中国人民大学
RENMIN UNIVERSITY OF CHINA

房产定价Hackathon Final-term 项目汇报

财政金融学院
王晨曦
2024年12月26日

数据处理

- ✓ 数据获取：

  兴趣点Point of Interest

  API接口？邮件？图书馆？盈利机构？同学？

- ✓ Inspiration:

  Mid-term中发现经纬度信息不好利用 + 葛老师引导

- ✓ Challenges：

  Pandas DataFrame → MomeryError 内存报错

  - ➢ Dask库：**Partitioning & Lazy Evaluation**

  运行速度过慢：

  - ➢ 计算复杂度：

    65,271,054 × (102117+17908) × geopy.distance.geodesic × n_features

    依据小样本运行速度判断：至少需要3500个小时

  数据更新问题：

  - ➢ 交易年份涉及 2018-2022 共**五**年

```python
import dask.dataframe as dd

for csv_file in csv_files:
    # 使用更大的blocksize读取CSV文件
```

服务	服务范围	当月调用量(次)	当日调用量(次)	日配额(次/日)	并发量上限(次/秒)
基础搜索服务	关键字搜索	0	0	100	3
	周边搜索	11	0	100	3
	多边形搜索	0	0	100	3
	ID查询	0	0	100	3
	输入提示	0	0	100	3

```python
    return partition

# 应用extract_first_type函数并输出中间结果
ddf_extracted = ddf_dropped.map_partitions(extract_first_type, meta=ddf_dropped)
print("提取type列中的第一部分内容后的数据：")
print(ddf_extracted.shape)  # 查看提取type列后的数据量
# print("数据行数：", ddf_extracted.shape[0].compute()) # 调用compute计算行数
print(ddf_extracted.head())
```

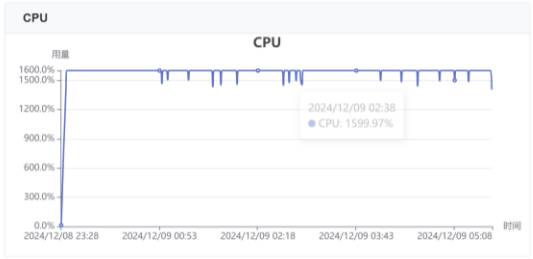
Code：1.1.5

数据处理

✓ **运行速度**及**数据更新**解决措施：

最终数据选取：选择样本中**2022年的POI数据**（购买）
对数据进行预处理——只考虑样本中五个城市附近的POI
最终：

样本城市名	真实城市名	POI数目	Train	Predict	Details
冰城	哈尔滨	250,042	13,735	1,405	**266**
近畿	廊坊	172,491	5,009	1,269	**127**
津门	天津	409,020	26,583	5,604	**683**
江城	武汉	4,306,553	5,864	2,972	**536**
长安	西安	398,638	19,102	2,576	**453**
天府	重庆	892,261	31,824	4,082	**600**
合计：		2,553,105	102,117	17,908	**2,665**



