实验1：金融数据预处理实验指南

一. 实验目的

1. 掌握数据预处理的几个基本流程。
2. 学习Pandas、Numpy等软件包的使用，学会基于Pandas进行数据预处理。

二. 实验步骤

1. 准备工作

安装软件包（如果已经安装 Anaconda，则不需要此步骤）

```
 pip install numpy
pip install pandas
```

导入 Pandas 和 numpy 到代码中：

```
import pandas as pd
import numpy as np
```

为了加载 csv 格式的数据集，使用 read_csv 函数将数据读入 DataFrame：

```
data = pd.read_csv('data/dataset.csv')
```

使用 shape 属性来查看生成的 DataFrame 大小：

```
data.shape
```

输出：(50000, 73)

检查读入数据的基本结构，head() 方法打印输出前五行数据，让我们对读入的数据有一个大致的了解：

```
data.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X64	X65	X66
0	9.0	1458.0	17147.0	10.0	0.0	800.0	0.0	NaN	0.0	679.0	...	2375.0	7.0	581.0
1	2.0	250.0	38.0	6.0	NaN	10000.0	0.0	NaN	1.0	12990.0	...	-16204.0	31.0	796.0
2	2.0	1054.0	178.0	1.0	0.0	1000.0	0.0	NaN	1.0	18710.0	...	22436.0	230.0	732.0
3	10.0	1398.0	679.0	7.0	0.0	10000.0	0.0	NaN	1.0	19010.0	...	21431.0	11.0	36.0
4	2.0	1095.0	305.0	11.0	0.0	10000.0	0.0	NaN	2.0	16410.0	...	19064.0	93.0	395.0

5 rows × 73 columns

2. 处理缺失数据

可以使用多种填充数据空值的方法：

1. 忽略元组：当一条样本的属性缺失比例超过阈值，或者缺少 label 属性，则删除这行。
2. 全局常量：指定空值常量，默认为 np.NAN。

3. 平均值：使用属性的平均值填充空值。

获得每列数据的缺失值个数

```
missing_values_count = data.isnull().sum()
```

查看前十列的缺失值情况

```
missing_values_count[0:10]
```

```
X1       5851
X2        390
X3        817
X4       4280
X5       8891
X6       3461
X7       4825
X8      48466
X9       4280
X10       955
dtype: int64
```

默认值填充

缺失值填充为 –1

```
data.fillna(-1, inplace=True)
```

将所有值为 –1 的数据替换为 np.NaN

```
data.replace(to_replace=-1, value=np.NaN, inplace=True)
```

平均值填充

对于某些数值属性，可以使用全部样本该列的平均值来填充

```
data.X2.fillna(data.X2.mean(), inplace=True)
```

删除不完整的行

删除任何包含空值的行

```
data.dropna(inplace=True)
```

删除所有值都为空的行

```
data.dropna(how='all', inplace=True)
```

行数据中至少要有 5 个非空值

```
data.dropna(thresh=5, inplace=True)
```

去除 label 为空的行

```
data.dropna(subset=['Y'], inplace=True)
```

数据变换与离散化

缩放（Scaling）

找到所有数值属性

```
numeric_feats = data.dtypes[data.dtypes != "object"].index
```

使用最大最小值规范化

```
data[numeric_feats] = data[numeric_feats].apply(lambda x: (x - x.min()) / (x.max() - x.min()))
```

规范化（Normalization）

缩放（Scaling）只会改变数据的范围，而规范化会同时改变数据范围和分布，使之符合正态分布。
通常，如果要使用假设数据是正态分布的机器学习技术，那么就需要规范化数据，例如，线性回归、神经网络、LDA 等。

使用零均值规范化

```
data[numeric_feats] = data[numeric_feats].apply(lambda x: (x - x.mean()) / (x.std()))
```

离散化（Discretization）

将指定的连续属性分箱 binning 处理，包括等深分箱或等宽分箱。

等深分箱
等深分箱将数据集按记录行数分箱，每箱具有相同的记录数，每箱记录数称为箱子的深度：

```
data.X65_bin = pd.qcut(data.X65, q=10, duplicates='drop')
```

等宽分箱
等宽分箱使数据集在整个属性值的区间上平均分布，即每个箱的区间范围是一个常量，称为箱子宽度。 例如，将年份以五年为一个区间进行划分：

```
data.X66_bin = pd.cut(data.X66, bins=[100,200,300,400,500,600])
```

特征构造（交叉）

在分类模型的训练中，特征的交叉组合特别重要，某些特征经过关联之后，与label之间的相关性就会提高。比如，性别特征和年龄类别特征组合。
进行特征交叉前，一般将要交叉的连续型原始特征进行离散化处理。

```
def add_cross_feature(data, feature_1, feature_2):
    comb_index = data[[feature_1, feature_2]].drop_duplicates()
    comb_index[feature_1 + '_' + feature_2] = np.arange(comb_index.shape[0])
    data = pd.merge(data, comb_index, 'left', on=[feature_1, feature_2])
    return data
```

数据集切分

指定训练集和测试集比例对数据集进行划分

```
num_train = int(data.shape[0] * 0.8)
train_data = data[:num_train]
test_data = data[num_train:]
train_data.shape, test_data.shape
```

((40000, 73), (10000, 73))

保存处理后的数据集：

```
train_data.to_csv("train.csv")
test_data.to_csv("test.csv")
```

三. 提交内容

1. 实验报告：数据预处理中每个阶段的结果展示。
2. 实验代码：数据预处理的相关代码。