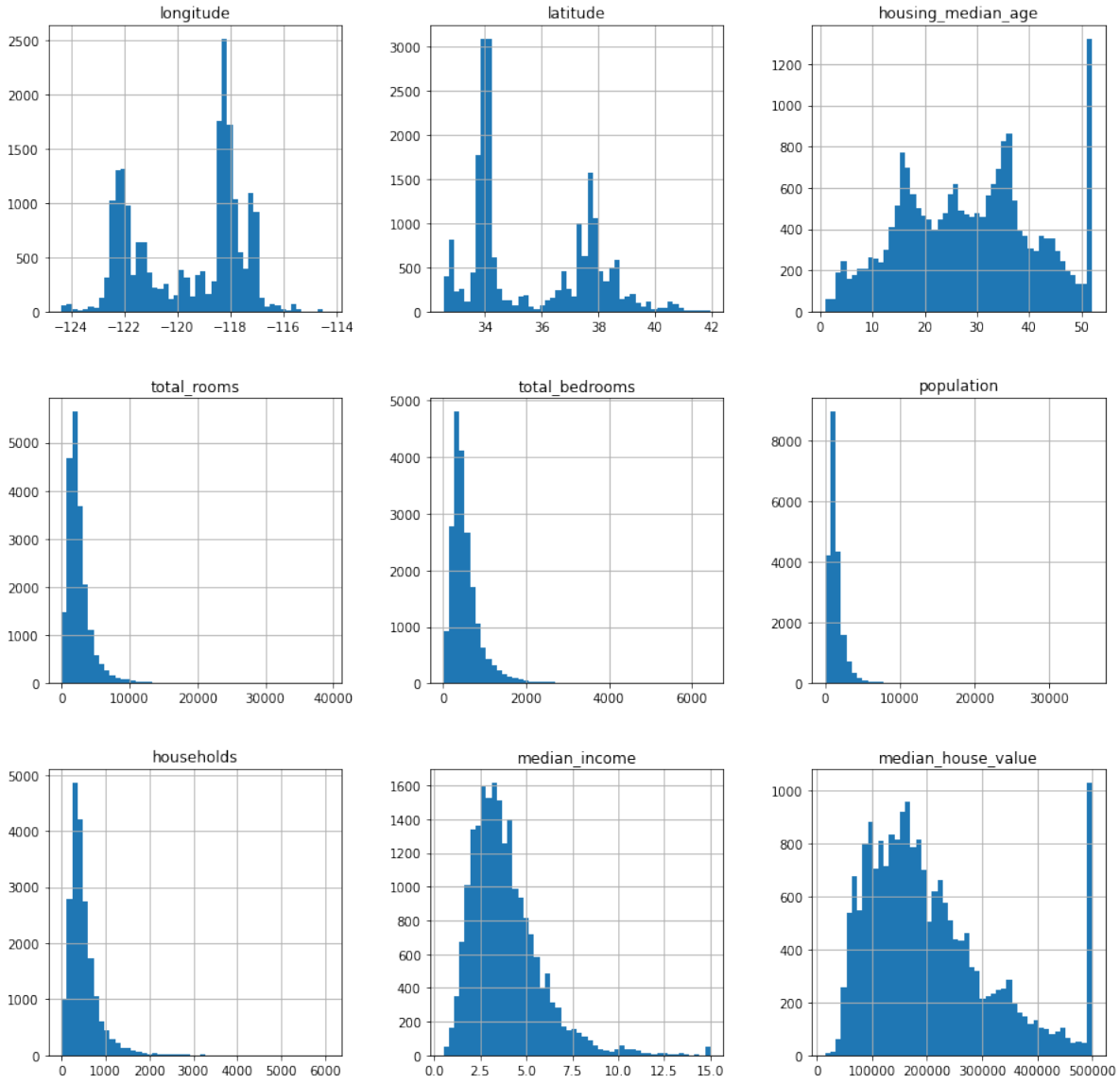
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

```
housing.describe()
```

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

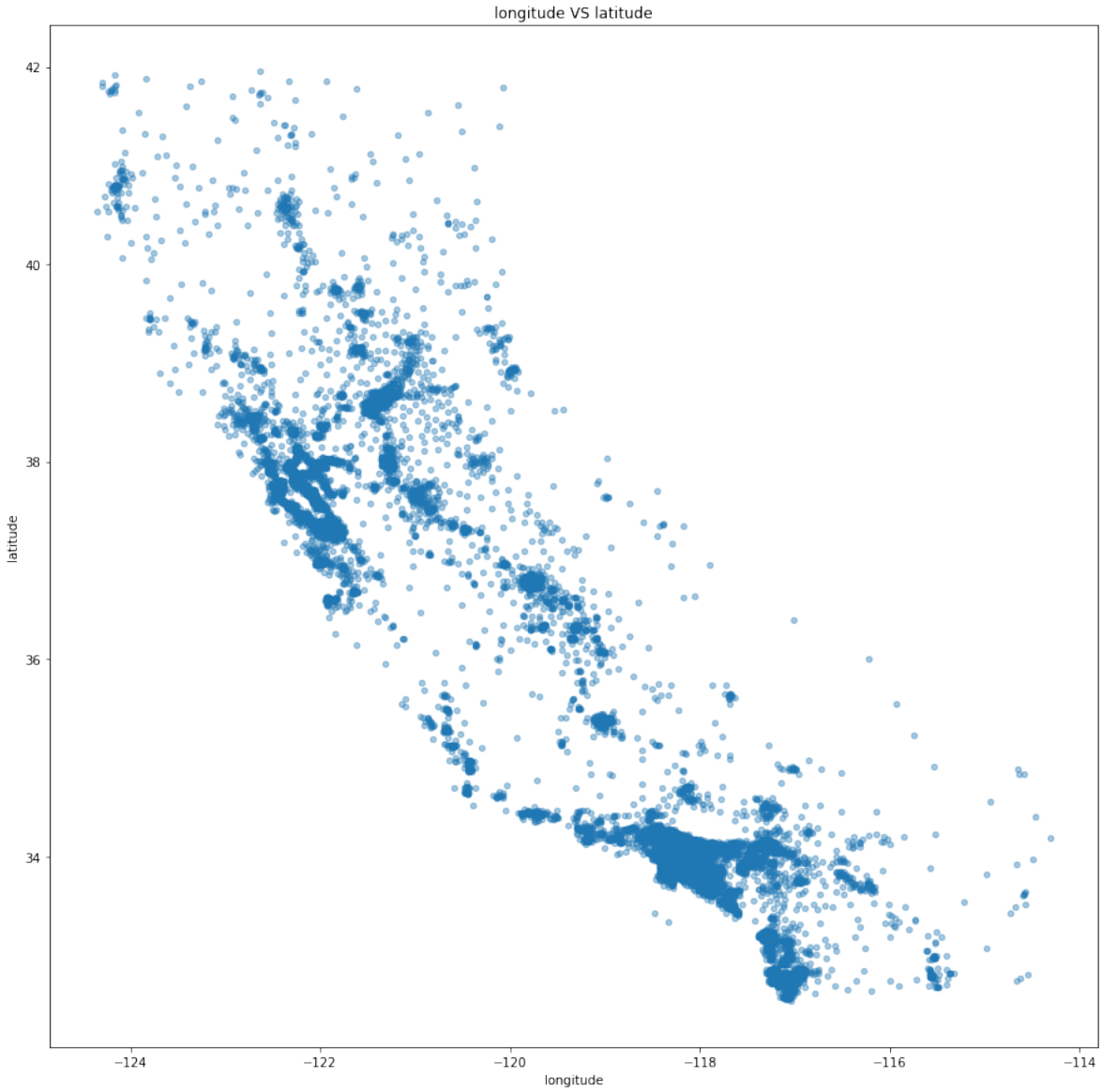
- 字段数据分布可视化

```
import matplotlib.pyplot as plt
housing.hist(bins=50,figsize=(15,15))
plt.show()
```

