第二次上机作业

一、通过 sklearn 中的 LogisticRegression，练习自学和使用 Python 网络开源包

1. 学习 sklearn 实现的原理

在网站 https://scikit-learn.org/stable/index.html 中找到 LogisticRegression 的简介，以及详细的函数用法，回答如下问题：

（1） sklearn 为 LogisticRegression 提供了哪几种正则化方法？公式分别是什么？

（2） 上述公式和课件中、周志华《机器学习》与《Python 机器学习》中有什么不同？仔细看一看官网中的公式说明，你能找到原因吗？

其中，周志华《机器学习》书中内容如下：

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{m} \left(-y_i \boldsymbol{\beta}^{\mathrm{T}} \hat{\boldsymbol{x}}_i + \ln \left(1 + e^{\boldsymbol{\beta}^{\mathrm{T}} \hat{\boldsymbol{x}}_i} \right) \right)$$

《Python 机器学习》书中内容如下：

$$J(\boldsymbol{w}) = -\sum_{i} y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \phi\left(z^{(i)}\right)\right)$$

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

（3） 上述公式中，C 是什么含义？C 越大，意味着什么？越小，意味着什么？

（4） 如何在命令中选择或关闭正则化？

（5） sklearn 为 LogisticRegression 提供了哪几种优化方法？请解释每种方法的特点和适用场合（根据官方文档进行理解，并用自己的话解释即可）。

（6） 如何在命令中选择优化方法？

（7） fit_intercept 参数是什么意思？如果设为 False，对模型有什么影响？

（8） 哪几个参数可以设置结束条件？其中哪个控制最多的迭代轮数？哪个控制梯度阈值？

（9） class_weight 参数是什么意思？你认为主要应用于什么样的数据集？为什么？

（10） 如何输出各个变量的权重？

（11） 在多分类问题中，LogisticRegression 默认用的是哪种方式？由哪个参数调节？

2. 利用 sklearn 的 LogisticRegression 进行分类

（1） 加载鸢尾花数据集（数据集和数据集说明见所附文件 iris.*）

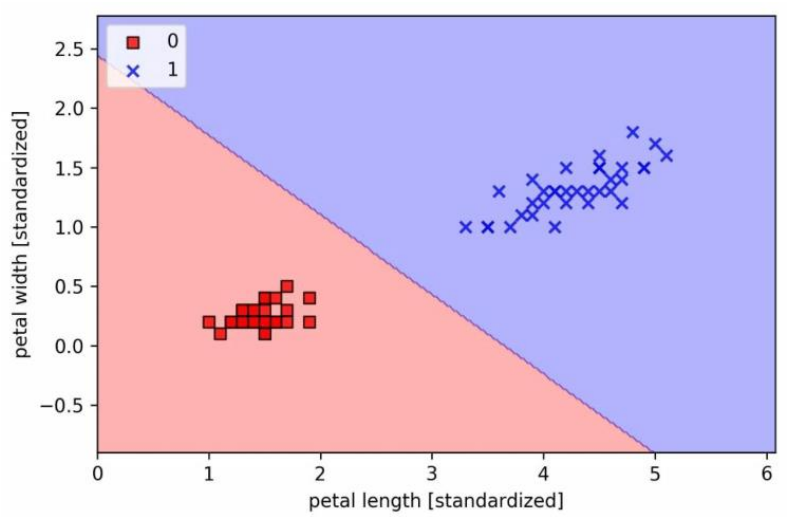
注意：

- 鸢尾花数据集的列属性请自行在数据说明文件中找，并将属性名赋到 dataframe 中。
- 鸢尾花一共三种，请将其编码为{0,1,2}。
- 如有必要，请将 feature 进行标准化。