Code：1.1.3、1.1.4、1.1.5

```
['旅游景点' '生活服务' '休闲娱乐' '交通设施' '汽车相关' '酒店住宿' '餐饮美食' '商务住宅' '运动健身' '购物消费'
 '医疗保健' '公司企业' '科教文化' '金融机构']
['景点' '公厕' '其他' '停车场' '加油站' '公园' '四星级酒店' '三星级酒店' '旅馆' '中国菜' '诊所' '便利店'
 '超市' '汽车维修' '农林牧渔' '信息咨询中心' '植物园' '住宅区' '图书馆' '家电数码' '幼儿园' '物流' '邮局'
 '摄影打印' '市场' '银行' '医药销售' '小吃快餐' '其他能源站' '文体用品' '家居建材' 'ATM' '美容理发' '公司'
 '洗衣' '洗车' '产业园' '综合医院' '农家乐' '酒吧' '保险' '中学' '洗浴推拿' '电讯营业厅' '彩票销售' '花鸟鱼虫'
 '专科医院' '蛋糕甜品店' '汽车配件' '汽车养护' '服务区' '火车' 'KTV' '小学' '疾病预防' '水族馆' '工厂'
 '别墅区' '冰雪运动' '长途汽车' '台球' '剧场' '度假养老' '青旅' '经济型连锁酒店' '商业街' '游乐场' '水上运动'
 '广场' '外国菜' '培训单位' '汽车租赁' '二手车' '科研单位' '纪念馆' '购物中心' '收费站' '公共事业' '港口码头'
 '宗教' '公交站' '咖啡' '中介' '文化宫' '会展展览' '驾校' '动物医疗' '百货商场' '艺术团体' '露营地' '轮渡'
 '成人教育' '职业技术教育' '汽车销售' '网吧' '动物园' '广播电视' '宿舍' '博物馆' '写字楼' '充电站' '商住两用楼宇'
 '美术展览' '高等教育' '跆拳道' '游泳' '综合体育馆' '加气站' '急救中心' '电影院' '健身中心' '篮球' '茶座'
 '五星级酒店' '科技馆' '足球' '马术&赛马' '棋牌室' '地铁' '高尔夫球' '免税店' '社区中心' '飞机' '乒乓球'
 '投资理财' '羽毛球' '网球' '工业大楼' '档案馆' '新闻出版' '户外健身场所' '天文馆' '红色旅游' '保龄球' '壁球'
 '世界遗产']
```

细类(中类)特征共136个

Code：1.1.1

Postscript：数据同样已上传至datahub上，感兴趣的同学可以联系我开共享~

➢ 什么信息对我们是**有用的**?

✓ 最近POI的距离?

✓ 距离<3km的POI点的个数?

joblib 并行计算

```
# 使用joblib进行并行处理
tasks = (
    delayed(process_category)(
        row, poi_df_dict[row['城市']], category, distance_threshold
    )
    for index, row in df.iterrows()
    for category in all_categories
)

results_list = Parallel(n_jobs=-1)(tasks)
```

模型Xgboost

- 网格调参
- 手动调参

+ 贝叶斯优化

```python
# 定义XGBoost训练的参数
params = {
    'objective': 'reg:squarederror',  # 回归问题，使用均方误差作为损失函数
    'max_depth': 7,  # 树的最大深度，控制模型的复杂度 7
    'eta': 0.034,  # 学习率，控制每次迭代更新的步长 0.033
    'subsample': 0.8,  # 训练时使用80%的样本，防止过拟合 0.8
    'colsample_bytree': 0.8  # 每棵树训练时，随机选择80%的特征 0.8
}

# 设置训练的最大迭代轮数和早停轮数
num_boost_round = 30000  # 最大训练轮数
early_stopping_rounds = 1000  # 如果在1000轮内验证集的误差没有改善，则提前停止训练

# 训练模型
evals_result = {}  # 存储每轮训练的评估结果
bst = xgb.train(
    params,  # 使用上述定义的参数
    xgb.DMatrix(X_train, label=y_train),  # 训练集数据
    num_boost_round,  # 最大迭代次数
    evals=[(xgb.DMatrix(X_test, label=y_test), 'eval')],  # 验证集
    early_stopping_rounds=early_stopping_rounds,  # 设置早停机制
    evals_result=evals_result  # 存储训练过程中的评估结果
)
```

图3.1：手动调整 Xgboost 超参数

Test_RMSE: 335335.939194466
Predict_Score: 83.112

```python
def objective(trial):
    # 贝叶斯优化的参数空间
    param = {
        'objective': 'reg:squarederror',
        'eval_metric': 'rmse',
        'max_depth': trial.suggest_int('max_depth', 5, 9),
        'learning_rate': trial.suggest_float('learning_rate', 0.03, 0.038, log=True),
        'subsample': trial.suggest_float('subsample', 0.75, 0.85),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.75, 0.85),
        'n_estimators': trial.suggest_int('n_estimators', 100, 1000, step=100),
        'gamma': trial.suggest_float('gamma', 0, 3),  # 控制是否后剪枝
        'lambda': trial.suggest_float('lambda', 0, 1),
        'alpha': trial.suggest_float('alpha', 0, 1)
    }
```