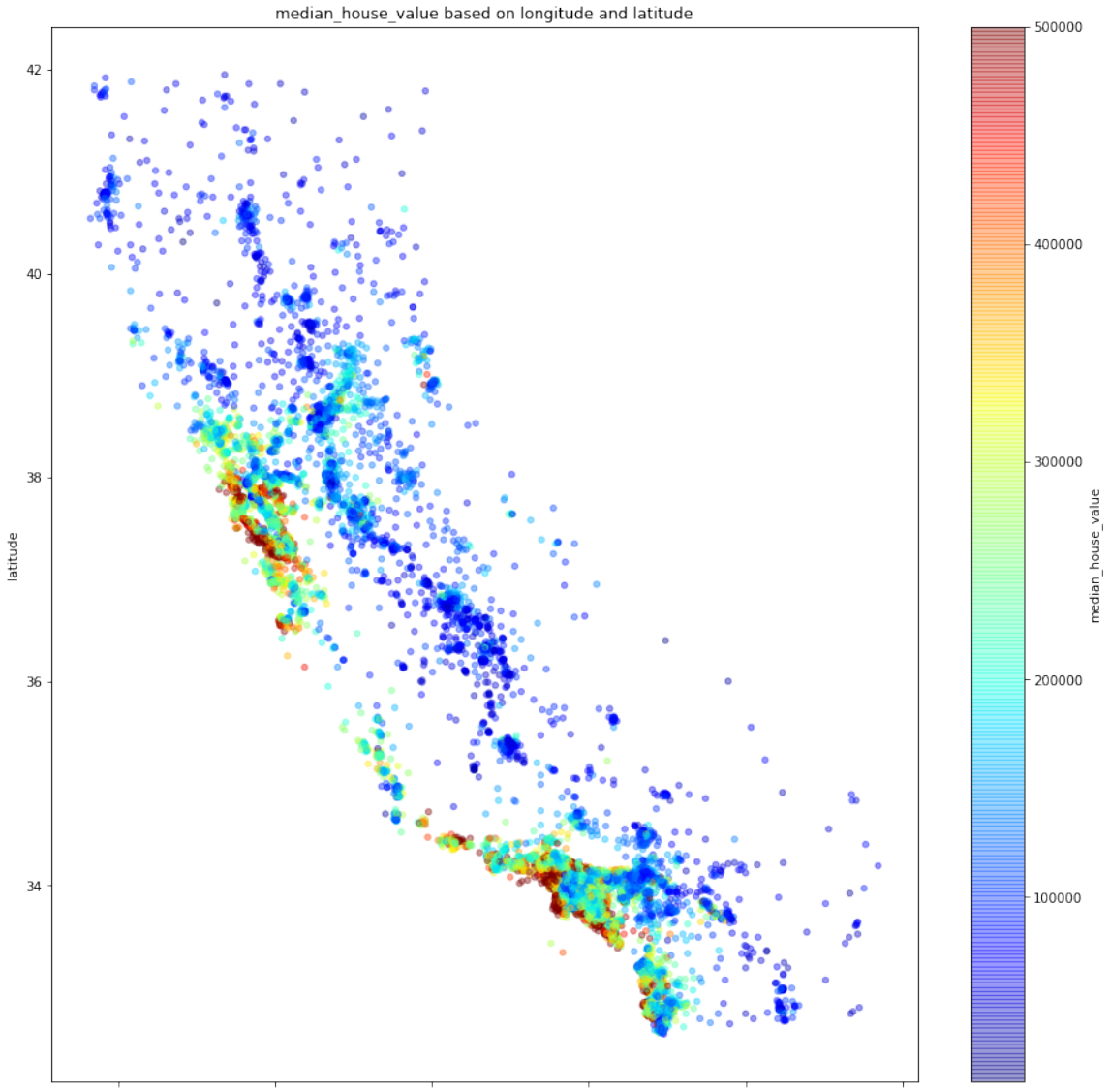


- 可视化分析
  - 查看按照经纬度数据的分布情况
  - 经纬度与房价分布情况
  - 经纬度、收入与房价分布情况
  - 主要特征之间的相关性可视化
  - 各主要特征与房价的相关性可视化

