机器学习课程报告：作业 1

一、实验简介

该实验室主要基于鸢尾花数据集，首先对数据分布进行了分析，接着分别使用 logistic 回归、决策树和神经网络（MLP 分类器）对数据进行了训练和预测，并使用 precision、recall 和 f1-score 衡量模型预测的效果。

编程语言：Python

Python 版本：python3.6

二、数据集

本实验使用的数据集是鸢尾花（Iris）数据集（http://archive.ics.uci.edu/ml/index.php），也是 2007 年以来最流行的机器学习数据集。鸢尾花数据集包含 150 个鸢尾花的信息，每 50 个鸢尾花取自三种鸢尾花之一：Setosa、Versicolour 和 Virginica，每种花的特性用下面 4 种属性描述：

- 萼片长度（sepal length）
- 萼片宽度（sepal width）
- 花瓣长度（patal length）
- 花瓣宽度（petal width）

数据集格式如下图：每行数据代表一个样本信息，其中前 4 个数值分别表示鸢尾花的萼片长度、萼片宽度、花瓣长度和花瓣宽度，最后一个字符串表示鸢尾花的类别。

```
1  5.1,3.5,1.4,0.2,Iris-setosa
2  4.9,3.0,1.4,0.2,Iris-setosa
```

图 1. 数据集格式

该数据集已经集成到了 sklearn 中，在 python 中通过以下命令便可以直接调用此数据集：

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
```

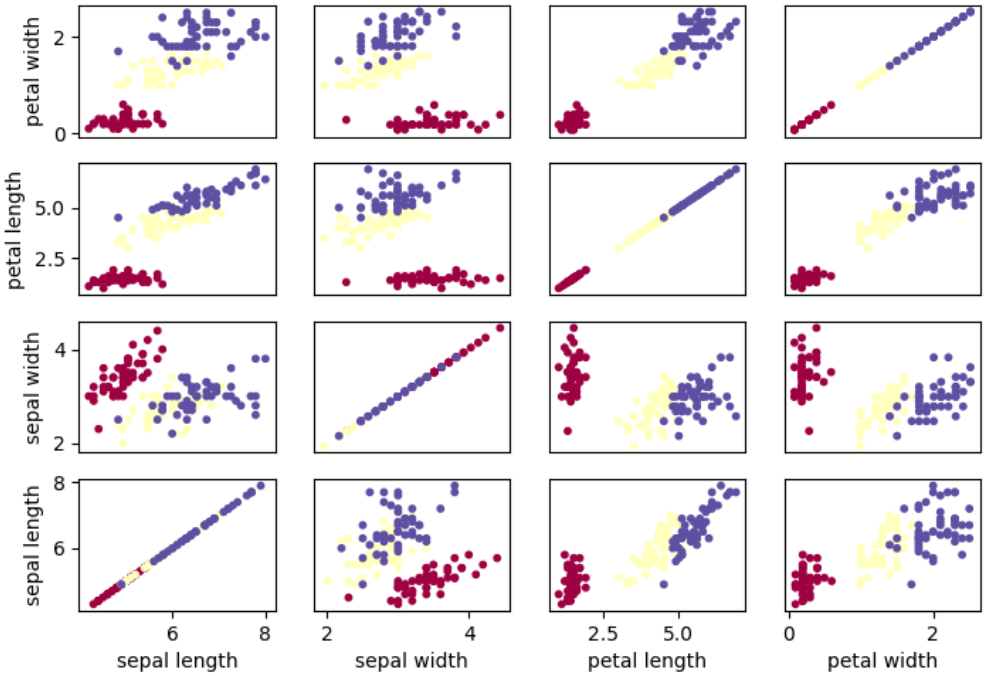
为了进一步了解数据特征，本次实验使用手动处理数据集并对其划分成训练集合测试集。代码如下：

```
def readData(filepath):
    '''
    :param: filepath: the path of samples
    :return: iris_matrix_X：a matrix of attributes
             iris_array_y: an array of label
    '''
    iris_matrix_X = np.zeros(150*4).reshape((150, 4))    # feature data
    iris_array_y = np.zeros(150)    # label, an array
    with open(filepath, encoding='UTF-8') as f:
        for i in range(0, 150):
```

