Homework 09 House-Price-Prediction

北京大学 2024 春季人工智能基础第九次课程作业

Arthals 2110306206

zhuozhiyongde@126.com

2024.06

1 背景介绍

本次作业的目的是利用一系列可能与房价相关的因素（特征）训练出一个房价预测模型。

房价预测是一个经典的机器学习问题，通过对历史房价数据和相关特征的分析，我们可以构建一个预测模型，用于预测未来的房价。

具体来说，我们需要做的包括筛选、处理特征，并在此基础上构筑一个回归预测模型。

2 数据探索性分析

通过对 CSV 的简要阅读分析，我们可以发现数据存在一些问题，包括数据缺失、异常值等，而且很多文本信息类的非数字信息无法直接送入模型进行学习。需要我们去处理。

3 特征工程

3.1 处理无效数据

对于 LAND SQUARE FEET 和 GROSS SQUARE FEET 两个数据字段，我们将无效值替换为0，并用中位数替代0值。

```
# 处理无效数据，替换无效值、0 为中位数
# 定义需要处理的列
invalid_columns = ["LAND SQUARE FEET", "GROSS SQUARE FEET"]

# 遍历每一列进行处理
for column in invalid_columns:
    # 替换无效值为 0
    train[column].replace(" -  ", "0", inplace=True)
    # 转换数据类型为 int64
    train[column] = train[column].astype("int64")

    # 筛选出非 0 的行，计算均值
    valid_index = train[column] ≠ 0
    column_mean = int(train[valid_index][column].median())
    print("Mean: ", column_mean)
```

```
    # 替换 0 值为均值
    train.loc[train[column] == 0, column] = column_mean

    # 同样的处理应用到 data_X 上
    data_X[column].replace(" -  ", "0", inplace=True)
    data_X[column] = data_X[column].astype("int64")
    data_X.loc[data_X[column] == 0, column] = column_mean
    print(data_X[column].value_counts().sort_values(ascending=False).head(10))
```

3.2 删除无效特征

根据经验，我们移除了一些大概与房价无关的信息字段。

```
del data_X["APARTMENT NUMBER"]
del data_X["BUILDING CLASS AT PRESENT"]
```

3.3 处理地址数据

经过观察，我们发现地址 ADDRESS 字段中，有一些太过具体了，完全属于多余信息，不利于后续进行 one-hot 编码处理，所以使用字符串处理将过于细致的部分移除。

```
# 处理 ADDRESS 列，移除多余信息
data_X["ADDRESS"] = data_X["ADDRESS"].apply(lambda x: x.split(",")[0])
```

3.4 类别变量处理

正如之前提到的，很多的文本数据不利于直接送入模型，我们将之转为分类变量。

对于 ADDRESS 字段，我们同时将之转为数值格式，方便后续归一化处理。

```
# 转换为分类变量
# 需要转换为category类型的列名列表
category_columns = [
    "BOROUGH",
    "BUILDING CLASS CATEGORY",
    "TAX CLASS AT PRESENT",
    "TAX CLASS AT TIME OF SALE",
    "NEIGHBORHOOD",
    "ZIP CODE",
    "BLOCK",
    "BUILDING CLASS AT TIME OF SALE",
]

numerical_catagory_columns = ["ADDRESS"]
```

```
# 进行类型转换
for column in category_columns:
    data_X[column] = data_X[column].astype("category")

# 对于 numerical_catagory_columns，需要转换为 category 类型，并且转换为数值
for column in numerical_catagory_columns:
    data_X[column] = data_X[column].astype("category")
    data_X[column] = data_X[column].cat.codes

# import seaborn as sns
# sns.regplot(x="ADDRESS", y="SALE PRICE", data=train)
data_X.info()
```