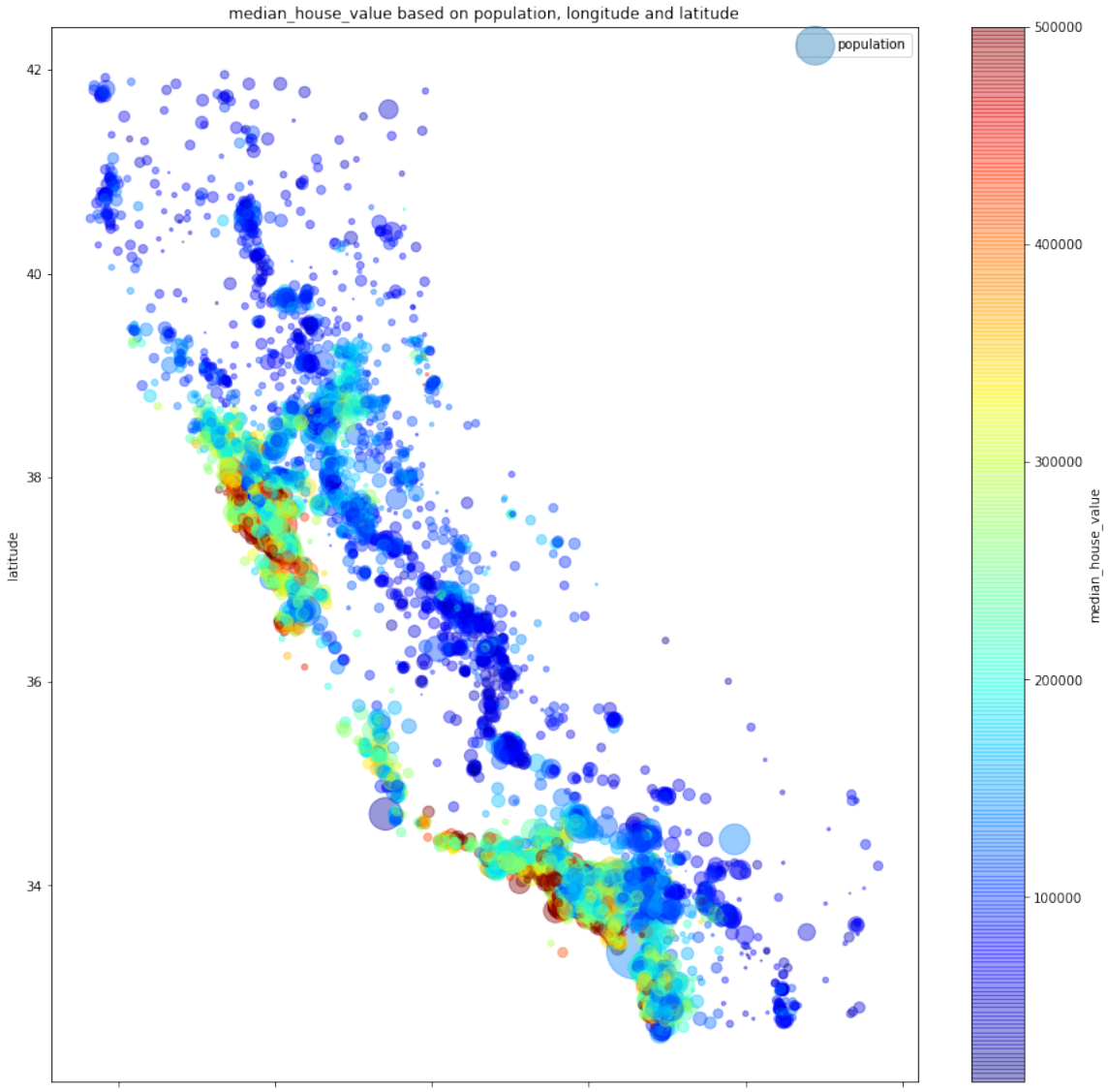
```
# 数据可视化
housing = strat_train_set.copy()
# 经纬度
housing.plot(figsize=(15,15),kind='scatter',x='longitude',y='latitude',alpha=0.4,title="longitude VS latitude")
plt.show()
```



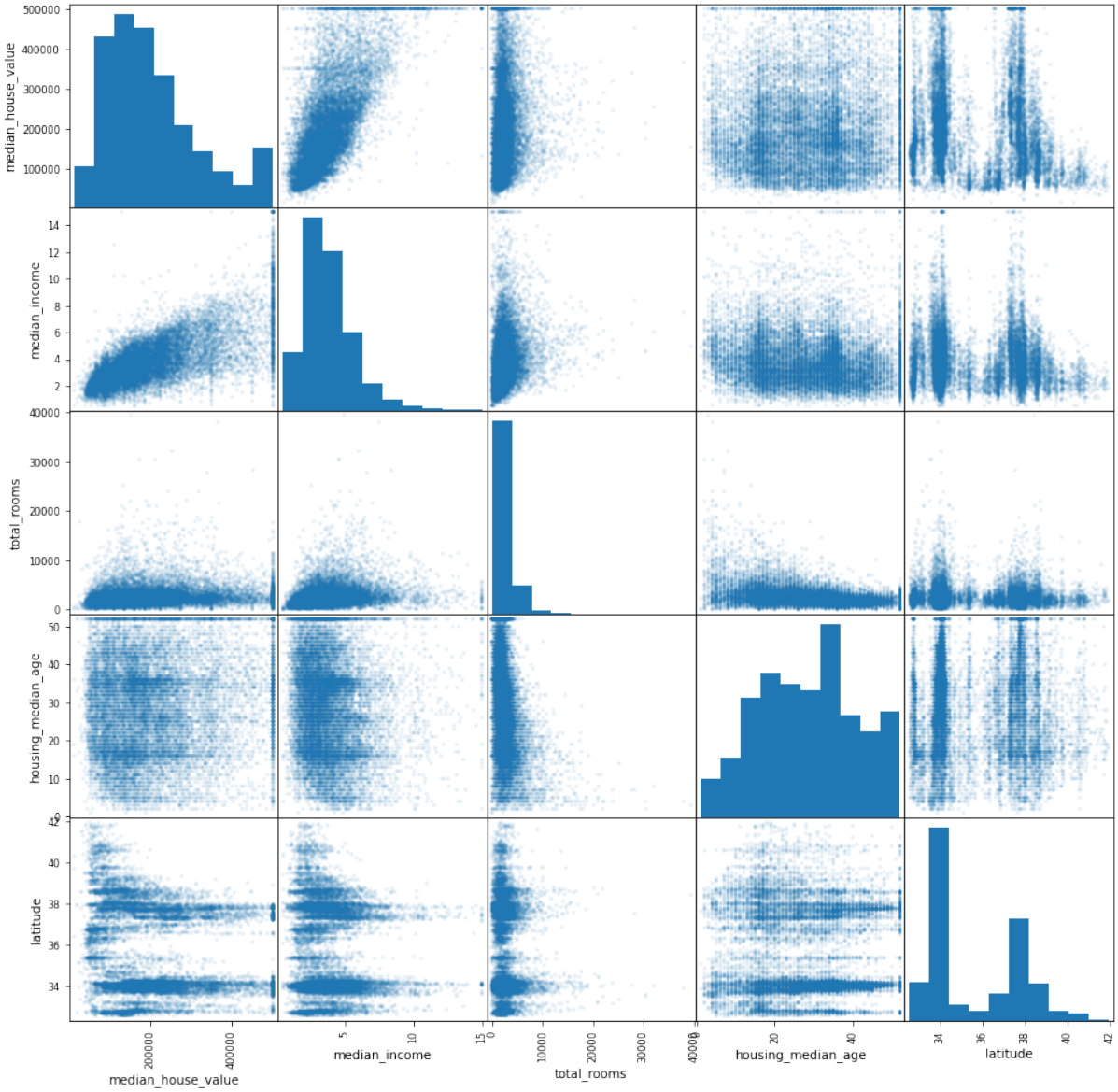

```
# 经纬度、房价
housing.plot(figsize=(15,15),kind='scatter',x='longitude',y='latitude',alpha=0.4,c = 'median_house_value',
             cmap=plt.get_cmap("jet"),colorbar = True,title="median_house_value based on longitude and latitude")
plt.show()

