第七组展示

2025.10.30

小组成员：唐子云、朱炫臻、陈子卓

1、通过构造FunctionTransformer将所有数据处理过程全部封装到pipeline里，使用只存在于训练集的特征处理和填充测试集与预测数据，

（1）完全避免训练集与测试集之间数据泄露问题

（2）测试集与六折交叉验证的MAE、RMAE评估结果更加客观，能更好反映模型性能，便于后续调整

（3）一键处理训练集、测试集、预测数据，方便快捷。

```python
def parse_house_type(X):
    # 直接操作 Series
    dims = {
        "室数": r'(\d+)室',
        "厅数": r'(\d+)厅',
        "厨数": r'(\d+)厨',
        "卫数": r'(\d+)卫'
    }
    df = pd.DataFrame(index=X.index)
    for key, pattern in dims.items():
        df[key] = X.iloc[:,0].fillna("").str.extract(pattern).fillna(0).astype(int)
    return df

# 构造 FunctionTransformer
house_type_transformer = FunctionTransformer(parse_house_type, validate=False)

# 构造 pipeline
house_type_pipeline = Pipeline(steps=[
    ("parse", house_type_transformer),
    ("standardize", StandardScaler()),
    ("impute", SimpleImputer(strategy='most_frequent'))
])
```

2、使用K-Means聚合城市、区域、板块、年份、区县，减少数据维度，既能提高模型计算速度，同也能时提高模型的泛化能力。

```python
cluster_features = ["城市", "区域", "板块", "年份", "区县"]  # 聚类特征

cluster_pipeline = Pipeline(steps=[
    ("impute", SimpleImputer(strategy="mean")),
    ("standardize", StandardScaler()),
    ("cluster", KMeans(n_clusters=10, random_state=42))
])
```

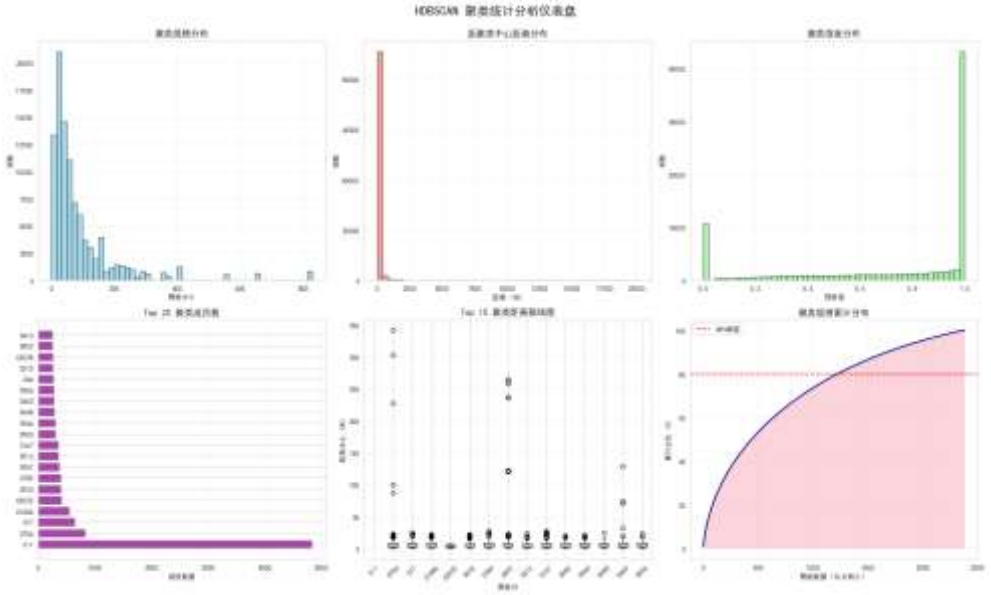
3、对于“客户反馈”列，引入通过京东商品评论数据微调过的RoBERTa-base中文情感二分类模型“roberta-base-finetuned-jd-binary-chinese”，通过“正面”类别的概率-“负面”类别的概率批量计算出每条中文文本的情感得分[-1,1]。同时，使用cuda调用显卡参与模型计算，大幅提高模型计算速度。

```python
from tqdm import tqdm
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
import numpy as np

def compute_sentiment_scores(X, batch_size=32, device='cuda'):
    """
    输入：中文客户反馈文本
    输出：每条文本的情感得分，范围在[-1, 1]
    """
    model_name = "uer/roberta-base-finetuned-jd-binary-chinese"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSequenceClassification.from_pretrained(model_name)

    model.to(device)
    model.eval()

    scores = []
    s = X.iloc[:,0].fillna("")

    # 批量处理文本
    for start_idx in tqdm(int)(range(0, len(s), batch_size), desc="计算情感得分"):
        batch_texts = s[start_idx:start_idx + batch_size]

        # 跳过空文本
        if batch_texts.str.strip().eq("").all():
            scores.extend([0.0] * len(batch_texts))
            continue

        # tokenizer 返回字典格式，并已经是张量 list
        inputs = tokenizer(batch_texts.tolist(), return_tensors="pt", truncation=True, padding=True).to(device)

        with torch.no_grad():
            outputs = model(**inputs)
            probs = torch.softmax(outputs.logits, dim=1)

            # 获取每条文本的情感得分：正面 - 负面
            batch_scores = (probs[:, 1] - probs[:, 0]).cpu().numpy()

        scores.extend(batch_scores)

    return np.array(scores).reshape(-1, 1)
```

