一、 实验目的

- 进一步巩固课程所学的有关**数据理解、数据分析、数据处理**的知识，为之后其它任务的开展（如**回归预测**等）奠定坚实基础；
- 进一步深化对课程所学的回归预测方法的理解，并对各种回归预测方法（如**决策树、随机森林、梯度提升机**等）的表现进行评价及分析；
- 进一步掌握 Python 语言、PyCharm 开发环境的用法，熟悉常用 Python 数据处理包 numpy、pandas 以及经典机器学习包 sklearn 的主要使用方法，完成一个数据分析、数据处理、数据挖掘、数据预测的实战案例。

二、 实验内容分析

（一）实验要求

- 给定一个房屋交易市场交易记录，其中训练集数据给出了交易价格，而测试集数据的交易价格则没有给出，设计一个高效、解释性强的算法对测试集的房屋交易价格进行预测；
- 全部编程实现，统一使用 numpy、pandas、sklearn 库，开发环境参考 requirements.txt。

（二）实验设计思路及设计方案

1.数据分析与数据预处理

首先本次实验允许使用经典机器学习库 sklearn，该库包含了许多可以用于回归预测的方法，我们可以直接调用，所以显然**影响预测准确度的关键之处就在于数据预处理**。

该数据集一共有 **20 个特征属性、1 个标签**（即**待预测的房价**），在 20 个特征值属性中，有 **16 个离散型（类别型）属性、4 个连续型（数值型）属性**。无论是离散型属性还是连续型属性，都有缺失（异常）值情况的出现，而且这些缺失（异常）值的情况并不占少数。例如，对于“土地平方英尺”属性，训练集总共 45000 条记录中一共有 13666 条为“-”，还有 6615 条为 0（显然土地平方英尺为 0 是不合理的），两者加起来共有超过 2 万条缺失（异常）值。因此，如果仿照本学期实验二的做法简单粗暴地删除掉缺失（异常）值，**将会导致近乎一半的训练集数据损失！**这种做法明显是不合理的。那么，我们应该怎么处理呢？

对于类别型属性，它们不同于数值型属性，属性的不同取值仅仅体现种类的差异，属性值（经过编号后）的绝对大小比较并没有意义（例如，我们无法根据 2 大于 1 这一大小关系，分析出编号为 1 的类别与编号为 2 的类别之间的某种**偏序关系**）。受上述类别型属性特点的启发，我们可以**直接把数据缺失的情况视为一个单独的类别**。也就是说，**赋予“数据缺失”这种情况一个新的类别**。

由于绝大部分类别型属性的取值都是字符串，难以作为模型输入，因此需要对类别型属性的不同取值进行编码。严格来说，上述过程最好采用**独热编码**的方式。假设 A、B、C 三种类别分别被编号为 1、2、4，那么**我们实际上是默认了 A、B 之间的距离要比 B、C 近，即 A、B 的相关程度更大**，人为引入这一前提假设，显然有失公允。如果将它们以独热编码的方式分别编码为(1, 0, 0, 0)、(0, 1, 0, 0)、(0, 0, 0, 1)，则不会有上述问题，因为这三种取值两两之间的**欧式距离都相等**，我们做到了没有偏袒任何一方。但是很遗憾，由于所给数据集的类别取值过多，如果全部使用独热编码，将会导致数据**过于稀疏**并**严重增加特征维度**，

使得数据处理、模型训练、模型预测的时间过长。为了**兼顾程序运行时长**，我最终还是没有使用独热编码的方式，改为对类别型属性进行从 0 开始依次递增 1 的编码。

对于数值型属性，同样存在有缺失（异常）值的情况。缺失值就是数值型属性的取值没有给出，异常值就是数值型属性的取值为 0（无论是土地平方英尺还是修建年份，取值为 0 都是没有意义的，因此**可以把 0 认为是异常值**）。我们不能像类别型属性那样将缺失（异常）值认为是一种新的类别，那应该怎么办呢？我设计了以下两种解决方案：①计算**其它取值正常的数据的平均值**，直接将缺失（异常）值**填充为该平均值**；②将全体**类别型属性**视为**特征**，将存在缺失（异常）值的**数值型属性**视为**标签**，提前使用预测模型进行回归分析，**用预测值进行填充**。

