第二次上机作业

注意事项

1. 上机作业所需上传的文件，应打包并命名为"学号-姓名-上机#序号"，如"18000xxxxx-张三-上机2"，并在截止时间前上传到教学网。截止时间后仍开放提交，但会酌情扣分。
2. 上机作业为半自动评分。上传的压缩包，会根据压缩包名称自动分类，并提取压缩包名称中的有效信息。请尽可能保持压缩包命名符合规范。压缩包名中的短横线'-'可替换为下划线'_'、加号'+'，但不能是空格' '、长横线'—'。
3. 上机作业中的代码会自动测试。要求代码中读入的外部数据文件存放在代码文件上一级目录下的data文件夹下，生成的图表、数据文件等应在代码文件所在目录下。

```
root
  ├─workdir (Compress this directory and upload it)
  |     ├─code.py (Required)
  |     ├─data.csv (Required)
  |     ├─chart.png (Optional)
  |     └─report.pdf (Required)
  └─data (DO NOT upload this directory)
        └─dataset.xlsx
```

4. 题目中要求生成的图表和计算的数据均列在报告中。
5. 题目中要求计算的数据，代码运算得出结果后，应输出到控制台。

1. 逻辑回归是解决二分类问题的一种简便、高效的方法。现有一保险公司希望预测不同客户群体是否有意愿购买汽车保险。

现给出数据集`Insurance_Prediction.csv`，其中包含字段：

- 序号（id）
- 性别（Gender）
- 年龄（Age）
- 是否有驾照（Driving_License）
- 区域码（Region_Code）
- 之前是否有购买汽车保险（Previously_Insured）
- 汽车年限（Vehicle_Age）
- 汽车是否曾损坏（Vehicle_Damage）
- 年保险费（Annual_Premium）
- 销售渠道（Policy_Sales_Channel）
- 服务客户天数（Vintage）
- 客户是否对汽车保险感兴趣（Response）

（1）**绘制柱状图+扇形图**，分析以下因素与客户购买汽车保险倾向之间的关系：

（a）性别

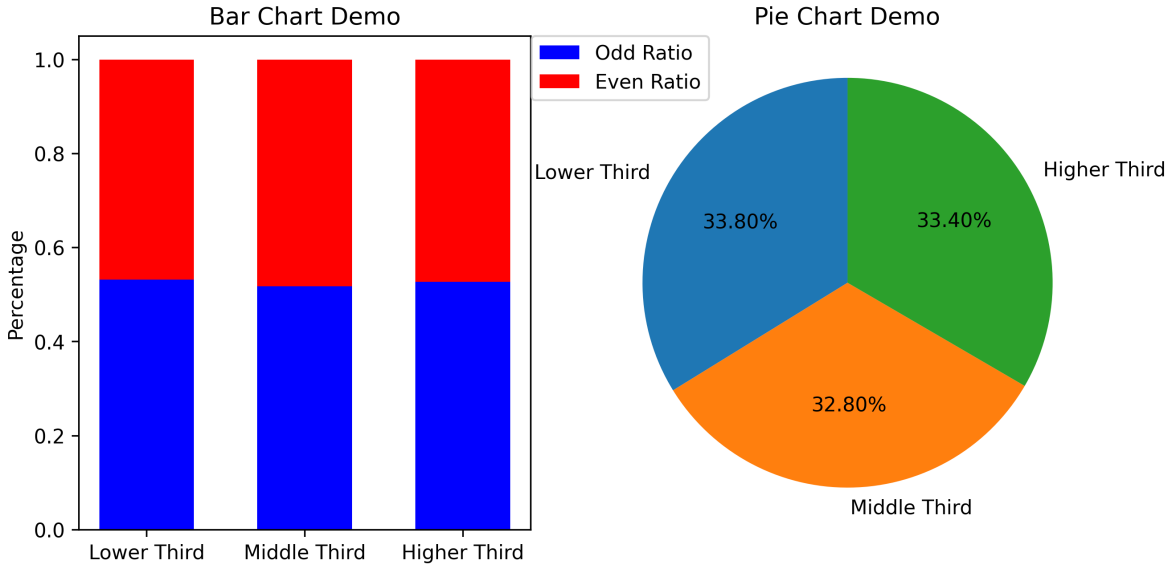
（b）汽车年限

（c）之前是否有购买汽车保险

（d）汽车是否曾损坏

绘制柱状图+扇形图示例代码：

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 10/25/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import numpy as np
import matplotlib.pyplot as plt