唐子云

基于HDBSCAN与防泄漏目标编码的特征

——如何用线性特征刻画非线性关系



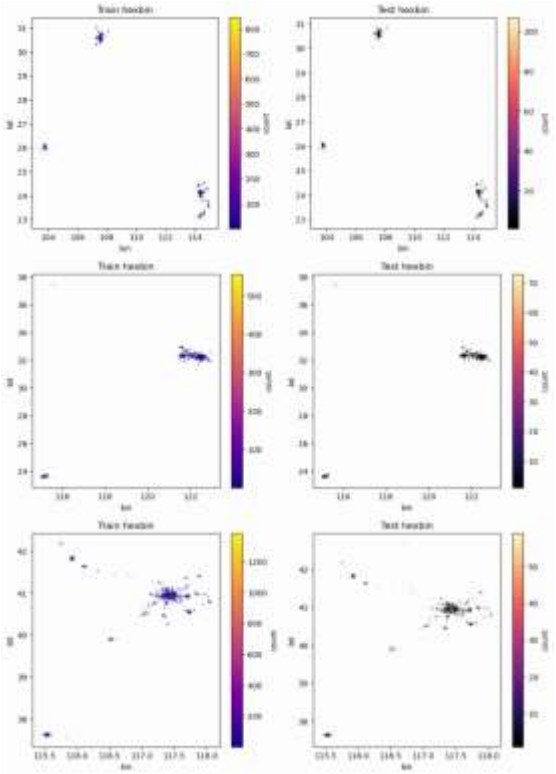
- **构思来源：**

1.现实场景，同一个小区类似户型的房子往往价格(price/m2)相近

2.对大量区域、板块特征进行降维，相当于利用地理因素对price先进行预训练

- **检验发现：**train集和test集的经纬度分布高度相似

（因此提出猜测train集和test集都是从同样的小区集合中提取的样本）



- **详细流程及注意点：**

1.每条数据的特征要用留一法计算（计算除自己之外cluster里其他price/m2的平均）

2.不能直接掉包使用CV和网格搜索，因为每次重新取训练集和验证集都需要重新生成一遍cluster_mean_price/m2特征，因此需要自己手写

3.对于无法聚类的点，此特征列采取“板块-区域-城市”层级填充

price

Metrics	IS	OS	CV	Kaggle
OLS	235,297.74	237,976.87	237,047.45	66.75
Lasso	230,060.47	231,617.17	232,002.60	68.82
Ridge	232,951.04	234,507.95	234,762.98	67.15
Best	230,060.47	231,617.17	232,002.60	68.82

rent

Metrics	IS	OS	CV	Kaggle
OLS	97,899.32	98,054.13	124,315.09	66.75
Lasso	97,332.68	97,679.85	78,904.35	68.82
Ridge	97,105.97	97,991.42	80,302.64	67.15
Best	97,332.68	97,679.85	78,904.35	68.82