上述第①种解决方案操作简便，程序运行时间也相对更短，但是由于所有缺失（异常）值都被**填充为同一种值**，数值型属性取值的**差异性被大大削减，相当于人为增加噪声**，不利于让模型学到更多信息；上述第②种解决方案虽然让填充值**更具有差异性**，但是如果**其它特征变量与存在缺失（异常）值的变量之间的相关性较弱**，那么预测出的结果将无意义，反而可能起到误导作用。由此可见，它们**各有优劣之处**，只能通过实验结果来判定到底哪一种方案更好。**我在编程时两种方案都有实现**。

值得一提的是，对于数值型属性“出售日期”，观察可知它在训练集和测试集中都没有缺失（异常）值。因此，我在处理时直接将每个出售日期的字符串**转换为其对应的时间戳**，用时间戳的大小来刻画不同出售日期之前的取值差异。

另外，观察可知，**无论是在训练集还是在测试集中，类别型属性“地役权”的取值都为空**。也就是说，全体样本在该特征上的取值都**缺失**了，因此可以直接在训练集、测试集中**删除这一属性**。

最后，为了**避免训练集的类别变量与测试集的类别变量编码不一致**，需要先将两种数据集**合并**，进行统一预处理、使用相同方式编码后再按照索引重新**分割**为训练集、测试集，以防止预测模型的**正确性**受到影响。

2.模型建立与测试集预测

完成好数据预处理后，建立模型进行预测也就不难了，因为 sklearn 库中有大量现成的机器学习方法可以直接使用。本次实验我所使用的回归预测模型主要有**决策树、随机森林、极端随机树、梯度提升机**等。对预处理后的数据直接调用对应函数即可进行预测。

为了尽可能地获取最佳预测表现，并充分比对不同的数据预处理方式、不同的预测模型之间的差异，我总共提交了 30 次预测结果。每一次预测结果所对应的数据处理方式、预测模型如下表所示。（具体也可参考 predict_results 目录下的“不同提交结果对应的模型方法.csv”文件）

提交序号	数值型属性缺失（异常）值处理方式	所使用的回归预测模型	模型建立方法
1	填充平均值	决策树	DecisionTreeRegressor()
2	填充平均值	决策树+提前剪枝	DecisionTreeRegressor(max_depth=20)
3	填充平均值	决策树+提	DecisionTreeRegressor(max_depth=30)

		前剪枝	
4	填充平均值	随机森林	RandomForestRegressor(n_estimators=100)
5	填充平均值	随机森林	RandomForestRegressor(n_estimators=500)
6	填充平均值	随机森林	RandomForestRegressor(n_estimators=600)
7	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=100, n_jobs=-1)
8	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, n_jaobs=-1)
9	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1)
10	去除所有数值型属性，只留下类别型	决策树	DecisionTreeRegressor()
11	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=300, bootstrap=True, n_jobs=-1)
12	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=600, bootstrap=True, n_jobs=-1)
13	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=800, bootstrap=True, n_jobs=-1)
14	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1, max_depth=30)
15	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=400, bootstrap=True, n_jobs=-1)
16	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=700, bootstrap=True, n_jobs=-1)
17	预测缺失(异常)值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1)
18	预测缺失(异常)值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1, max_depth=30)
19	预测缺失(异常)值	极端随机树	ExtraTreesRegressor(n_estimators=600, bootstrap=True, n_jobs=-1)
20	预测缺失(异常)值	极端随机树	ExtraTreesRegressor(n_estimators=700, bootstrap=True, n_jobs=-1)
21	预测缺失(异常)值	梯度提升机	GradientBoostingRegressor(n_estimators=500)
22	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=500)
23	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=400)
24	填充平均值	梯度提升	GradientBoostingRegressor(n_estimator