```




```
# 经纬度、收入、房价
housing.plot(figsize=(15,15),kind='scatter',x='longitude',y='latitude',alpha=0.4,
             s=housing['population']/20,label='population', c = 'median_house_value',cmap=plt.get_cmap("jet"),colorbar = True,
             title="median_house_value based on population, longitude and latitude")
plt.show()
```



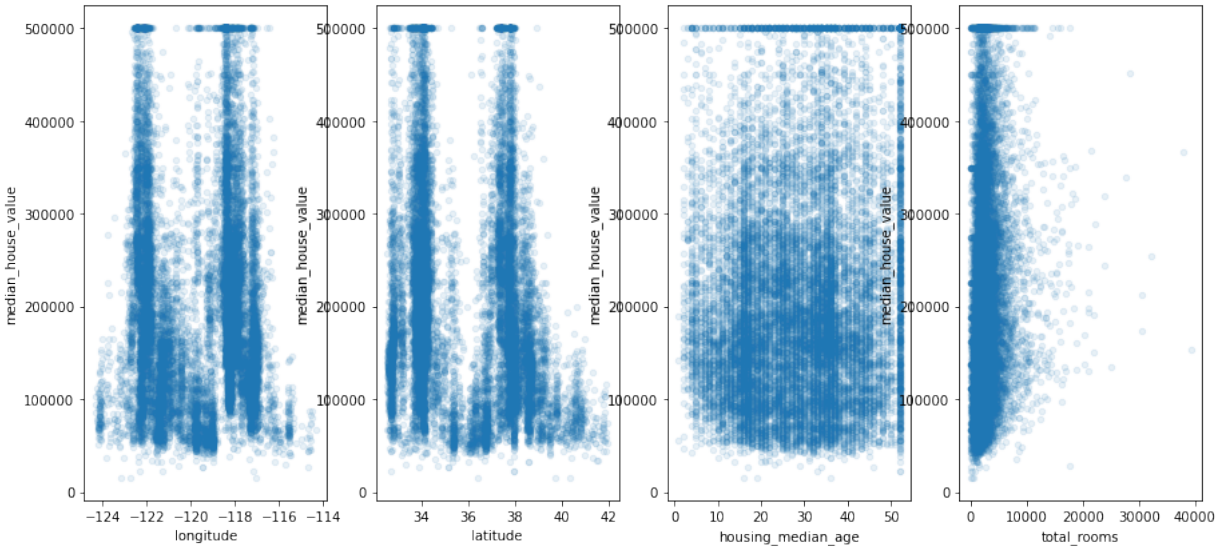

```
# 主要特征之间的相关性
from pandas.plotting import scatter_matrix
attributes = ["median_house_value","median_income","total_rooms","housing_median_age","latitude"]
scatter_matrix(housing[attributes],figsize=(15,15),alpha=0.1)
plt.show()

```

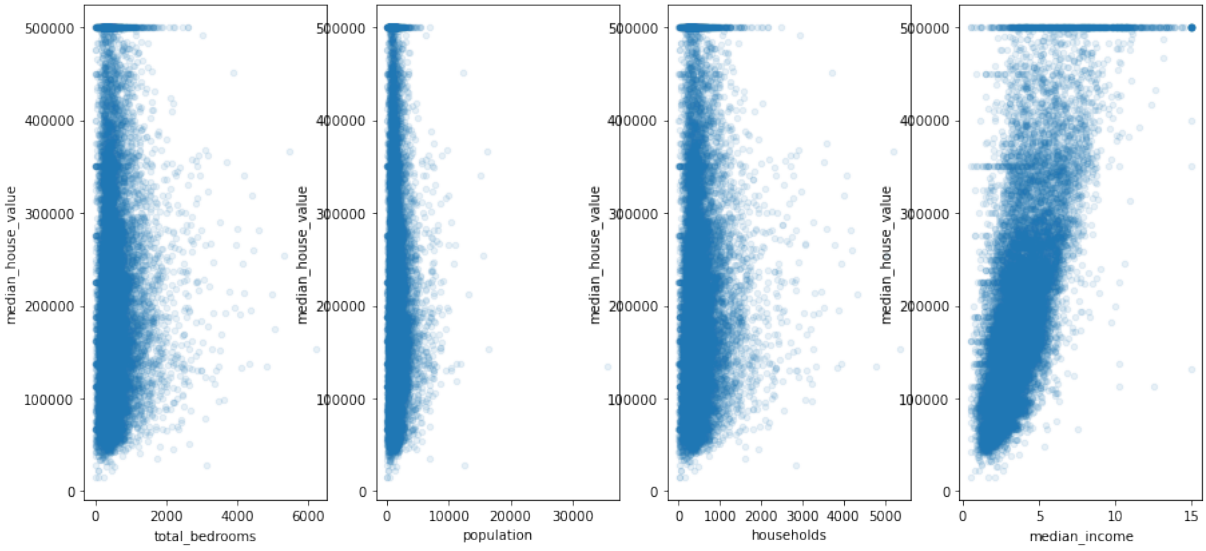



```
# 计算相关系数
corr_matrix = housing.corr()
corr_matrix['median_house_value'].sort_values(ascending=False)
```

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

```
# 主要特征与房价的相关性
fig,axes=plt.subplots(2,4,figsize=(15,15))
axs = axes.ravel()
colums =list(housing.columns)
colums.remove("median_house_value")
colums.remove("ocean_proximity")
for index in range(len(colums)):
    housing.plot(figsize=(15,15),kind='scatter',x=colums[index],y='median_house_value',alpha=0.1,ax=axs[index])
plt.show()

