数据处理报告

1. 数据提取

extract.sql　　SQL　复制代码

```sql
SELECT uid,
       tran_flag,
       tran_amt,
       cac_intc_pr,
       dr_cr_code,
       pay_term
FROM dm.dm_v_tr_huanx_mx INTO OUTFILE '/home/vboxuser/dm_v_tr_huanx_mx.csv'
 FORMAT CSV
……
```

使用了clickhouse自带的csv导出功能

导出了24张表中需要的数据，导出的数据汇总在”数据库导出的csv数据（已筛选）”文件夹中。

2. 数据特征提取

使用pandas库read_csv函数读取提取好的csv文件并创建对应列表。

feature_credit.ipynb Python 复制代码

```
es = ft.EntitySet(id = "my_set")
es.entity_from_dataframe(entity_id='pri_credit_info',dataframe=pri_credit_info,index='index_column')
……
```

使用featuretool库创建实体集EntitySet以便操作数据，然后使用entity_from_dataframe函数添加数据，其中entity_id表示要从哪个表获取数据，dataframe表示要从哪个DataFrame中获取数据

credit_info.ipynb Python 复制代码

```
relationships = [
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_as_djk_info"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_as_djkfq_info"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_contract_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_djk_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_dsf_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_duebill_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_huanb_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_huanx_mx"]["uid"]),
    ft.Relationship(es["pri_star_info"]["uid"], es["dm_v_tr_sa_mx"]["uid"]),
    ……]
es.add_relationships(relationships)
```

数据添加完成后，再使用relationship函数为不同表中相关数据（uid）添加关系；

credit_info.ipynb Python 复制代码

```
feature_matrix, features_defs = ft.dfs(
    entityset=es,
    target_entity='pri_star_info',
)
feature_matrix
print(type(feature_matrix))
feature_matrix.to_csv('./result.csv',sep=',',index=True,header=True)
```

最后使用 ft.dfs 函数（代表深度特征合成）提取数据特征，返回feature_matrix和features_defs，其中features_matrix表示特征矩阵，features_defs表示每个维度上的注释信息。打印feature_matrix及其类型信息，并将feature_matrix转换成csv文件保存。

3. 数据预处理

3.1. 缺失值

deal_data_credit.ipynb Python 复制代码

```
for column in df.columns.to_list():
    if df[column].dtype == 'object':
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        df[column].fillna(df[column].median(), inplace=True)
df
```

首先读取到文件中的所有列，并将列名转换为列表。

其次对于数据类型进行判断，如果是object类别型变量则使用众数填充，使用df[column].mode()[0]计算当前数值出现的次数，并且读取到第一个数据为众数；如果不是类别型变量，则使用中位数进行填充，使用df[column].median()计算当前列的中位数

3.2. 异常值处理

deal_data_credit.ipynb Python 复制代码

```python
for column in df.columns.to_list():
    if df[column].dtype != 'object' and column != 'credit_level':
        print(column)
        if(column == 'credit_level'):
            print("error")
        if(column == 'uid'):
            print("error")
        low = df[column].quantile(0.05)
        high = df[column].quantile(0.95)
        # 使用clip方法替换小于5%和大于95%的值
        df[column] = df[column].clip(lower=low, upper=high)
```

首先，如果当前列的数据类型不是 object且列名不为 'credit_level'，则执行以下代码：计算当前列的 5% 分位数，即数据中排在第 5% 位置的数值，将其赋值给变量 low。计算当前列的 95% 分位数，即数据中排在第 95% 位置的数值，将其赋值给变量 high。

接着，使用 clip 方法对当前列的数据进行剪裁，将小于 low 的值替换为 low，将大于 high 的值替换为 high。

3.3. 数据转换

deal_data_credit.ipynb Python 复制代码

```python
df['MODE(dm_v_tr_huanb_mx.tran_type)'] = pd.to_numeric(df['MODE(dm_v_tr_huanb_mx.tran_type)'],errors="coerce")
df['MODE(dm_v_tr_huanb_mx.tran_type)'] = df['MODE(dm_v_tr_huanb_mx.tran_type)'].astype(float)
print(df['MODE(dm_v_tr_huanb_mx.tran_type)'].unique())
df['MODE(dm_v_tr_huanb_mx.tran_type)'] = df['MODE(dm_v_tr_huanb_mx.tran_type)'].fillna(0.0)
print(df['MODE(dm_v_tr_huanb_mx.tran_type)'].unique())
```