N = 1000   # Sample size
lowlimit, upplimit = 0, 10000   # Data range, upper limit can not be drawn
np.random.seed(20201026)   # Set random seed
data = np.random.randint(lowlimit, upplimit, N)   # Generate integers in range
[lowlimit, upplimit)

one_third = lowlimit + round((upplimit - lowlimit) / 3)
two_thirds = upplimit - round((upplimit - lowlimit) / 3)

# Divide data series into three parts
lower_third = data[data < one_third]
middle_third = data[(data >= one_third) & (data < two_thirds)]
higher_third = data[data >= two_thirds]

odd_ratios, even_ratios = [], []
part_sizes = []
parts = [lower_third, middle_third, higher_third]
for part in parts:
    part_sizes.append(part.size)
    even_ratio = lower_third[lower_third %2 == 0].size / part.size
    odd_ratios.append(1 - even_ratio)
    even_ratios.append(even_ratio)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9., 4.5), dpi=320)

xlocs = np.arange(0.5, 3.5)
ax1.bar(xlocs, odd_ratios, 0.6, 0., color="blue", label="Odd Ratio")
ax1.bar(xlocs, even_ratios, 0.6, odd_ratios, color="red", label="Even Ratio")
ax1.set_title("Bar Chart Demo")
ax1.set_ylabel("Percentage")
ax1.set_xticks(xlocs)
ax1.set_xticklabels(["Lower Third", "Middle Third", "Higher Third"])
ax1.legend(bbox_to_anchor=(.98, 1.02))

ax2.pie(part_sizes, labels=["Lower Third", "Middle Third", "Higher Third"],
autopct="%.2f%%", startangle=90)
ax2.set_title("Pie Chart Demo")
ax2.axis('equal')   # Draw pie chart as a circle

plt.savefig("./sample_bar_pie_charts.png", bbox_inches="tight")
```



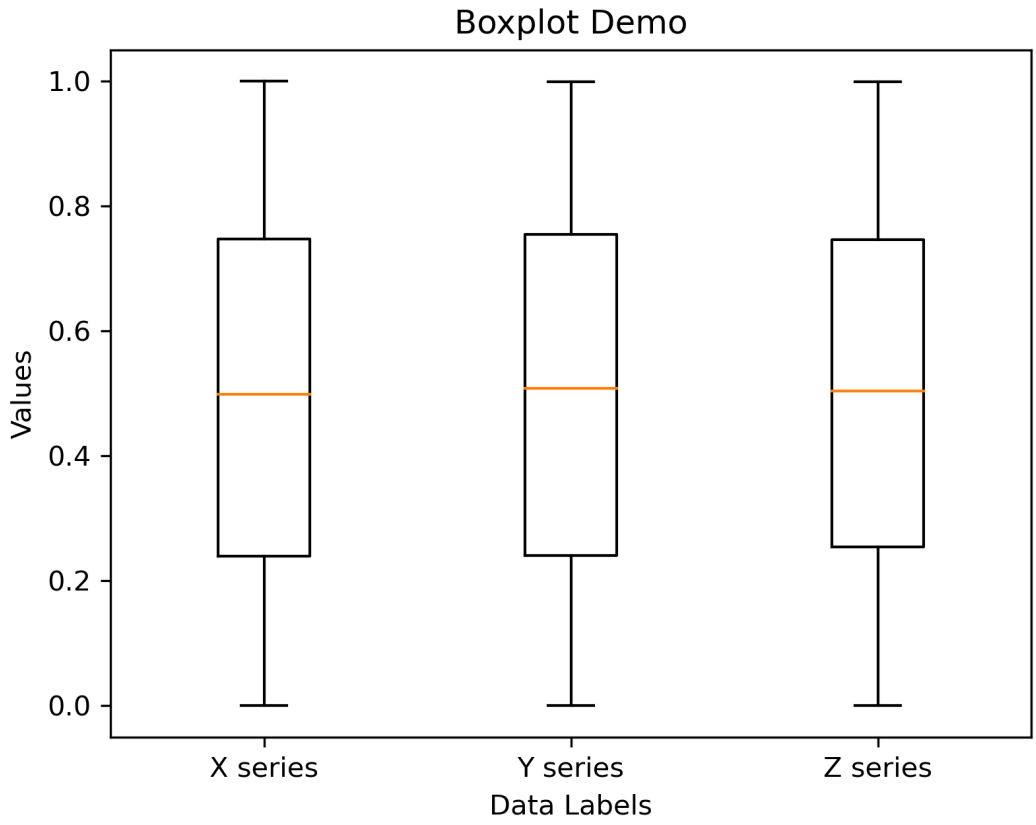
(2) **绘制箱线图**，分析以下因素与客户购买汽车保险倾向之间的关系：

(a) 年龄

(b) 年保险费

绘制箱线图示例代码：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 10/25/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import matplotlib.pyplot as plt
import numpy as np

N = 1000   # Sample size
np.random.seed(20201026)   # Set random seed
data = np.random.random((N, 3))   # Generate random data sample with 3 series

plt.figure(figsize=(6, 4.5), dpi=300)
plt.boxplot(data, labels=["X series", "Y series", "Z series"])
plt.title("Boxplot Demo")
plt.xlabel("Data Labels")
plt.ylabel("Values")
plt.savefig("./sample_boxplot.png")
```