```
            row = f.readline()
            elememts = row.split(",")      # split the 5 elements of one row
            iris_matrix_X[i, :] = [float(elememts[index]) for index in range(0, 4)]
            map = {"Iris-setosa": 1, "Iris-versicolor": 2, "Iris-virginica": 3}
            iris_array_y[i] = map.get(elememts[4].strip())    # get rid of the '\n' after ir
is label.
    return iris_matrix_X, iris_array_y
```

首先，将鸢尾花数据集中的每行读出，将属性放在 iris_matrix_X 中，对应的 label 将其转化为类别标签后放在 iris_array_y 中，返回值 iris_matrix_X 是一个 150*4 的矩阵，表示所有鸢尾花样本的属性信息，iris_array_y 是一个 150 维的 array，表示 150 个鸢尾花的类别。

除此之外，我分析了鸢尾花的数据分布，如下图：



由于数据具有 4 个维度，因为我们两两观察属性间的分布，从上图可以看出两两属性之间的分界线还是比较明显的，所以使用 Logistic 回归、决策树和神经网络应该都是可行的。下面便具体分析这三种算法的效果。

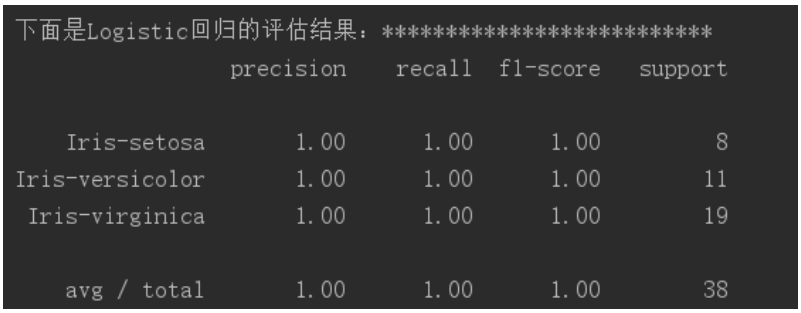
三、Logistic 回归算法

基于前面预处理得到的属性集和标签集，我们对其使用 Logistic 回归算法，首先将训练集合测试集划分，此处训练集和测试集的比例为 3:1，即有 37.5≈38 个测试数据，我使用此测试数据评估训练集训练得到的 Logistic 回归模型。

```
def logisticRegression(iris_X, iris_y):
    '''
    :param iris_X: training dataset
    :param iris_y: testing dataset
    :return: evaluation_result, evaluation matrics, including precision, recall and f1-score
    '''
```

```
    X_train, X_test, y_train, y_test = train_test_split(iris_X, iris_y, test_size=0.25,
random_state=33)  # split training set and testing set
    lr = LogisticRegression()
    lr.fit(X_train, y_train)  # training datasets
    lr_y_predict = lr.predict(X_test)  # testing samples
    evaluation_result = classification_report(lr_y_predict, y_test,
                                              target_names=["Iris-setosa", "Iris-versicolor", "Iris-
virginica"])  # generating evaluation report automatically
    return evaluation_result
```

模型的输出结果如下：



从上图可知，划分得到的 38 个测试样本中，类别为 Setosa、versicolor 和 virginica 的鸢尾花数目分别为：8、11、19，且预测结果无论是 precision、recall 还是 f1-score 都是 1，因此说明训练集训练得到的模型在测试集上的表现仍然是非常完美，所以 Logistic 回归在此数据集上是有效的。

注意：一般而言，为了提高效率，我们在训练数据之前应该对数据首先进行归一化，这里之所以没有进行归一化操作，是因为使用默认的 Logistic 回归模型在归一化之后，效果会变差，无论是哪个衡量指标，其效果都有所下降。

四、决策树算法

决策树算法代码的实现大体和 Logistic 回归一致，其代码如下：

```
def DecisionTree(iris_X, iris_y):
    '''
    :param iris_X: training dataset
    :param iris_y: testing dataset
    :return:  evaluation_result, evaluation matrics, including precision, recall and f1-score
    '''
    X_train, X_test, y_train, y_test = train_test_split(iris_X, iris_y, test_size=0.25,
random_state=33)  # split training set and testing set
    tree = DecisionTreeClassifier()
    tree.fit(X_train, y_train)      # train data
    tree_y_predict = tree.predict(X_test)     # test samples
    evaluation_result = classification_report(tree_y_predict, y_test,
                                              target_names=["Iris-setosa", "Iris-versicolor", "Iris-
virginica"])     # generating evaluation report automatically
    return evaluation_result
```

代码结果如下：

```
下面是决策树的评估结果：**************************
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      0.73      0.85        15
 Iris-virginica       0.79      1.00      0.88        15

    avg / total       0.92      0.89      0.89        38
```

为更好地观察效果，验证指标值的大小，下面输出样本真实结果和样本预测结果，如下图：