图3.2：贝叶斯优化使用的超参数范围空间

```python
    # 使用交叉验证（K折交叉验证）
    kfold = KFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(model, X, y, cv=kfold, scoring='neg_root_mean_squared_error', n_jobs=-1)

    # 返回平均的负RMSE作为优化目标（注意我们需要最小化RMSE，所以使用负值）
    rmse = -np.mean(scores)
    return rmse
```

图3.3：贝叶斯优化 5 折交叉验证

- 特征标准化 + 对价格取对数处理

Train_RMSE: 0.12112
Score：83.646

Code：2.4 优化后的Xgboost模型

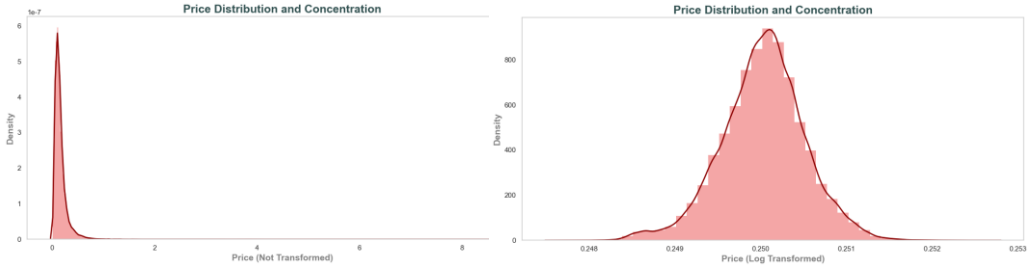
模型keras ANN

- 对于Price，实行对数变换

$$Price_{log} = \ln(1 + Price)$$

```
# 确保目标变量不具有太大的偏差（对数变换）
y_train_log = np.log1p(y_train)  # 取对数处理目标变量
y_test_log = np.log1p(y_test)  # 同样处理测试集
```