```

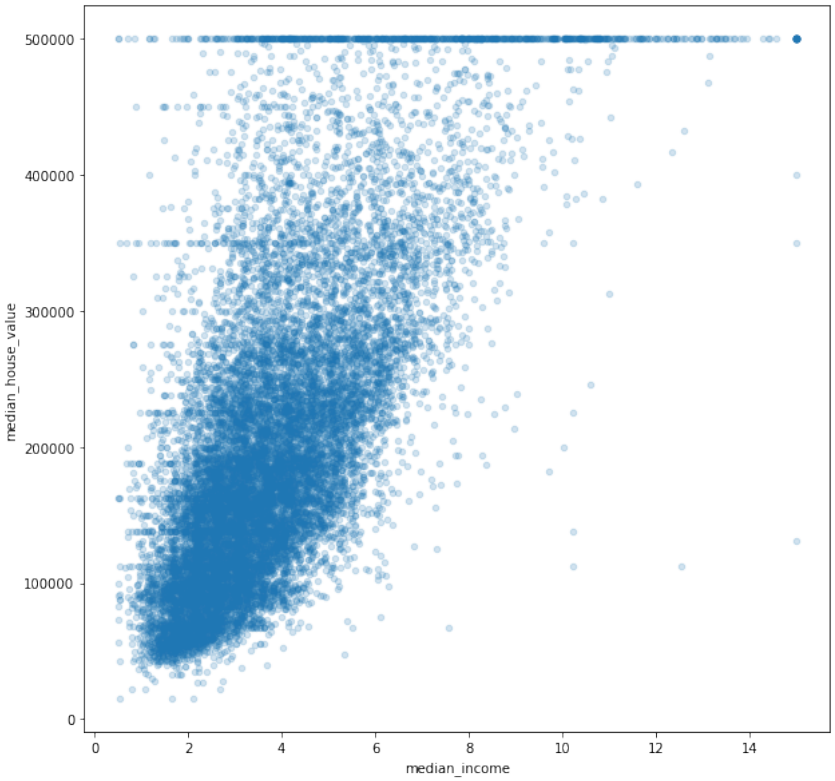





```
colums
```

```
['longitude',
 'latitude',
 'housing_median_age',
 'total_rooms',
 'total_bedrooms',
 'population',
 'households',
 'median_income']
```

```
# 收入与房价的相关性
housing.plot(figsize=(10,10),kind='scatter',x='median_income',y='median_house_value',alpha=0.2)
plt.show()
```



数据拆分

- 生成收入等级
  - 将收入进行缩放，并取整分级
  - 增加一列income_cat临时存储收入的分级
- 数据拆分
  - 使用StratifiedShuffleSplit函数，构建随机拆分器
  - 根据收入等级来将数据分组，在每组随机抽样
  - 将数据分成测试集和验证集
- 删除临时保存的收入等级数据

```
# 分层抽样
from sklearn.model_selection import StratifiedShuffleSplit
housing['income_cat'] = np.ceil(housing['median_income']/1.5)
housing['income_cat'].where(housing['income_cat']<5,5.0,inplace=True)
split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index,test_index in split.split(housing,housing['income_cat']):
    strat_train_set = housing.iloc[train_index]
    strat_test_set = housing.iloc[test_index]
strat_train_set.drop(["income_cat"],axis =1, inplace = True)
strat_test_set.drop(["income_cat"],axis =1, inplace = True)

```

数据清洗

- 数值型数据
  - 缺失数据填充
    - 使用SimpleImputer，通过中位数策略进行填充
  - 补充特征，添加户型平均室数和人均房屋数
  - 使用标准差对数据进行缩放
- 文本型数据
  - 独热编码进行数字化

```
# 中位数补充缺失值
housing = strat_train_set.drop("median_house_value", axis=1) # drop labels for training set
housing_labels = strat_train_set["median_house_value"].copy()
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
housing_num = housing.drop("ocean_proximity",axis=1)
imputer.fit(housing_num)
housing_num.median()
X = imputer.transform(housing_num)
X.shape

```

```
(16512, 8)
```

```
# 查看补充后数据
housing_tr = pd.DataFrame(X,columns=housing_num.columns,index=housing_num.index)
# columns=housing_num.columns
housing_tr.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
17606	-121.89	37.29	38.0	1568.0	351.0	710.0	339.0	2.7042
18632	-121.93	37.05	14.0	679.0	108.0	306.0	113.0	6.4214
14650	-117.20	32.77	31.0	1952.0	471.0	936.0	462.0	2.8621
3230	-119.61	36.31	25.0	1847.0	371.0	1460.0	353.0	1.8839
3555	-118.59	34.23	17.0	6592.0	1525.0	4459.0	1463.0	3.0347

```
# 处理文本数据：oneHot_encoder
housing_cat = housing[['ocean_proximity']]
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
# encoder = LabelEncoder()
# housing_cat_encoded = encoder.fit_transform(housing_cat)
# # housing_cat_encoded
oneHot_encoder = OneHotEncoder()
housing_cat_1hot = oneHot_encoder.fit_transform(housing_cat)
housing_cat_1hot.shape
# housing_cat_1hot[1].toarray()
# from sklearn.processing import La

```

```
array([[1., 0., 0., 0.]])
```

```
# 处理文本数据：LabelBinarizer
from sklearn.preprocessing import LabelBinarizer
encoder = LabelBinarizer(sparse_output=True)
housing_cat_1hot = encoder.fit_transform(housing_cat)
housing_cat_1hot
```

```
<16512x5 sparse matrix of type '<class 'numpy.int64'>'
  with 16512 stored elements in Compressed Sparse Row format>
```

```
# 自定义转换器
## 将Dataframe 转换为Nmumpy数组
from sklearn.base import BaseEstimator, TransformerMixin
class DataFrameSelector(BaseEstimator,TransformerMixin):
    def __init__(self,attributes):
        self.attributes = attributes
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        return X[self.attributes].values
```

```
# 添加新的特征
from sklearn.preprocessing import FunctionTransformer
rooms_ix, bedrooms_ix, population_ix, household_ix = [
    list(housing.columns).index(col)
    for col in ("total_rooms", "total_bedrooms", "population", "households")]

def add_extra_features(X, add_bedrooms_per_room=True):
    rooms_per_household = X[:, rooms_ix] / X[:, household_ix]
    population_per_household = X[:, population_ix] / X[:, household_ix]
    if 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 = FunctionTransformer(add_extra_features, validate=False,
                                 kw_args={"add_bedrooms_per_room": False})
housing_extra_attribs = attr_adder.fit_transform(housing.values)
housing_extra_attribs = pd.DataFrame(
    housing_extra_attribs,
    columns=list(housing.columns)+["rooms_per_household", "population_per_household"])
housing_extra_attribs.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
0	-121.89	37.29	38	1568	351	710	339	2.7042	<1H OCEAN
1	-121.93	37.05	14	679	108	306	113	6.4214	<1H OCEAN
2	-117.2	32.77	31	1952	471	936	462	2.8621	NEAR OCEAN
3	-119.61	36.31	25	1847	371	1460	353	1.8839	INLAND
4	-118.59	34.23	17	6592	1525	4459	1463	3.0347	<1H OCEAN