```
样本真实值标签为:
[ 2.  2.  1.  2.  3.  3.  1.  1.  3.  3.  3.  1.  3.  2.  3.  2.  3.  1.
  2.  3.  1.  1.  3.  1.  3.  3.  2.  2.  3.  3.  2.  2.  3.  3.  3.  3.
  3.  2.]
样本预测值标签为:
[ 2.  2.  1.  2.  2.  3.  1.  1.  3.  3.  3.  1.  3.  2.  3.  2.  2.  1.
  2.  3.  1.  1.  3.  1.  2.  2.  2.  2.  3.  3.  2.  2.  3.  3.  3.  3.
  3.  2.]
```

可以看出，在测试的 38 个样本中，决策树将其中 8 个分给了 setosa 类别、15 个分给了 versicolor 类别、15 个分给了 virginica 类别。setosa 是全部预测正确的，其 precision、recall 和 f1-score 都为 1。但是，有 4 个 virginica 类别的鸢尾花被错误预测为 versicolor 类别，因此导致这两个类别的 f1-score 有所降低。其中，support 是被预测为某类别的样本个数。总之，决策树算法将一部分的 virginica 样本预测成了 versicolor 样本，因此样本没有完全被正确预测，因此相较于 Logistic 回归，决策树的效果不是很好。

五、神经网络算法

本例中使用 MLP（multilayer perceptron）分类器，按照同样的思路，下面是 MLP 分类器的实现代码和评估结果。其代码如下：

```
def MLPClass(iris_X, iris_y):
    '''
    :param iris_X: training dataset
    :param irix_y: testing dataset
    :return: predication is the class of predicted result, proba is the predication probability,
    '''
    X_train, X_test, y_train, y_test = train_test_split(iris_X, iris_y, test_size=0.25,
random_state=33)  # split training set and testing set
    # solver is used to specify the optimization method.
    clf = MLPClassifier(solver='lbfgs', alpha=1e-5, activation='relu', hidden_layer_sizes=(5,
3), random_state=1)
    clf.fit(X_train, y_train)
    clf_y_predict = clf.predict(X_test)    # predict category
    evaluation_result = classification_report(clf_y_predict, y_test,
                                              target_names=["Iris-setosa", "Iris-versicolor", "Iris-
```

```
virginica"])    # generating evaluation report automatically
    return evaluation_result
```

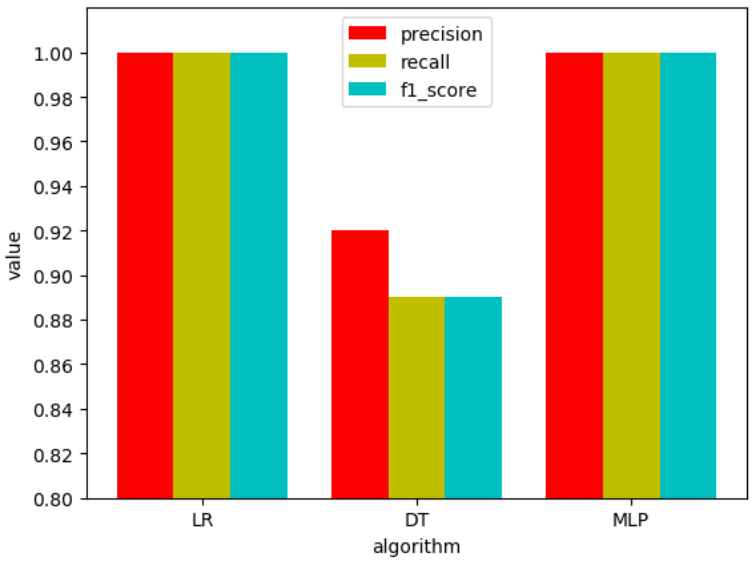
其评估结果如下：

```
下面是MLP分类器的评估结果：**************************
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      1.00      1.00        11
 Iris-virginica       1.00      1.00      1.00        19

    avg / total       1.00      1.00      1.00        38
```

从上图可以看出，MLP 分类器对鸢尾花的预测比较准确，无论是 precision、recall 还是 f1-score 都是 1。

六、算法比较

下图是 Logistic Regression、Decision Tree 和 MLP Classification 三种算法的 precision、recall 和 f1-score 的比较：



观察上图可知，Logsitic 回归和 MLP 分类器对算法的测试结果都是完全正确的，其 precision、recall 和 f1-score 都是 1。因此，就此数据集而言，决策树的性能更差一些。

七、总结

本次实验简单的尝试了 Logistic 回归、决策树和 MLP 分类器的使用，对训练集和测试集的构造在代码层面更加清晰，掌握了简单的调用规则。但是对于参数的理解还没有完全掌握，除此之外，这三种方法都是对数据集直接划分，没有进行交叉验证，可能对于实际应用来说，交叉验证的应用更加广泛。