3.5 日期处理

现有的日期字段属于文本信息，我们需要从中提取出数值信息。

```
# 从 SALE DATE 中提取出年、月、日
data_X["SALE DATE"] = pd.to_datetime(data_X["SALE DATE"])
data_X["SALE YEAR"] = data_X["SALE DATE"].dt.year
data_X["SALE MONTH"] = data_X["SALE DATE"].dt.month
data_X["SALE DAY"] = data_X["SALE DATE"].dt.day
del data_X["SALE DATE"]
data_X.info()
```

3.6 标准化数值特征

使用 `StandardScaler` 对数值特征进行了标准化。

```
all_columns = data_X.columns
# 删除类别列 category_columns
one_hot_columns = [
    "BOROUGH",
    "BUILDING CLASS CATEGORY",
    "TAX CLASS AT PRESENT",
    "TAX CLASS AT TIME OF SALE",
    "NEIGHBORHOOD",
    "ZIP CODE",
    "BLOCK",
    "BUILDING CLASS AT TIME OF SALE",
]
numerical_columns = list(set(all_columns) - set(one_hot_columns))
print("numerical_columns:", numerical_columns)
scaler = StandardScaler()
data_X[numerical_columns] = scaler.fit_transform(data_X[numerical_columns])
```

3.7 独热编码

对于分类变量，使用独热编码将其转换为数值形式。

```
# 对于类别数据，使用 one-hot 编码
one_hot_encoded = pd.get_dummies(data_X[one_hot_columns])
one_hot_encoded.info(verbose=True, memory_usage=True)
data_X = data_X.drop(one_hot_columns, axis=1)
data_X = pd.concat([data_X, one_hot_encoded], axis=1)
```

3.8 重新提取训练集、测试集

```
# 重新从 data_X 中分离出 train_X 和 test_X
train_X = data_X[:num_train_samples].to_numpy()
test_X = data_X[num_train_samples:].to_numpy()
```

4 预测模型的建立

我们使用了 LightGBM 作为模型，进行梯度提升决策树（GBDT）训练。

手动调整参数后，我们获得了达到了预期的性能要求。

```
# 使用梯度提升决策树（GBDT）来训练模型
lgb_train = lgb.Dataset(train_X, train_y)
lgb_eval = lgb.Dataset(test_X, test_y, reference=lgb_train)

params = {
    "num_leaves": 400,
    "feature_fraction": 0.45,
    "learning_rate": 0.05,
    "objective": "mape",
}

lgb_t = lgb.train(params=params, train_set=lgb_train, num_boost_round=2000)
y_pred = lgb_t.predict(test_X)

mean_absolute_percentage_error(test_y, y_pred)
```

5 结果的分析与总结

5.1 结果分析

在本次实验中，我们使用了 LightGBM 进行梯度提升决策树（GBDT）训练，并将模型应用于测试集上。最终输出的平均绝对百分比误差（MAPE）如下：

```
0.309445619254423
```

> 平均绝对百分比误差（MAPE）是评估预测模型准确性的一种常用指标。它的公式为：
>
> $$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{1}$$
>
> 其中：
>
> - n 是样本数量
> - y_i 是真实值
> - $\hat{y}_i$ 是预测值

特征工程有效性：通过处理无效数据、删除无用特征、处理地址列和日期列、标准化数值特征以及使用独热编码处理分类变量，我们成功地为模型提供了干净且有用的数据。这些步骤在提高模型准确性方面起到了关键作用。

模型选择：LightGBM 是一种高效的梯度提升框架，其性能远远优于默认示例代码中给出的 sklearn，适用于大规模数据集和高维特征。

参数调优：我们选择了合适的参数，如 `num_leaves`、`feature_fraction` 和 `learning_rate`，这些参数在一定程度上优化了模型的性能。

5.2 总结

尽管我们取得了较好的结果，但仍有改进空间：

1. **特征选择**：可以进一步探索和选择更有代表性的特征，或者通过特征工程生成新的特征。
2. **模型调优**：可以尝试更多的模型参数组合，进一步优化模型性能。

本次实验展示了数据预处理和特征工程在构建高效预测模型中的重要性。通过合理的特征处理和模型选择，我们能够构建出较为准确的房价预测模型。