（2） 在二分类问题上，拟合模型并画出决策区域（代码参照多分类问题；决策区域大致如下）。

两类：Iris-setosa、Iris-versicolor

使用属性：花萼长度、花萼宽度



（3） 在多分类问题上，拟合模型并画出决策区域（代码和决策区域近似如下）。

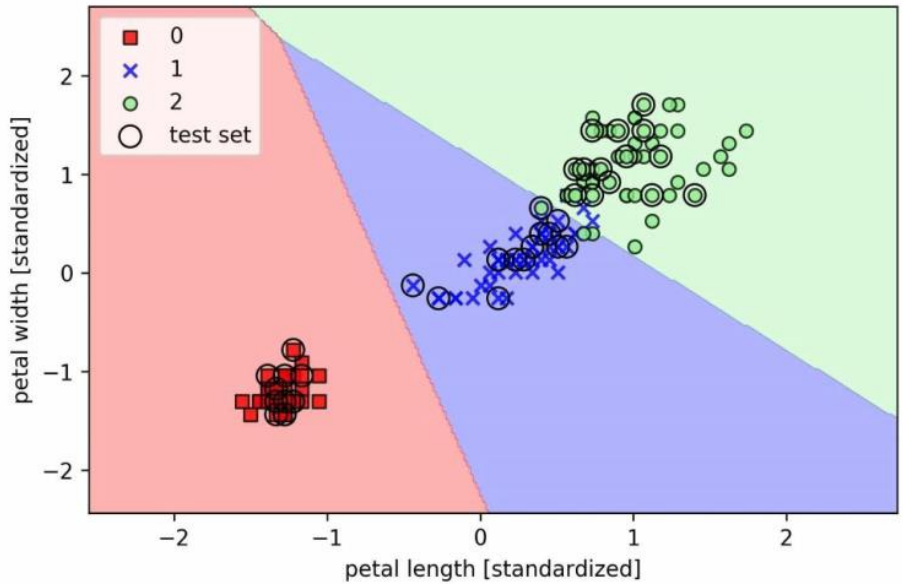
注意：

- **为避免单纯重复代码，请解释每一条语句中各参数的含义，写在作业报告中。**

三类：全部三种鸢尾花

使用属性：花萼长度、花萼宽度

```
>>> from sklearn.linear_model import LogisticRegression
>>> lr = LogisticRegression(C=100.0, random_state=1)
>>> lr.fit(X_train_std, y_train)
>>> plot_decision_regions(X_combined_std,
...                       y_combined,
...                       classifier=lr,
...                       test_idx=range(105, 150))
>>> plt.xlabel('petal length [standardized]')
>>> plt.ylabel('petal width [standardized]')
>>> plt.legend(loc='upper left')
>>> plt.show()
```



（4） 探索不同的正则化方法、和不同的 C 值对权重的影响，并解释结果图（以下代码和结果图为参考。）

注意：

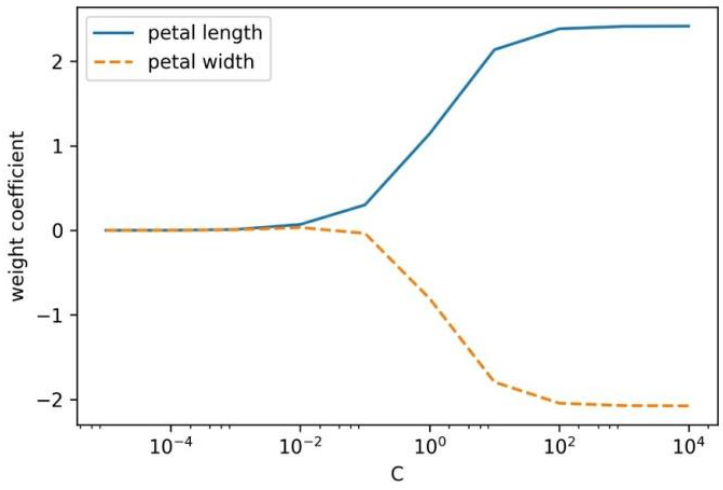
- 以下代码未选择正则化方法，作业请自行选择正则化方法，分布绘制 L1、L2 正则化情况下不同 C 影响权重的结果图。
- 为避免单纯重复代码，请解释每一条语句中各参数的含义，写在作业报告中。