（3）将数据集划分为训练集`train.csv`和验证集`test.csv`。分析划分好的训练集和验证集，**分别给出训练集和验证集中的下列数据：**

（a）男性和女性客户的比例

（b）年龄的最小值、最大值、平均值、中位数

（c）有驾照的比例

（d）之前有购买汽车保险的比例

（e）汽车年限分别在1年以下、1~2年、2年以上的比例

（f）汽车曾经损坏的比例

（g）年保险费的最小值、最大值、平均值、中位数

划分数据集示例代码

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 10/25/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import numpy as np
from sklearn.model_selection import train_test_split

N = 1000   # Sample size
lowlimit, upplimit = 0, 10000   # Data range, upper limit can not be drawn
```

```
data = np.random.randint(lowlimit, upplimit, N)   # Generate integers in range
[lowlimit, upplimit)

train_data, test_data = train_test_split(data)
print(f"Train dataset size = {train_data.size}")
print(f"Test dataset size = {test_data.size}")
"""
Train dataset size = 750
Test dataset size = 250
"""
```

（4）在训练一个机器学习模型之前，我们往往需要确定哪些特征是重要的。对于一些简单问题，我们可以根据先验性的知识选取合适的特征，但有些情况下，特征数太多或是我们对问题的相关知识缺乏了解。我们可以用一些简单的方法筛选出较为重要的一些特征。

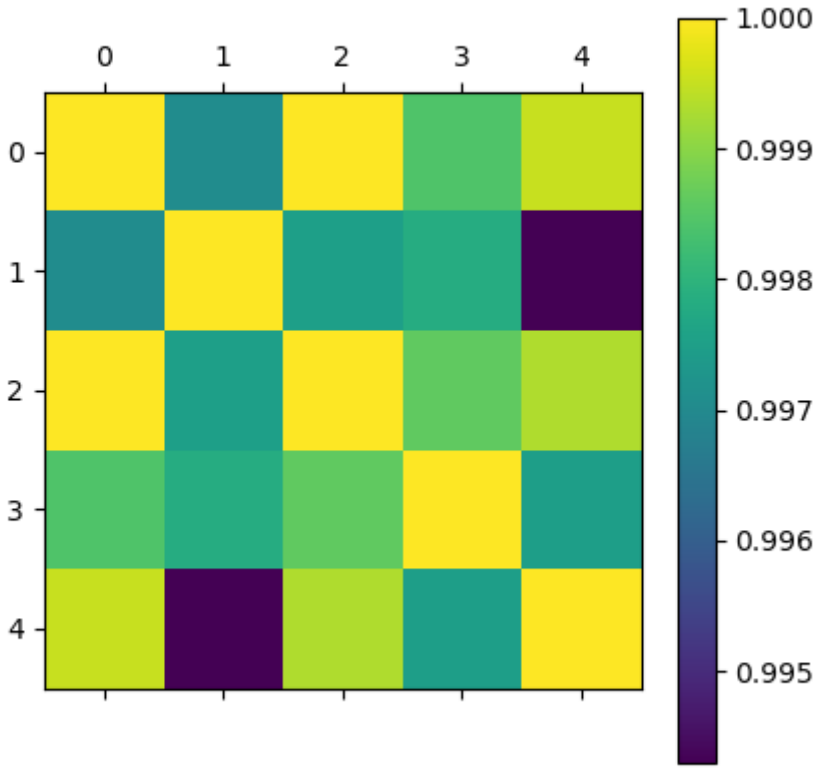
- 去除小方差的特征

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 11/22/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import numpy as np

N = 1000   # Sample size
nfeature = 10   # Number of features
np.random.seed(20201122)
X = np.random.random((N, nfeature))
for i in range(nfeature):
    X[:, i] = X[:, i] > (0.05 + i * 0.10)
X = np.asarray(X, dtype=int)

# 0-1 features are Bernoulli random variables, Variance = p(1-p)
# Remove features that more than 90% of samples are all 0 or 1
# The first and last feature will be removed
from sklearn.feature_selection import VarianceThreshold
sel_X = VarianceThreshold(threshold=(.9 * (1 - .9))).fit_transform(X)
print(sel_X.shape)   # (1000, 8)
# OR
sel_X = X[:, X.var(axis=0) > .9 * (1 - .9)]
print(sel_X.shape)   # (1000, 8)
```