Price

Metrics	IS	OS	CV	kaggle
OLS	221,499.56	222,814.73	223,432.07	68.82
Lasso	217,733.47	219,114.81	220,104.42	70.5
Ridge	219,681.67	220,647.68	221,861.68	69.23
Best	217,733.47	219,114.81	220,104.42	70.5

Rent

Metrics	IS	OS	CV	kaggle
OLS	97,004.28	97,710.47	159,293.09	68.82
Lasso	99,604.68	99,979.01	67,889.52	69.23
Ridge	97,855.86	98,318.53	67,840.58	70.5
Best	97,855.86	98,318.53	67,840.58	70.5

- **结果：baseline：59.69 → 加入cluster防泄漏目标编码：68.82→参考结合组员亮点:70.5**

朱炫臻

亮点1： 基于房地产专业知识（楼层类型、朝向偏好、户型分类）创造新特征将文本信息（如“南北通透”）转化为有意义的数值特征

```python
def direction_score(self, x):
    """计算朝向评分"""
    if pd.isna(x): return 0
    x_str = str(x)
    score = 0

    direction_scores = {'南': 3, '东': 2, '西': 1, '北': 0.5}
    for direction, points in direction_scores.items():
        if direction in x_str:
            score += points

    # 特殊组合加分
    if '南北' in x_str:
        score += 2
    if '东南' in x_str or '南东' in x_str:
        score += 1

    return min(score, 8)
```

亮点2： 同一城市/区域/板块的房屋特征相似，分组填充更准确，异常处理确保即使分组失败也能完成填充

```python
def group_fill_numeric_missing(self, df: pd.DataFrame) -> pd.DataFrame:
    """数值型缺失值按城市、区域、板块分组填充"""
    df = df.copy()

    # 检查是否有分组列
    group_cols = []
    for col in ['城市', '区域', '板块']:
        if col in df.columns:
            group_cols.append(col)

    if not group_cols:
        print("未找到城市、区域、板块列，使用全局中位数填充")
        numeric_cols = df.select_dtypes(include=[np.number]).columns
        for col in numeric_cols:
            if df[col].isnull().any():
                median_val = df[col].median()
                if pd.notna(median_val):
                    df[col] = df[col].fillna(median_val)
                else:
                    df[col] = df[col].fillna(0)
        return df

    print(f"使用 {group_cols} 进行分组填充")

    # 先填充分组列本身的缺失值
    for col in group_cols:
        if df[col].isnull().any():
            df[col] = df[col].fillna("未知")

    # 对数值列进行分组填充
    numeric_cols = df.select_dtypes(include=[np.number]).columns

    for col in numeric_cols:
        if df[col].isnull().any():
            try:
                # 创建分组填充的值
                group_means = df.groupby(group_cols)[col].transform('median')
                # 如果分组后仍有NaN，用全局中位数填充
                global_median = df[col].median()
                df[col] = df[col].fillna(group_means).fillna(global_median)
                print(f"列 {col}: 使用分组填充 + 全局中位数填充")
            except Exception as e:
                print(f"列 {col} 分组填充失败：{e}，使用全局中位数填充")
                df[col] = df[col].fillna(df[col].median())

    return df
```

亮点3： 利用面积室比、卫室比等反映房屋使用舒适度，捕捉面积平方项捕捉面积与价格的非线性关系；将连续的房间数转换为分类标志，便于模型理解

```python
# 面积相关特征
if "建筑面积" in df.columns and "室" in df.columns:
    df["面积室比"] = df["建筑面积"] / (df["室"] + 1)
    df["建筑面积2"] = df["建筑面积"] ** 2
    df["面积分箱"] = pd.qcut(df["建筑面积"], q=5, labels=False, duplicates="drop")

# 户型相关特征
if {"室", "厅", "卫", "厨"}.issubset(df.columns):
    df["房间总数"] = df["室"] + df["厅"] + df["卫"] + df["厨"]
    df["卫室比"] = df["卫"] / (df["室"] + 1)
    df["厅室比"] = df["厅"] / (df["室"] + 1)

    # 户型分类
    df["是否一居室"] = (df["室"] == 1).astype(int)
    df["是否二居室"] = (df["室"] == 2).astype(int)
    df["是否三居室"] = (df["室"] == 3).astype(int)
    df["是否大户型"] = (df["室"] >= 4).astype(int)
```

陈子卓