		机	s=600)
25	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=700)
26	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=800)
27	预测缺失(异常)值	梯度提升机	GradientBoostingRegressor(n_estimators=400)
28	预测缺失(异常)值	梯度提升机	GradientBoostingRegressor(n_estimators=600)
29	预测缺失(异常)值	梯度提升机	GradientBoostingRegressor(n_estimators=700)
30	预测缺失(异常)值	梯度提升机	GradientBoostingRegressor(n_estimators=550)

这 30 个提交结果对应的预测误差存放在 predict_results **目录**下的“**不同提交结果对应的误差情况.csv**”文件中。其中，序号为 27 的提交结果**均方误差**为 **4982065.205**，在全部 30 个提交结果中最小。由于本次实验以**均方误差**为预测评价指标，因此我最终提交的是序号为 27 的版本（根目录下的 model.py 文件、预测结果 submit.csv 以及数据预处理结果 total_data_processed.csv 均对应该版本的提交）。

在最新一次公布的大作业提交情况中（截至 2022 年 11 月 5 日），我所提交版本的误差表现（以 RMSE 为评价指标）在全体同学当中排名第 3。

对于不同提交结果相应的预测误差，以及分析得到的**实验结论**，将会在**实验结果**部分进一步讨论。

三、 部分文件说明

- origin_model
  - origin_model.py：将数值型属性缺失（异常）值**填充为平均值**的代码
- predict_results
  - submit_1.csv
  - submit_2.csv
  - submit_3.csv
  - ……
  - submit_29.csv
  - submit_30.csv
  - 不同提交结果对应的模型方法.csv
  - 不同提交结果对应的误差情况.csv
- model.py：将数值型属性缺失（异常）值**填充为预测值**的代码（对应提交序号为 27 的预测结果，该结果的 **RMSE** 在 30 次提交中**表现最佳**）
- submit.csv：代码 model.py 对应的预测结果（与目录 predict_results 下 submit_27.csv 文件的内容完全相同）
- total_data_processed.csv：执行 model.py 后得到的数据预处理结果
- train.csv
- test.csv
- requirements.txt

四、 实验过程及结果

（一）实验过程

1. 加载训练集、测试集数据

首先从 csv 文件中读入数据，并维护好类别型（离散型）属性与数值型（连续型）属性的列表，如下所示。

```
        df_train = pd.read_csv(train_path, encoding='utf-8')
        df_test = pd.read_csv(test_path, encoding='utf-8')

        self.y_train = df_train['出售价格'].values
        df_train = df_train.drop(labels=['出售价格'], axis=1)
        self.category_columns = ['所属区域', '社区', '建筑类型', '当前税收级别', '街区', '地段', '当前建筑类别', '地址', '公寓号', '邮编', '居住单元', '商业单元', '总单元', '出售时税收级别', '出售时建筑类别']
        self.numeric_columns = ['土地平方英尺', '总平方英尺', '修建年份', '出售日期']
```

注意到属性“**地役权**”在全体样本中都缺失，可以删去，所以没有被放入类别型属性的列表中。

2. 加载上一次数据预处理的结果（如果有的话）

由于每次数据预处理的时间较长，因此程序会将上一次执行后得到的数据预处理结果输出到文件 total_data_processed.csv 中，以减少运行时长。这样，再次执行程序就可以直接读入文件 total_data_processed.csv 进行预测。

```
    # 数据预处理
    def preprocess(self, df_train, df_test):
        train_size = df_train.shape[0]
        # 由于数据预处理时间过久，因此将上一次数据预处理的结果存起来便于下一次加载使用
        # 检测到目录下有经过预处理的数据时，直接加载，不必重复进行数据预处理步骤
        if os.path.exists(self.data_path):
            total_data = pd.read_csv(self.data_path)
        else:
            total_data = pd.concat([df_train, df_test], ignore_index=True)
            total_data = total_data.drop(labels=['地役权'], axis=1)
                    # (接下来是真正的数据预处理过程...)
```