- 基于Pearson相关系数选取相关性强的特征

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 11/22/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''
```

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import pearsonr
from sklearn.datasets import load_iris

# Load iris dataset
X, y = load_iris(return_X_y=True)
data = np.hstack([X, y.reshape(-1, 1)])
nfeature = data.shape[1]

# Compute Pearson correlation coefficient matrix
# It should be a symmetric matrix, so only compute the upper half
corrmat = np.ones((nfeature, nfeature), dtype=float)
for i in range(nfeature):
    for j in range(i + 1, nfeature):
        corrmat[j, i] = corrmat[i, j] = pearsonr(data[i], data[j])[0]

# Display Pearson correlation coefficient matrix with color bar
plt.figure(figsize=(6, 4.5), dpi=320)
plt.matshow(corrmat)
plt.colorbar()
plt.savefig('./pearson_corr.png')
```



- 基于χ^2检验选取相关性强的特征

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
```

```
    Author: Pengbo Song
    Date created: 11/22/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest, chi2

# Load iris dataset
X, y = load_iris(return_X_y=True)
print(X.shape)   # (150, 4)

# Select the best k features based on chi squared test
sel_X = SelectKBest(chi2, k=2).fit_transform(X, y)
print(sel_X.shape)   # (150, 2)
```

这里只提供最简单的筛选特征的方法。在scikit-learn的文档中（https://scikit-learn.org/stable/modules/feature_selection.html）有给出更多选取特征的方法，相关API以及示例代码，有兴趣的同学可以自行了解，这里不做要求。

选取你认为重要的特征，构建合适的逻辑回归模型。优化模型参数，**给出你认为预测效果最好的模型。**将模型对训练集和测试集的预测结果分别保存为`train_predict.csv`和`test_predict.csv`，**并计算模型在训练集和验证集上的预测准确率。**

训练集/测试集预测结果文件示例：