创建管道与模型训练

通过管道方式对数据清洗和训练过程进行封装

- 数值类特征管道
- 文本类特征管道
- 合并管道
- 训练管道
- 模型选择
  - 线性回归
  - 决策树

```
# 数据清洗管道-数字类
from sklearn.pipeline import FeatureUnion,Pipeline
from sklearn.preprocessing import StandardScaler
# 增加两列，9-》11
num_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ("attribs_adder",FunctionTransformer(add_extra_features,validate=False)),
    ('stander',StandardScaler())
])
#数据清洗管道-文本类
#ColumnTransformer vs FeatureUnion
# https://blog.csdn.net/fendouaini/article/details/109211421
from sklearn.compose import ColumnTransformer
num_attribs = list(housing_num.columns)
# 从一列，变为5列
```

```
cat_attribs = ["ocean_proximity"]
full_pipelie = ColumnTransformer([
    ('num',num_pipeline,num_attribs),
    ('cat',OneHotEncoder(),cat_attribs)
])
```

```
housing_prepared = full_pipelie.fit_transform(housing)
housing_prepared.shape
```

```
(16512, 16)
```

```
# 搭建训练管道
from sklearn.linear_model import LinearRegression,
from sklearn.metrics import mean_squared_error
def train_model(model):
    train_pipeline = Pipeline([
    ('datas',full_pipelie),
    ('model',model)
    ])
    return train_pipeline.fit(housing, housing_labels)

```

```
# 模型选择-线性回归
lr_model = train_model(LinearRegression())
```

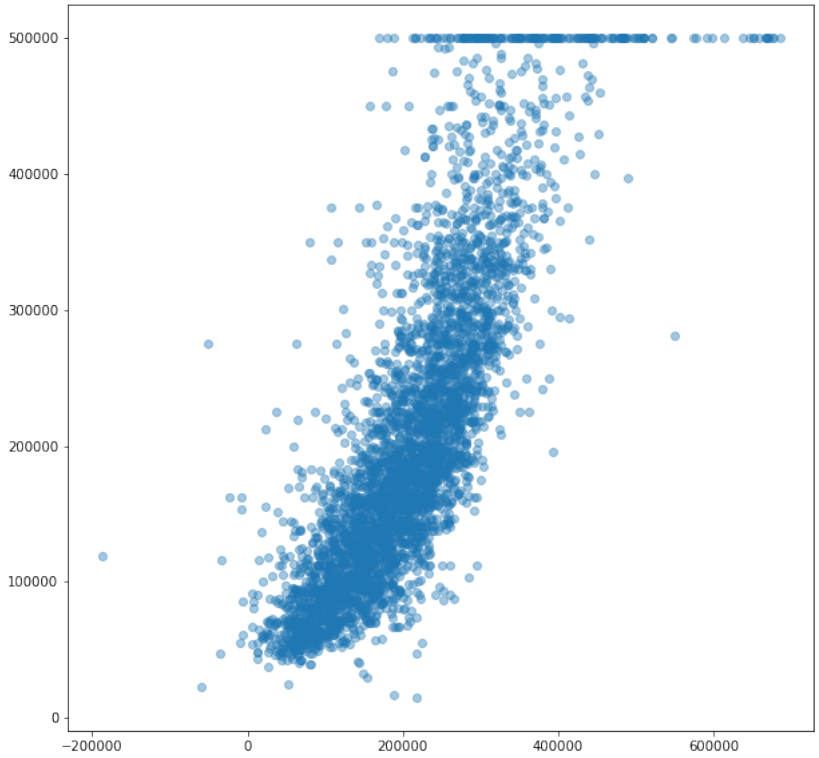
```
# 模型选择-决策树
from sklearn.tree import DecisionTreeRegressor
dt_model = train_model(DecisionTreeRegressor())

```

模型评估

- 预测数据分布
- 标准差
- 模型保存与加载

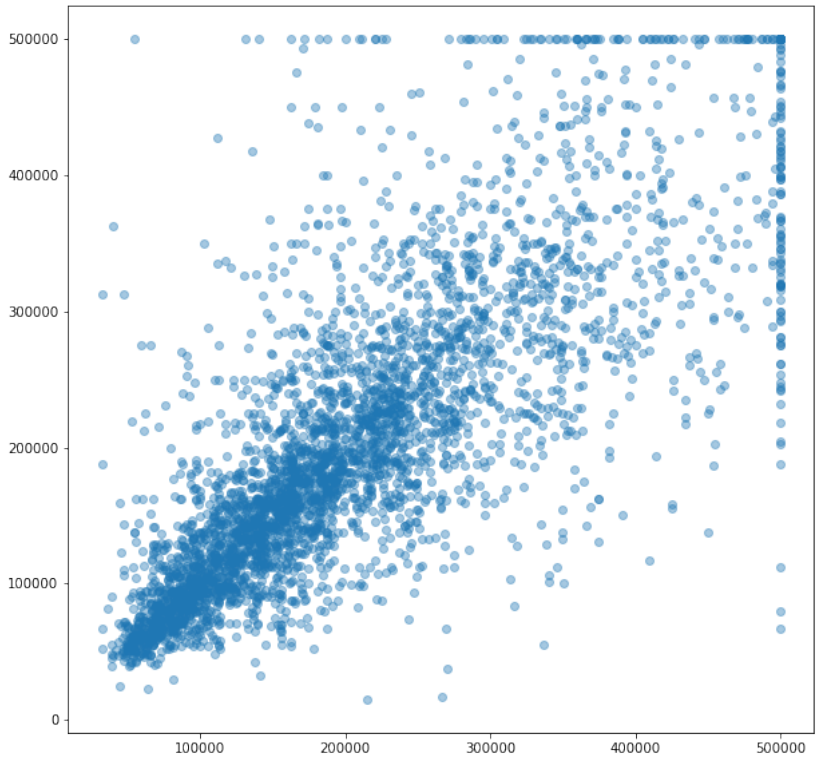
```
# 画出预测值与真实值的分布情况，返回方差
test_data = strat_test_set.drop("median_house_value", axis=1) # drop labels for training set
test_labels = strat_test_set["median_house_value"].copy()
def test(model):
    test_data = strat_test_set.drop("median_house_value", axis=1) # drop labels for training set
    test_labels = strat_test_set["median_house_value"].copy()
    predictions = model.predict(test_data)
    plt.figure(1,figsize=(10,10))
    plt.scatter(x=predictions,y=test_labels,alpha=0.4)
    plt.xlabel = 'predictions'
    plt.ylabel = 'ground truth'
    plt.show()
    mse = mean_squared_error(test_labels,predictions)
    return np.sqrt(mse)
```

```
test(lr_model)