三类：全部三种鸢尾花

使用属性：花萼长度、花萼宽度

导出 weight：由于是 OvR，只展示 Iris-versicolor vs Rest 的模型权重即可。

```
>>> weights, params = [], []
>>> for c in np.arange(-5, 5):
...     lr = LogisticRegression(C=10.**c, random_state=1)
...     lr.fit(X_train_std, y_train)
...     weights.append(lr.coef_[1])
...     params.append(10.**c)
>>> weights = np.array(weights)
>>> plt.plot(params, weights[:, 0],
...          label='petal length')
>>> plt.plot(params, weights[:, 1], linestyle='--',
...          label='petal width')
>>> plt.ylabel('weight coefficient')
>>> plt.xlabel('C')
>>> plt.legend(loc='upper left')
>>> plt.xscale('log')
>>> plt.show()
```


petal length
petal width
weight coefficient
2
1
0
−1
−2
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
C

二、自学并使用 sklearn 中的线性回归、岭回归、LASSO 和弹性网络。

1. 在 sklearn 官网上分别找到（1）线性回归、（2）岭回归、（3）LASSO 和（4）弹性网络的实现，并写出 sklearn 实现中所使用的公式和各部分的含义。

2. 学习探索性数据分析。

（1）加载住房数据集（见 housing.data.txt）

列定义如下：

```
>>> df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS',
...               'NOX', 'RM', 'AGE', 'DIS', 'RAD',
...               'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
```

数据说明如下：

- `CRIM`: Per capita crime rate by town
- `ZN`: Proportion of residential land zoned for lots over 25,000 sq. ft.
- `INDUS`: Proportion of non-retail business acres per town
- `CHAS`: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `NOX`: Nitric oxide concentration (parts per 10 million)
- `RM`: Average number of rooms per dwelling
- `AGE`: Proportion of owner-occupied units built prior to 1940
- `DIS`: Weighted distances to five Boston employment centers
- `RAD`: Index of accessibility to radial highways
- `TAX`: Full-value property tax rate per $10,000
- `PTRATIO`: Pupil-teacher ratio by town
- `B`: 1000(Bk - 0.63)^2, where *Bk* is the proportion of [people of African American descent] by town
- `LSTAT`: Percentage of lower status of the population
- `MEDV`: Median value of owner-occupied homes in $1000s