当**检测到目录下没有预处理结果**时，将训练集、测试集进行**合并**，正式开始进行真正的数据预处理过程，统一编号、填充缺失（异常）值。

3. 处理类别型属性

维护一个已经被编号的属性值列表 value_list，当检测到 value_list 中没有的元素时，说明这是一个新的属性值，产生一个新类别（**缺失值、异常值的出现也视为一个新类别**），将其添加到 value_list 中。直接把每个属性值在

value_list 中的索引作为其编号。

```
            # 处理类别型属性(直接编号)
            print("正在处理类别型属性...")
            for feature in self.category_columns:
                value_list = []
                total_data_index = 0
                for value in total_data[feature]:
                    if value not in value_list:
                        value_list.append(value)
                    total_data.loc[total_data_index, feature] =
value_list.index(value)
                    total_data_index += 1
                print("类别型属性[%s]被处理完毕" % feature)
            print("全部类别型属性都已被处理完毕！")
```

4. 处理数值型属性的缺失值、异常值（**方案 1：直接填充为平均值**）

根据前面的分析，可以将缺失（异常）值填充为平均值。具体如文件 origin_model/origin_model.py 中的以下代码所示。

```
            # 处理数值型属性(填充为平均数)
            print("正在处理数值型属性...")
            for feature in self.numeric_columns:
                if feature != '出售日期':
                    valid_list = []
                    for value in total_data[feature]:
                        # 土地面积或年份为 0 显然不合理，故 0 也要被视为异常值
                        if value != ' -  ' and value != '0':
                            valid_list.append(int(value))
                    # 求出正常数据的平均值来进行填充
                    mean = np.mean(valid_list)
                    total_data_index = 0
                    for value in total_data[feature]:
                        if value == ' -  ' or value == '0':
                            total_data.loc[total_data_index, feature] =
int(mean)
                        total_data_index += 1
```

5. 处理数值型属性的缺失值、异常值（**方案 2：用其它属性进行预测**）

还可以提前使用回归分析模型进行预测，将缺失（异常）值填充为预测值。具体如文件 model.py 中的以下代码所示。

```
            # 处理数值型属性(用类别型属性和正常值进行预测)
            print("正在处理数值型属性...")
            for feature in self.numeric_columns:
                if feature != '出售日期':
                    # 获得数值型属性没有异常的项
```

```
                    numeric_valid = total_data.loc[
                        (total_data[feature] != '0') &
(total_data[feature] != ' -  ')]
                    numeric_valid_X =
numeric_valid[self.category_columns]
                    numeric_valid_y = numeric_valid[feature]
                    self.data_predict_model.fit(numeric_valid_X,
numeric_valid_y)
                    total_data_index = 0
                    for value in total_data[feature]:
                        if value == ' -  ' or value == '0':
                            numeric_abnormal =
total_data.iloc[total_data_index, :]
                            numeric_abnormal =
np.array([numeric_abnormal[self.category_columns]])
                            predict_res =
self.data_predict_model.predict(numeric_abnormal)
                            # 使用预测值来填充数值型属性的缺失(异常)值
                            total_data.loc[total_data_index, feature] =
int(predict_res[0])
                        total_data_index += 1
```

6. 处理数值型属性“出售日期”（**直接转换为时间戳**）

```
                else:
                    total_data_index = 0
                    for value in total_data[feature]:
                        time_array = time.strptime(value,
"%Y-%m-%d %H:%M:%S")
                        timestamp = time.mktime(time_array)
                        # 将出售日期对应的字符串转换为对应时间戳
                        total_data.loc[total_data_index, feature] =
int(timestamp)
                        total_data_index += 1
```

7. 数据预处理的结束

将处理完的数据存入文件 total_data_processed.csv 中，便于下次加载使用，减少运行时间。同时，根据索引切割之前合并的数据集，重新得到训练集和预测集。

```
            # 将处理后的数据存入 csv
            total_data.to_csv(self.data_path, index=False,
encoding='utf-8-sig')
            print("数据处理完毕且已经被存入文件%s 中！ " % self.data_path)
        # total_data = total_data.drop(labels=['土地平方英尺', '总平方英尺
', '修建年份', '出售日期'], axis=1)
```

```
        self.X_train = total_data.iloc[:train_size, :]
        self.X_test = total_data.iloc[train_size:, :]
```