```




```
66911.98070857547
```

```
test(dt_model)
```

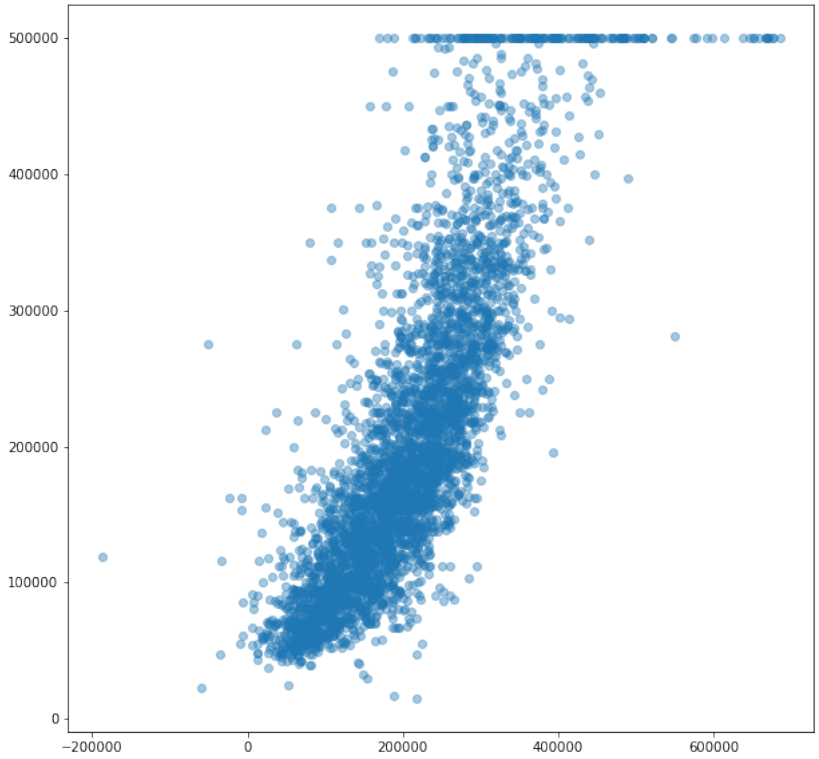


```
70537.35639423419
```

```
# 模型保存
import joblib
joblib.dump(lr_model,'lr_model.pkl')
```

```
['lr_model.pkl']
```

```
# 模型加载
lr = joblib.load('lr_model.pkl')
test(lr)
```




```
66911.98070857547
```

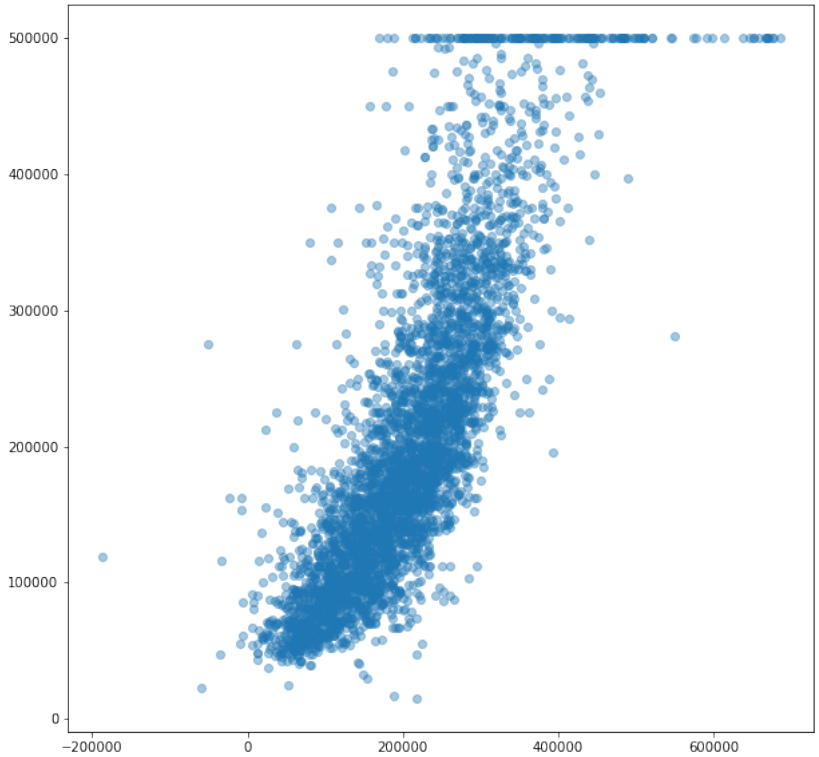
核心代码

```
import os
import tarfile
import urllib.request
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder,FunctionTransformer,StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# download data setting
```