（2）分析哪些因素与预测目标最相关

预测目标：MEDV（房价中位价）

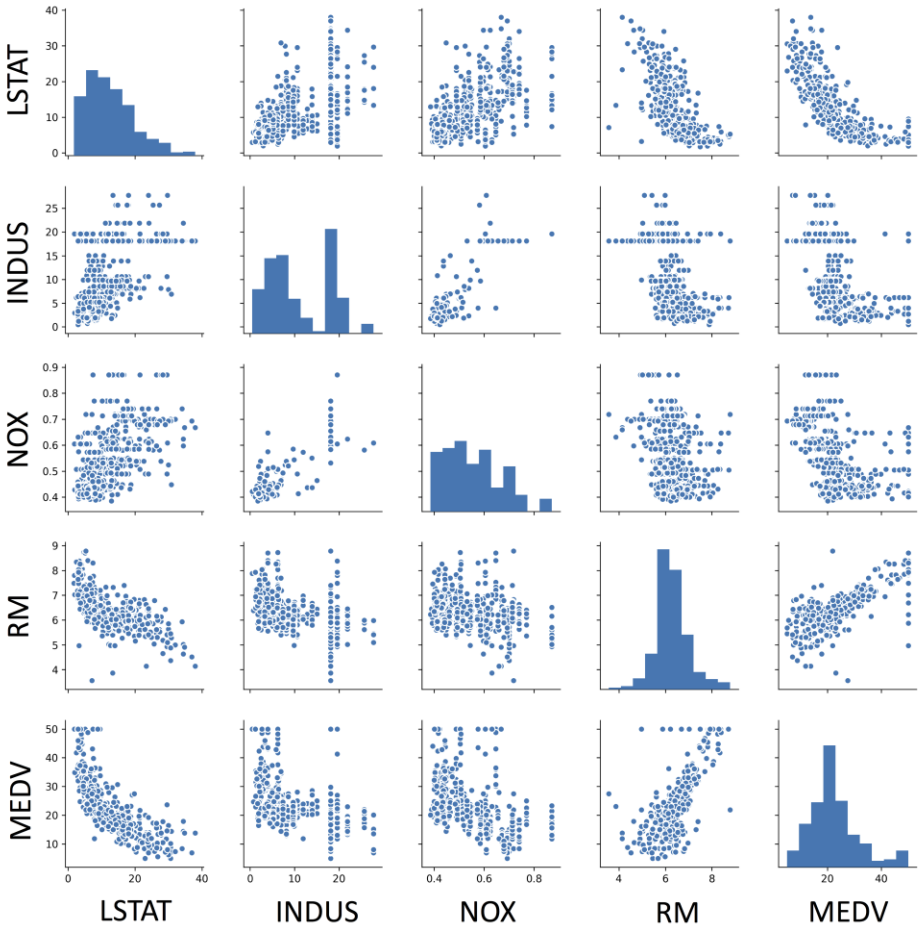
以下代码可以用图示的方式，展示四个因素与房价之间的相关情况。这段代码只分析了 4 个因素，请你模仿这段代码，把所有你认为可能与房价有关的因素都分析一下。注意！因素之间可以不一一分析相关性，重点在它们与房价的关系。

附代码：

You can install the `seaborn` package via `conda install seaborn` or `pip install seaborn`. After the installation is complete, you can import the package and create the scatterplot matrix as follows:

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> cols = ['LSTAT', 'INDUS', 'NOX', 'RM', 'MEDV']
>>> sns.pairplot(df[cols], size=2.5)
>>> plt.tight_layout()
>>> plt.show()
```

得到如下图：



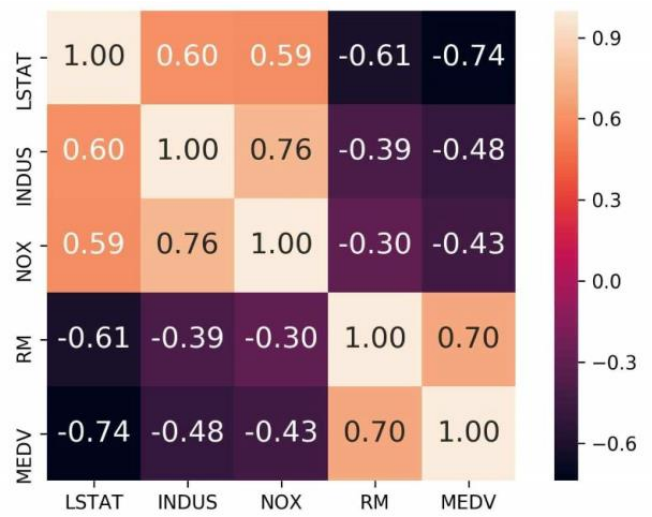
（3）用关联矩阵查看关系

仿照以下代码，分析所有因素之间的相关性，形成一个 14x14 的大矩阵。

In the following code example, we will use NumPy's `corrcoef` function on the five feature columns that we previously visualized in the scatterplot matrix, and we will use Seaborn's `heatmap` function to plot the correlation matrix array as a heat map:

```
>>> import numpy as np
>>> cm = np.corrcoef(df[cols].values.T)
>>> sns.set(font_scale=1.5)
>>> hm = sns.heatmap(cm,
...                  cbar=True,
...                  annot=True,
...                  square=True,
...                  fmt='.2f',
...                  annot_kws={'size': 15},
...                  yticklabels=cols,
...                  xticklabels=cols)
>>> plt.show()
```

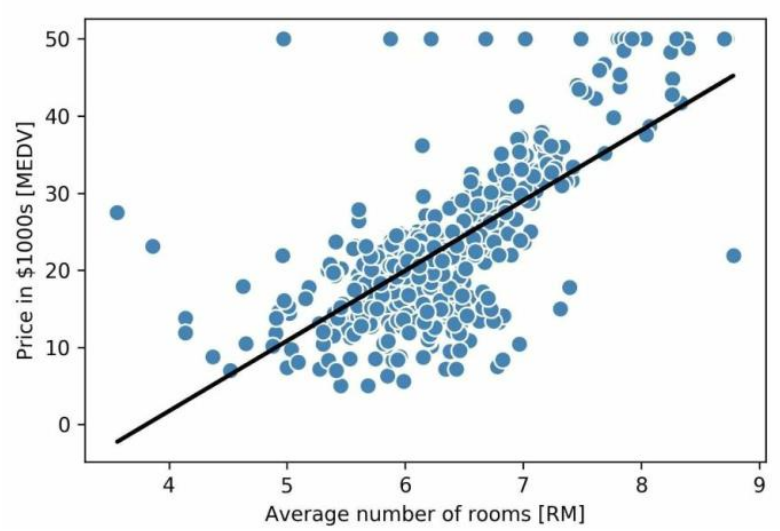
As we can see in the resulting figure, the correlation matrix provides us with another useful summary graphic that can help us to select features based on their respective linear correlations:



（4）练习简单线性回归，并绘图

选取 RM 作为因素，分析其与房价的回归线，画出类似如下的图。

请付上你的代码。



（4）练习多元回归。

从房价数据集中，随机抽取 90%作为训练集，10%作为测试集。

分别用（1）线性回归、（2）岭回归、（3）LASSO 和（4）弹性网络，进行如下分析。

- 在训练集上拟合模型
- 在训练集和测试集上进行预测
- 分别输出训练集和测试集的 MSE。（对于有正则项的回归方法，请将正则化强度的参数至少取 5 个不同的值，观察结果的变化）

上述四种方法的使用方法，请在 sklearn 官网上自学。

（4）练习多项式回归

参考以下代码，以 LSTAT 为因素变量，分析其与房价的二次关系。

重点在于练习将普通特征转化为多项式特征。

附参考代码及参考图：

1. Add a second degree polynomial term:

```
from sklearn.preprocessing import PolynomialFeatures
>>> X = np.array([ 258.0, 270.0, 294.0, 320.0, 342.0,
...               368.0, 396.0, 446.0, 480.0, 586.0])\
...               [:, np.newaxis]
>>> y = np.array([ 236.4, 234.4, 252.8, 298.6, 314.2,
...                342.2, 360.8, 368.0, 391.2, 390.8])
>>> lr = LinearRegression()
>>> pr = LinearRegression()
>>> quadratic = PolynomialFeatures(degree=2)
>>> X_quad = quadratic.fit_transform(X)
```

2. Fit a simple linear regression model for comparison:

```
>>> lr.fit(X, y)
>>> X_fit = np.arange(250,600,10)[:, np.newaxis]
>>> y_lin_fit = lr.predict(X_fit)
```

3. Fit a multiple regression model on the transformed features for polynomial regression:

```
>>> pr.fit(X_quad, y)
>>> y_quad_fit = pr.predict(quadratic.fit_transform(X_fit))
```

4. Plot the results:

```
>>> plt.scatter(X, y, label='training points')
>>> plt.plot(X_fit, y_lin_fit,
...          label='linear fit', linestyle='--')
>>> plt.plot(X_fit, y_quad_fit,
...          label='quadratic fit')
>>> plt.legend(loc='upper left')
>>> plt.show()
```