8. 建立预测模型并进行预测

根据所选用的不同机器学习模型（**以梯度提升机为例**）建立预测模型，如下所示。

```
        # 初始化模型
        self.predict_model = GradientBoostingRegressor(n_estimators=400)
```

定义好模型的训练方法和预测方法，便于之后调用得到预测结果。另外，交易价格不可能是负数，再怎么低也只能是 0，**所以小于 0 的结果肯定不合理，可以将其置 0**。具体如以下所示。

```
    # 模型训练，输出训练集平均绝对误差和均方误差
    def train(self):
        self.predict_model.fit(self.X_train, self.y_train)
        y_train_pred = self.predict_model.predict(self.X_train)
        y_train_pred = [max(0, num) for num in y_train_pred] # 将小于 0 的
结果置 0
        return mean_absolute_error(self.y_train, y_train_pred),
mean_squared_error(self.y_train, y_train_pred)

    # 模型测试，输出测试集预测结果
    def predict(self):
        y_test_pred = self.predict_model.predict(self.X_test)
        y_test_pred = [max(0, num) for num in y_test_pred] # 将小于 0 的结
果置 0
        self.df_predict['出售价格'] = y_test_pred
        return self.df_predict
```

（二）实验结果

为了尽量获得**最佳预测表现**，同时便于探究**各种因素对预测准确度的影响**，我一共提交了 30 次预测结果，不同预测结果所使用的模型方法如图 1 所示。

提交序号	数值型属性缺失（异常）值处理方式	所使用的回归预测模型	模型建立方法
1	填充平均值	决策树	DecisionTreeRegressor()
2	填充平均值	决策树+提前剪枝	DecisionTreeRegressor(max_depth=20)
3	填充平均值	决策树+提前剪枝	DecisionTreeRegressor(max_depth=30)
4	填充平均值	随机森林	RandomForestRegressor(n_estimators=100)
5	填充平均值	随机森林	RandomForestRegressor(n_estimators=500)
6	填充平均值	随机森林	RandomForestRegressor(n_estimators=600)
7	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=100, n_jobs=-1)
8	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, n_jaobs=-1)
9	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1)
10	去除所有数值型属性，只留下类别型	决策树	DecisionTreeRegressor()
11	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=300, bootstrap=True, n_jobs=-1)
12	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=600, bootstrap=True, n_jobs=-1)
13	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=800, bootstrap=True, n_jobs=-1)
14	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1, max_depth=30)
15	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=400, bootstrap=True, n_jobs=-1)
16	填充平均值	极端随机树	ExtraTreesRegressor(n_estimators=700, bootstrap=True, n_jobs=-1)
17	预测缺失（异常）值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1)
18	预测缺失（异常）值	极端随机树	ExtraTreesRegressor(n_estimators=500, bootstrap=True, n_jobs=-1, max_depth=30)
19	预测缺失（异常）值	极端随机树	ExtraTreesRegressor(n_estimators=600, bootstrap=True, n_jobs=-1)
20	预测缺失（异常）值	极端随机树	ExtraTreesRegressor(n_estimators=700, bootstrap=True, n_jobs=-1)
21	预测缺失（异常）值	梯度提升机	GradientBoostingRegressor(n_estimators=500)
22	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=500)
23	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=400)
24	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=600)
25	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=700)
26	填充平均值	梯度提升机	GradientBoostingRegressor(n_estimators=800)
27	预测缺失（异常）值	梯度提升机	GradientBoostingRegressor(n_estimators=400)
28	预测缺失（异常）值	梯度提升机	GradientBoostingRegressor(n_estimators=600)
29	预测缺失（异常）值	梯度提升机	GradientBoostingRegressor(n_estimators=700)
30	预测缺失（异常）值	梯度提升机	GradientBoostingRegressor(n_estimators=550)