```
id,Response
381110,0
381111,0
381112,0
```

（5）对于二分类问题，如果样本不平衡，使用准确率作为评估模型的指标效果很差，而混淆矩阵（confusion matrix）可以很好描述模型的预测效果，可作为分类模型评估的重要依据。**请给出（4）中模型在训练集和验证集上的混淆矩阵，并计算准确率（accuracy）、精度（precision）、召回率（recall）、F-score和AUC。**

计算混淆矩阵、准确率、精度、召回率、F-score和AUC示例代码：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 10/26/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import numpy as np
from sklearn import metrics

N = 1000   # Sample size
np.random.seed(20201026)   # Set random seed
y_true = np.asarray(np.random.random(N) > 0.5, dtype=int)   # Generate random
true labels with half 0 and half 1
y_pred = np.asarray(np.random.random(N) < 0.5, dtype=int)   # Generate random
predicted labels with half 0 and half 1
```

```
cm = metrics.confusion_matrix(y_true, y_pred)
acc = metrics.accuracy_score(y_true, y_pred)
recall = metrics.recall_score(y_true, y_pred)
precision = metrics.precision_score(y_true, y_pred)
f1 = metrics.f1_score(y_true, y_pred)
auc = metrics.roc_auc_score(y_true, y_pred)
print(f"""Confusion Matrix
{cm}
Accuracy = {acc:>.3f}
Precision = {precision:>.3f}
Recall = {recall:>.3f}
F1 = {f1:>.3f}
AUC = {auc:>.3f}""")
"""
Confusion Matrix
[[226 261]
 [263 250]]
Accuracy = 0.476
Precision = 0.489
Recall = 0.487
F1 = 0.488
AUC = 0.476
"""
```

（6）将文件打包上传，压缩包中应有文件

- Python源代码 `logistics_classfication.py`
- 划分好的训练集 `train.csv` 和验证集 `test.csv`
- 模型在训练集和测试集的预测结果 `train_predict.csv` 和 `test_predict.csv`
- 分析报告 `*.doc/*.docx/*.pdf`

（如果使用Markdown写报告，请将报告转为pdf格式）

重要提示：

上机作业中，参考模板类合理组织代码，在编写代码时尽可能保持代码良好的可阅读性

代码自动测试时，代码文件 `logistics_classfication.py` 中应有类 `MLChemLab2`，类中要求有以下方法：

```
add_model(model : str, **kwargs)
```

用于指定拟合使用的模型类型，和模型所使用的超参数。

方法参数：

`model`：用于指定模型的关键词。至少应支持关键词 `logistic`，初始化一个逻辑回归模型。

`kwargs`：用于初始化模型的关键词参数。对scikit-learn中的逻辑回归模型，可指定的关键词参数包括：

- `solver`：模型使用的优化算法。默认指定为L-BFGS算法，`solver=lbfgs`。
- `penalty`：损失函数的范数。默认指定为L2范数，`penalty=l2`。
- `C`：正则化强度因子，`C` 越小，正则化强度越高。
- `class_weight`：数据集每个标签类的权重原子，默认每个类的权重均为1。

逻辑回归模型关键词参数化的详细说明见https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

返回值：`None`

```
fit(X : np.ndarray[2d, float], y : np.ndarray[1d, float], featurization_mode :
str)
```

用于指定对输入X的特征化方法，并用特征化处理后的X和y训练模型。

方法参数：

`X`：二维数组，数组的每一行对应于每个样本的各个特征。

`y`：一维数组，为x中每个样本对应的标签，长度与x的行数一致。

`featurization_mode`：指定X的特征化方法。至少应支持关键词`normalization`，对应于对X进行标准化及其他预处理的方法。

返回值：训练后的模型。

异常处理：如果模型未初始化，则抛出警告并返回`None`。

```
predict(X : np.ndarray[2d, float])
```

用训练好的模型给出给定X的预测结果。

方法参数：

`X`：二维数组，数组的每一行对应于每个样本的各个特征。

返回值：基于给定X预测得到的y，应为一维数组，长度与`X`的行数一致。

异常处理：如果模型未初始化，则抛出警告并返回`None`。

```
evaluation(y_true : np.ndarray[1d, float], y_pred : np.ndarray[1d, float], metric
: str)
```

基于真实的y和预测的y对模型表现进行评估。

方法参数：

`y_true`：一维数组，为真实的y。

`y_pred`：一维数组，为预测得到的y。

`metric`：指定预测方法的关键词。应支持以下参数：

- `accuracy`：计算准确率。
- `precision`：计算精密度。
- `recall`：计算召回率。
- `F1`：计算精密度和召回率的调和平均值。
- `CM`：计算混淆矩阵。
- `AUC`：计算ROC曲线下的面积。

返回值：不同评估方法的计算结果。