- 构建神经网络，共五层
- 标准化和对数变换 restate
  - ✓ 价格明显右偏 —— 对数变换



  - ✓ 各变量度量标准统一 —— 标准化
  - ✓ ANN中为避免出现梯度消失或梯度爆炸的情况 —— 标准化（BatchNormalization）

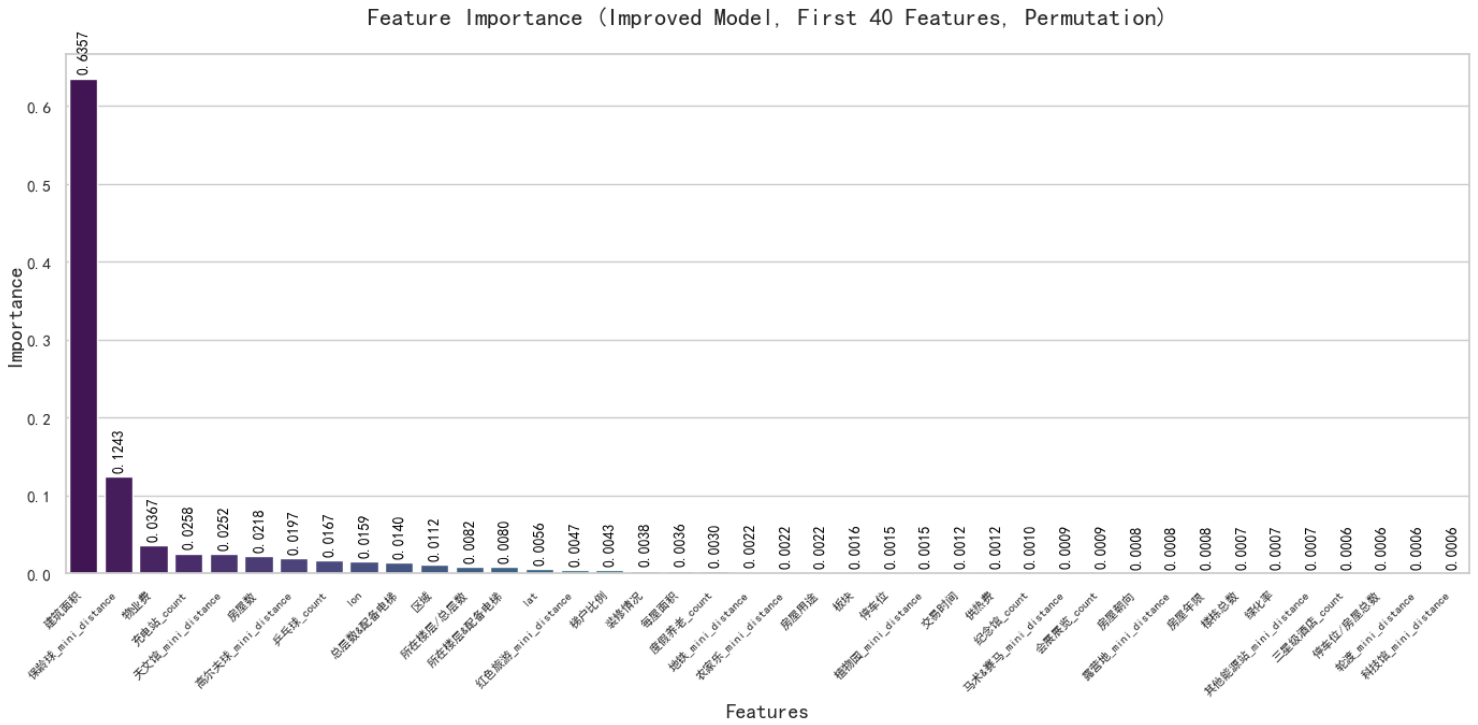
```
# 构建神经网络模型
model = Sequential()

# 第一层，加入BatchNormalization和Dropout
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='gelu'))  # 第一个隐藏层，使用 GELU 激活函数
model.add(BatchNormalization())  # 添加批标准化层
model.add(Dropout(0.13))  # 添加Dropout层，防止过拟合

# 第二层
model.add(Dense(256, activation='gelu'))
model.add(BatchNormalization())  # 添加批标准化层
model.add(Dropout(0.13))  # 添加Dropout层

# 第三层（可选）
model.add(Dense(128, activation='gelu'))
model.add(BatchNormalization())  # 添加批标准化层
model.add(Dropout(0.13))  # 添加Dropout层

# 第四层
model.add(Dense(64, activation='gelu'))
model.add(BatchNormalization())  # 添加批标准化层
model.add(Dropout(0.13))  # 添加Dropout层

# 第五层
model.add(Dense(32, activation='gelu'))
model.add(BatchNormalization())  # 添加批标准化层
model.add(Dropout(0.13))  # 添加Dropout层

# 输出层，注意使用线性激活，因为目标是连续值
model.add(Dense(1))  # 输出层，预测房产价格
```

Train_RMSE: 0.18755
Predict_Score: 77.606

模型结果及改进

Xgboost 特征重要性排名：

- Weight
- Permutation Importance



Code: 2.2.1 Xgboost-手动调参
& Code: 2.4 改进后的Xgboost

未来的改进：

- 数据处理
  - 利用**交易年份**寻找对应年份的POI数据
  - 对于**details**中没有的小区，应重新搜索
  - 对文本的利用，课程学习的**语义向量**等知识
  - **交互项**的改进
- 模型训练
  - **ANN** 可以考虑更改参数、激活函数
  - **交叉验证**优化可能能够提升参数的同时，兼顾模型的**泛化能力**，防止overfitting

中国人民大学
RENMIN UNIVERSITY OF CHINA

复兴栋梁 强国先锋

谢谢大家！

Thank the experts for listening and welcome the criticism!