图 1

以**均方误差**为评价指标，提交序号为 27 的预测结果在我的全部提交记录中表现最好，**均方误差**的大小为 **4982065.205**，在全体同学当中排名第 3（统计数据截至 2022 年 11 月 5 日）。

通过对比、分析不同的预测模型以及它们相对应的误差表现，可以得到以下**实验结论：**

第一，单个决策树的预测表现**远远不如集成学习方法**。序号为 1、2、3、10 的提交记录均使用单个决策树来进行预测，最终也是**这 4 次提交记录在全部结果中的表现最差**。剩余的 26 次提交记录都使用了**集成学习**的方法，RMSE 的取值范围大约是从 498 万至 719 万，误差在 **200 万**左右；然而，**集成学习表现最差**的记录和**单个决策树表现最好**的记录（即第 7 次、第 2 次提交）之间的 RMSE 差异达到了将近 **300 万**，对于表现更差的单个决策树，差异甚至达到了将近 **500 万**！由此可见，单个决策树与集成学习之间预测表现的差距要**显著高于**不同集成学习方法之间的差异。这是因为集成学习方法构建并结合多个学习器来完成学习任务，即使某个学习器得到了错误的预测，其它学习器也能把错误纠正过来，预测性能往往会**显著优于**单个决策树模型。

第二，对于不同的集成学习方法，从整体来看，**梯度提升机**的方法要**优于极端随机树**的方法，**极端随机树**的方法要**优于随机森林**的方法。序号为 **4、5、6** 的记录使用了**随机森林**，我将它们用**黄色**标注；序号为 **7 到 9、11 到 20** 的记录使用了**极端随机树**，我将它们用**蓝色**标注；序号为 **21 到 30** 的记录使用了**梯度提升机**，我将它们用**绿色**标注。最终得到如图 2 所示的结果。观察不难发现，不同的集成学习方法按**预测准确度**从高到低排序依次是**梯度提升机、极端随机树、随机森林**。这是由于**极端随机树**随机选择特征值的划分点位，而非像**随机森林**那样直接选择最优点位进行划分，能增大所生成的决策树的**规模**，从而在某些情况下泛化能力比**随机森林**更好（当然本次实验中两者的差距也并不大）。**梯度提升机**则**更加注重之前错误预测的实例**以自适应地改变训练数据分布的迭代过程，因此在本次实验中的预测表现要比其它两种方法更好。

	RMSE
submit_27	4982065.205
submit_21	5127310.672
submit_30	5223901.174
submit_28	5238390.48
submit_29	5278667.117
submit_23	5919893.889
submit_22	6064139.519
submit_11	6066563.511
submit_15	6119816.91
submit_24	6240305.014
submit_12	6300215.697
submit_9	6300853.958
submit_25	6303055.06
submit_26	6434972.968
submit_16	6460682.141
submit_13	6470088.091
submit_14	6510171.65
submit_19	6597305.901
submit_18	6624372.784
submit_8	6653981.297
submit_20	6667746.879
submit_17	6725187.779
submit_6	6863634.544
submit_5	6889454.942
submit_4	6956775.715
submit_7	7198038.775
submit_2	10318123.85
submit_3	10383145.05
submit_1	10500853.98
submit_10	12192967.63

图 2

第三，剪枝对于提高预测准确度**确实有一定的帮助**，但是提升的**效果并不明显**，部分情况下**甚至会降低预测准确度**。序号为 2、3 提交记录相较于序号为 1 的提交记录，以及序号为 18 的提交记录相较于序号为 17 的提交记录，都对决策树进行了剪枝且提高了预测准确度，但提升并不明显；序号为 14 的提交记录相较于序号为 9 的提交记录也进行了剪枝，但反而使误差升高了。