对于数据进行转换处理，过程如下：

1 将该列的数据类型转换为数值型（numeric），如果无法转换则将其转为缺失值（coerce）。

2 将该列的数据类型转换为浮点型（float）。

3 输出该列不同的唯一值（unique）。

4 将该列中的缺失值填充为 0。

5 再次输出该列不同的唯一值，观察是否存在缺失值。

最终将**非数值的编码转换为数值类型**，同时对于缺失值进行处理

3.4. 数据标准化

deal_data_credit.ipynb　Python　复制代码

```python
scaler = StandardScaler()
df = pd.read_csv("./credit_encoded.csv",dtype=None)
df_columns = df.columns.to_list()
df_columns.remove('uid')
df_columns.remove('credit_level')
print(df_columns)
df[df_columns] = scaler.fit_transform(df[df_columns])
df.head()
```

代码通过 df_columns = df.columns.to_list() 将数据框中的列名保存到名为 df_columns 的列表中，并使用 df_columns.remove('uid') 和 df_columns.remove('credit_level') 从列表中删除了不需要处理的 'uid' 和 'credit_level' 两列。接下来，代码使用 StandardScaler 对数据框中除 'uid' 和 'credit_level' 以外的所有列进行标准化处理，并将处理后的结果保存回数据框中。

4. 特征工程（计算相关性）

4.1. 计算相关性

feature_credit　Python　复制代码

```python
df = pd.read_csv("./data/credit_final.csv")
corr_matrix = df.corr()
```

计算相关性主要使用pandas库中的corr函数，从之前生成的csv文件中生成相关系数矩阵

筛选（未采用）

Python | 复制代码

```python
to_drop = corr_series[abs(corr_series) >= 0.7].index.tolist()

drop = []
index = 0
for t in to_drop:
    if (t[0][0:4] == 'MEAN' and t[1][0:4] == 'MAX(') or (t[1][0:4] == 'MEAN' and t[0][0:4] == 'MAX('):
        if t[1][0:4] == 'MAX(':
            to_drop.pop(index)
            drop.append(t[1])
        else:
            to_drop.pop(index)
            drop.append(t[0])
    if (t[0][0:4] == 'MEAN' and t[1][0:4] == 'MIN(') or (t[1][0:4] == 'MEAN' and t[0][0:4] == 'MIN('):
        if t[1][0:4] == 'MIN(':
            drop.append(t[1])
            to_drop.pop(index)
        else:
            drop.append(t[0])
            to_drop.pop(index)
    index = index + 1

df.drop(columns=drop,axis=1,inplace=True)
df.to_csv("./data/credit_after_corr.csv")
```

（由于删去数据之后导致训练结果不好，最后选择没有删去）

筛选相关系数高于指定值的，并记录其列名），并且从dataframe中删除这些相关系数高于指定值的，得到处理后的数据，并保存到对应csv文件中。

4.2. 计算多重共线性VIF

计算vif

Python | 复制代码

```python
import pandas as pd
import statsmodels.api as sm

df = pd.read_csv("./credit_after_corr.csv")

df.drop(columns='Unnamed: 0',axis=1,inplace=True)
x_list = df.columns.to_list()
print(x_list)

from statsmodels.stats.outliers_influence import variance_inflation_factor

x_list.remove('Unnamed: 0.1')
x_list.remove('uid')
x_list.remove('credit_level')
X = df[x_list]
Y = df['credit_level']
model = sm.OLS(Y, X).fit()
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns

vif.to_csv("./credit_vif_1.csv")
```

计算VIF方差膨胀因子时，从statsmodels库中引入了variance_inflation_factor函数以计算VIF值，在计算前提前删除了无用的数据，计算时VIF结果存储在一个dataframe中，其中VIF Factor是计算出的值，features是列名。

筛选（未采用）

Python | 复制代码

```python
vif_threshold = 10
high_vif_features = list(vif[vif['VIF Factor'] > vif_threshold]["features"])

df.drop(columns=high_vif_features,axis=1,inplace=True)
df.to_csv('./credit_after_vif.csv')
```

（由于删去数据之后导致训练结果不好，最后选择没有删去）

并筛选VIF值高于指定值的，并记录其features（即列名），并且从dataframe中删除这些VIF值高于指定值的得到处理后的数据，并保存到对应csv文件中。

5. 模型评估

5.1. 客户星级

模型	准确率	精确率	召回率	F1分数	Cohen's Kappa系数
随机森林模型	0.9738573071604875	0.9740772053204735	0.9738573071604875	0.9739031748805760	0.9582329939228935
XGBoost模型	0.7379495272495462	0.9202710328280544	0.5432862868389917	0.5654842522505414	0.8731477277261196

5.2. 信用等级

模型	准确率	精确率	召回率	F1分数	Cohen's Kappa系数
随机森林模型	0.9633147603033224	0.9570262562690530	0.8455545355692532	0.8913416340143352	0.9203203653612596
XGBoost模型	0.8522306355918314	0.6773776797965199	0.5835956243077896	0.6156770921619316	0.7289625204956423