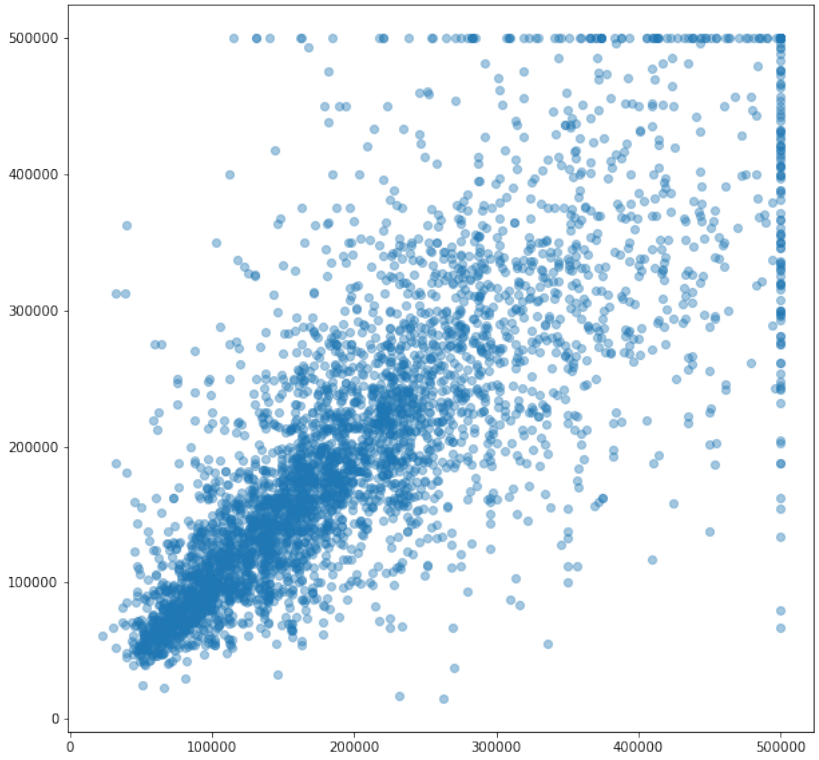
```
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

# 下载tar文件并解压
def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    # stores data on local disk
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()

# load data
def load_hossing_data(path = HOUSING_PATH):
    csv_ptah = os.path.join(path,'housing.csv')
    return pd.read_csv(csv_ptah)


#fetch_housing_data()
housing = load_hossing_data()

# 数据拆分
from sklearn.model_selection import StratifiedShuffleSplit
housing['income_cat'] = np.ceil(housing['median_income']/1.5)
housing['income_cat'].where(housing['income_cat']<5,5.0,inplace=True)
split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index,test_index in split.split(housing,housing['income_cat']):
    strat_train_set = housing.iloc[train_index]
    strat_test_set = housing.iloc[test_index]
strat_train_set.drop(["income_cat"],axis =1, inplace = True)
strat_test_set.drop(["income_cat"],axis =1, inplace = True)

# 准备管道数据
housing_train = strat_train_set.drop("median_house_value", axis=1) # drop labels for training set
housing_labels = strat_train_set["median_house_value"].copy()
housing_num = housing_train.drop("ocean_proximity",axis=1)

# 添加新的特征
rooms_ix, bedrooms_ix, population_ix, household_ix = [
    list(housing.columns).index(col)
    for col in ("total_rooms", "total_bedrooms", "population", "households")]

def add_extra_features(X, add_bedrooms_per_room=True):
    rooms_per_household = X[:, rooms_ix] / X[:, household_ix]
    population_per_household = X[:, population_ix] / X[:, household_ix]
    if 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]
# 构造数值特征管道
num_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ("attribs_adder",FunctionTransformer(add_extra_features,validate=False)),
    ('stander',StandardScaler())
])

# 数值类特征
num_attribs = list(housing_num.columns)
# 从一列，变为5列
cat_attribs = ["ocean_proximity"]

# 所有特征管道
full_pipelie = ColumnTransformer([
    ('num',num_pipeline,num_attribs),
    ('cat',OneHotEncoder(),cat_attribs)
])

# 搭建训练管道
def train_model(model):
    train_pipeline = Pipeline([
    ('datas',full_pipelie),
    ('model',model)
    ])
    return train_pipeline.fit(housing_train, housing_labels)

```

```python
# 模型选择-线性回归
lr_model = train_model(LinearRegression())
# 模型选择-决策树
dt_model = train_model(DecisionTreeRegressor())

# 画出预测值与真实值的分布情况，返回方差
test_data = strat_test_set.drop("median_house_value", axis=1) # drop labels for training set
test_labels = strat_test_set["median_house_value"].copy()
def test(model):
    test_data = strat_test_set.drop("median_house_value", axis=1) # drop labels for training set
    test_labels = strat_test_set["median_house_value"].copy()
    predictions = model.predict(test_data)
    plt.figure(1,figsize=(10,10))
    plt.scatter(x=predictions,y=test_labels,alpha=0.4)
    plt.xlabel = 'predictions'
    plt.ylabel = 'ground truth'
    plt.show()
    mse = mean_squared_error(test_labels,predictions)
    return np.sqrt(mse)
print('线性回归标准差: '+str(test(lr_model)))
print('决策树标注差: '+str(test(dt_model)))

# 模型保存
import joblib
joblib.dump(lr_model,'lr_model.pkl')
# 模型加载
# lr = joblib.load('lr_model.pkl')
# test(lr)
```

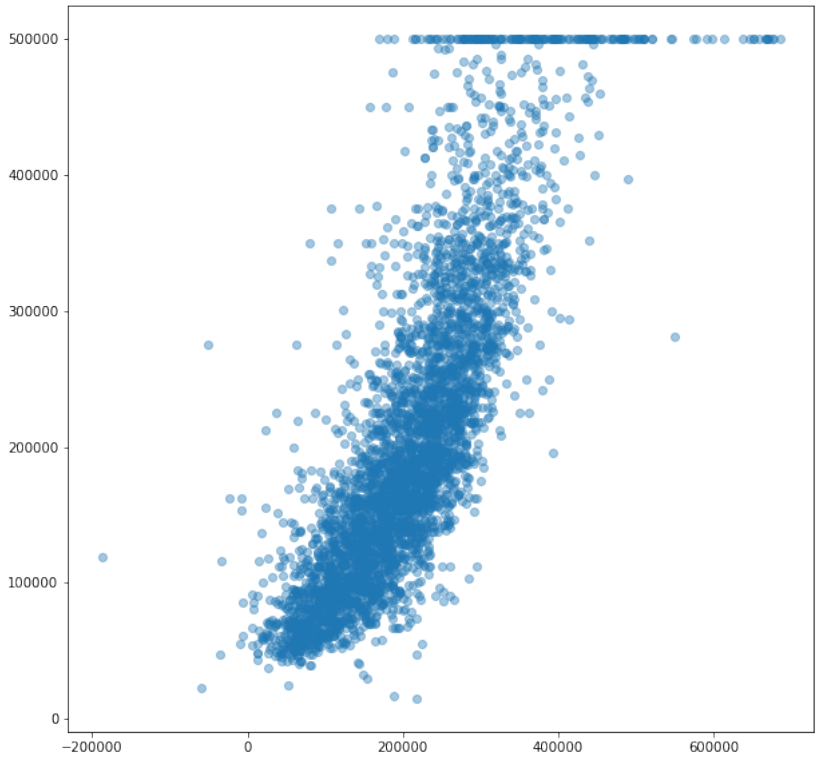
```
/root/anaconda3/envs/ai/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```




```
线性回归标准差: 66911.98070857547
```




```
决策树标注差：70375.68286769741
```




```
66911.98070857547
```