第四，数值型属性虽然存在较多的缺失（异常）值，但是对于提高预测准确度仍然起到了**至关重要的作用，不能因为缺失（异常）值多就将它们剔除掉**。在处理数值型属性时，我发现缺失（异常）值实在是太多了，部分特征存在将近一半的缺失（异常）值。我当时就在想数值型属性会不会反而对预测起到了负面作用，剔除掉它们后模型表现会不会更好呢？不过，对比**第 10 次、第 1 次**提交结果，我们可以发现，**剔除后预测误差更大了**，数值型属性还是起到了重要的**促进**作用，**哪怕将大量的缺失（异常）值全部填充为平均值，也比完全抛弃它们要强**。

第五，填充为平均值、填充为预测值这两种数值型属性缺失（异常）值的处理方式**无绝对的优劣之分，分别适用于不同的预测模型**。序号为 17、18、19、20 的记录，对应于序号为 9、14、12、16 的记录，两组均使用**极端随机树**且**函数参数对应相等**，不同之处仅在于前者使用填充预测值的方式，后者则直接填充平均值，**填充平均值的方式表现更好**。序号为 21、27、28、29 的记录，对应于序号为 22、23、24、25 的记录，两组均使用**梯度提升机**且**函数参数对应相等**，不同之处仅在于前者使用填充预测值的方式，后者则直接填充平均值，但这次则是**填充预测值的方式表现更好**。由此可见，两种数值型属性的缺失（异常）值处理方

案适用于不同的模型，各有千秋。对于**极端随机树**，**填充平均值**的方法更好；对于**梯度提升机**,**填充预测值**的方法更好。当然,由于梯度提升机的整体表现更好，我最终提交的版本还是使用了**填充预测值**的方法。

五、 实验心得

通过本次大作业的锻炼，我进一步巩固了课程所学的有关数据理解、数据分析、数据挖掘、数据预处理的知识，对传统机器学习算法（如决策树、梯度提升机、随机森林等）有了更深入的理解。在这个过程中，我动手编程完成一个**预测房屋交易价格的实战项目**，进一步熟悉 Python 语言以及 PyCharm 开发环境的用法，熟悉 numpy、pandas、sklearn 等 Python 包中主要库函数的用法，深刻锻炼了**编程调试**能力、**分析设计**能力、**验证探究**能力以及**切实解决问题**的能力，为后续课程学习乃至未来科研、工作奠定坚实的基础。

在刚拿到大作业题目以及数据集的时候，我对于如何进行**分析、设计、预测**是比较迷茫的。虽然，我们不需要自己从头开始**“造轮子”**，可以在不深入理解决策树、随机森林、梯度提升机等经典机器学习算法**具体实现细节**的情况下，直接调用 sklearn 库中相应的 API 函数完成**预测**任务。但是，所给的数据集非常“刁钻”，如果预处理的工作没有做好，直接调用库函数预测出来的结果恐怕会不尽如人意。不过，在分析清楚本次大作业数据集的特点、数据集到底复杂在什么地方之后，我们设计、处理的方式就可以更有针对性了。

本次大作业数据集的**复杂性**主要体现在以下两个方面：

第一，数据量**大**，需要分析的特征**数量多、种类多**。训练集一共有 4 万多条记录，测试集也有 4 千多条记录，如果对它们进行较为复杂的分析、处理操作，势必会导致程序的运行时间过长。此外，部分特征的取值情况非常多（例如，训练集中“**地址**”这一特征属性就有超过 3 万种取值，唯一项数目甚至超过了 1 万个!)。这意味着属性取值相同的情况很少出现，某些取值只在少部分样本中存在（**甚至可能只有一条记录**),不利于模型在训练时提取到有关该特征取值的信息，从而导致回归分析模型的预测准确度受到影响。此外，特征的种类多、特征的取值情况多，还意味着**难以对类别型属性使用相对更为严谨的独热编码方法**（因为使用独热编码会导致变换后特征的维度过高，从而使运算时间大为加长），不利于模型性能的提升。

第二，数据中存在较多的缺失、异常情况，且缺失值、异常值占全体样本的比重大。由于缺失（异常）值的数量并不少，我们不能像**本学期实验二**那样直接删除包含缺失(异常)值的记录，因为这会导致大量的样本损失。针对这一问题，我分别为类别型属性和数值型属性设计了不同的解决方案。对于类别型属性，可以直接将缺失值**视为一种新的类别**。对于数值型，采用填充**平均值**或者填充**预测值**的方法进行处理。由于两种填充方法**各有利弊**，为了获得最大的预测准确度，我在编程时只能将它们**都实现**，并通过实验结果评估它们的实际表现。

数据预处理结束后，其实**本次实验的主要工作就已经完成了**。因为对于建立模型、训练模型并进行预测，我们直接调用 sklearn 中的库函数就能完成任务，无需关心底层的实现细节。但是，我在完成大作业时还是充分进行了多组对照实验,在**探索最佳模型参数**的同时,也对**不同因素的影响及作用**有了更深刻的认识，并得出了**五个重要的实验结论**（已经在**实验结果**部分给出，此处不再赘述），**与理论课所学内容遥相呼应**。

本次大作业我的设计方案还可以从以下三个方面进行**改进**：

第一，可以为每一个存在缺失（异常）值的数值型属性**新增一列对应的“xxx数值属性是否缺失或异常”属性**，将变量**映射到更高维的空间**。对于取值正常的数据项，“xxx 数值属性是否缺失或异常”的值为 0，对于取值缺失或异常的数据项，除了要使用前面介绍的**填充平均值**或**填充预测值**的方法进行**填充**外，还要将属性“xxx 数值属性是否缺失或异常”的值赋为 1。这种方式**类似于处理类别型属性时将“缺失或异常”视为一个新类别**的做法，额外增加字段来标记是否异常或缺失，能让模型在训练时**知道当前连续型特征取值是否因缺失或异常被填充**，更有针对性地进行分析和预测。

第二，可以从训练集中**划分出验证集**，使用 **K 折交叉验证法**，更高效地寻找出**最优超参数**。本次实验中，我没有划分出**验证集**，在为模型调整超参数的时候，主要通过多次提交（即前面介绍的 30 次提交结果）、查看模型在**测试集**上预测误差的方式来寻找更合适的**超参数**与**预测模型**。这种方法效率不高。因此，应当遵循**机器学习回归任务的经典范式**，划分出验证集评估模型表现，确定优化方向，更高效地进行调整。

第三，可以尝试对部分类别型属性使用**独热编码**。前面已经介绍过，在对类别型属性进行编码时最严谨的做法应当是使用独热编码的方式。当然，直接对本次实验的全体数据进行独热编码会导致特征维度增加过多，降低模型的运行性能。不过，我们还是可以尝试对**部分取值情况少的类别型属性进行独热编码**，并通过模型在**验证集**、**测试集**上的表现来查看独热编码方式是否真的提高了预测准确度。

总之，通过完成本次大作业，我体验了一次**使用机器学习方法进行回归预测的完整流程**，也对如何分析数据、如何处理数据有了更深入的认识，对各类机器学习方法有了更透彻的理解，进一步夯实理论课所学的内容。在实验过程中，我免不了会遭遇**代码无法跑通、程序报错、程序运行无法出现预期现象**的情况，查找 bug 的过程既是一场**对人心性与定力的考验**，也是一次**难能可贵的历练机会**。通过控制台打印、断点调试等方式，我最终也顺利地解决了遇到的各类问题，并取得了不俗的预测表现（排名第 3）。当然，不足之处和可以改进的地方还有很多。实验结束后实验报告的撰写，也有利于我养成严谨认真的科学态度，同时也便于我日后反思、回顾本次大作业。撰写报告虽然耗时较长，但却在无形之中加深了我对本次大作业任务的理解，让我重新审视自己的设计方案到底是什么、有什么优缺点、取得了什么样的效果、可以得出什么结论、未来还有哪些可以改进的地方等等，令人受益匪浅。

总的来说，本学期《大数据导论》课程以及本次《大数据导论》课程的大作业还是让我收获较大。