一、 实验过程

1. 数据集及预处理

1.1 数据描述

本实验采用的数据来自于 UCI,下载地址如下：

http://archive.ics.uci.edu/ml/datasets/Adult

这是一组“人口普查收入”数据集，根据人口普查数据预测人们的收入是否超过 5 万美元/年。这组数据由 Barry Becker 从 1994 年的人口普查数据库中提取，

数据集信息描述如下表：

Adult Data Set

Download: Data Folder, Data Set Description

Abstract: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	1140500

预测任务是确定一个人的年收入是否超过 50K。

该数据集一共有 14 个属性，属性列表如下。

属性名	类型
age	continuous.
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt	continuous.
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num	continuous.
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex	Female, Male.
capital-gain	continuous.

capital-loss	continuous.
hours-per-week	continuous.
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
> 50K, <= 50K	

1.2 数据处理

- 缺失值处理

通过过对数据进行简单观察，所给的属性数据中有多处标记为“？”的数据缺失，对于这部分缺失数据的处理采用均值替代。

```
# 填充连续型变量缺失值，以均值替代
data[['age','fnlwg','capital-gain', 'capital-loss', 'hours-per-week']]=data[['age','fnlwg','capital-gain', 'capital-loss', 'hours-per-week']].replace('?',np.NaN)
data.fillna(data.mean(),inplace=True)
print(data)
```

- 去重处理

查找出数据重复行，进行删除

```
#查找出数据重复行
dIndex = data.duplicated()
print(dIndex)
#去除重复
data = data.drop_duplicates()
```

- One-Hot 编码（标签数值化）

该组数据一共包含 14 个属性，其中 age、fnlwgt、capital-gain、capital-loss、hours-per-week 5 个属性为连续形数值，其余的 9 个属性均为文字形的类别，为了数据分析，需要将这 9 个类别标签进行转换，通过 sklearn 的模块实现对离散变量的 one-hot 编码，由于 OneHotEncoder 无法直接对字符串型的类别变量编码,先用 LabelEncoder 将离散变量替换为数字，OneHotEncoder 则实现对替换为数字的离散变量进行 one-hot 编码。

首先将 workclass、education、education-num、marital-status、occupation、relationship、race、sex、native-country、> 50K，<= 50K 用 LabelEncoder()标签分配一个 0—n_classes-1 之间的编码 将各种标签分配一个可数的连续编号：标准化标签，将标签值统一转换成 range(标签值个数-1)范围内。

```
# 标签转换
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
data['workclass']=le.fit_transform(data['workclass'].values)
data['education']=le.fit_transform(data['education'].values)
data['marital-status']=le.fit_transform(data['marital-status'].values)
data['occupation']=le.fit_transform(data['occupation'].values)
data['relationship']=le.fit_transform(data['relationship'].values)
data['race']=le.fit_transform(data['race'].values)
data['sex']=le.fit_transform(data['sex'].values)
data['native-country']=le.fit_transform(data['native-country'].values)
data['50K']=le.fit_transform(data['50K'].values)
```

可以观察到相关的文字属性标签已经被替换为相应的类别数字

```
/anaconda3/bin/python3.6 /Users/miya/PycharmProjects/miya/adult.py
   age  workclass   fnlwg  education  education-num  marital-status  \
0   39          7   77516          9             13               4
1   50          6   83311          9             13               2
2   38          4  215646         11              9               0
3   53          4  234721          1              7               2
4   28          4  338409          9             13               2
5   37          4  284582         12             14               2
6   49          4  160187          6              5               3
7   52          6  209642         11              9               2
```

需要将这个分类值的特征数字化，采用 One-Hot 编码的方式进行编码

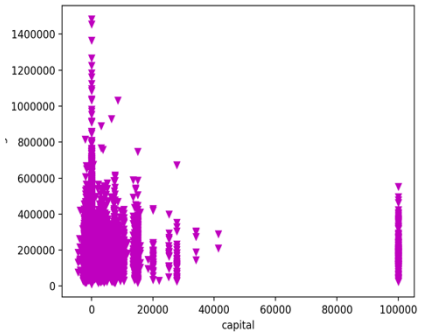
```
#OneHotEncoder 编码
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(categorical_features=np.array([1,3,5,6,7,8,9,13]))
enc.fit(X_train)
X_train = enc.transform(X_train).toarray()
X_test = enc.transform(X_test).toarray()
```

- 对数据列进行处理

数据列中有 capital-gain（正数）, capital-loss（正数）两项针对资产投资是正收益还是负收益的情况，我们用 pandas 将两项进行相减处理合并到 capital-gain 列中，所得正数即为资产投资所得的盈利，负数则为资产投资的损失值。

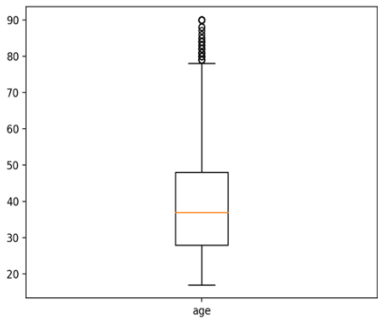
```
# 资产评估
data['capital-gain']=data['capital-gain']-data['capital-loss']
y=data['fnlwg']
plt.plot(data['capital-gain'],y,'mv',label='capital' )
plt.ylabel("fnlwg")
plt.xlabel("capital")
plt.show()
```

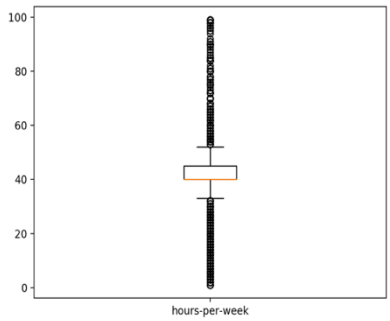
绘制出 capital-gain 减去 capital-loss 值后的散点图如下，观察到个人资产情况有正有负（注意 x 轴 0 两边的数字）

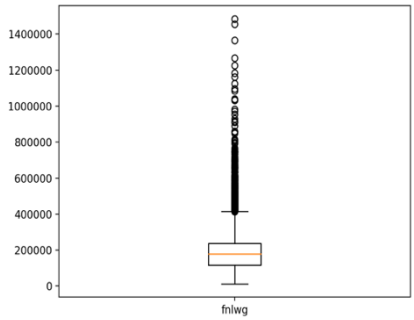


- 异常值处理

异常值处理主要是对 age、fnlwgt、capital-gain、capital-loss、hours-per-week 这五个连续型属性数据进行处理，主要依靠箱型图的方式来观察异常值







通过箱形图观察，在年龄 79 以上的点为异常点，根据常识不作为异常点处理。而 fnlwgt 在 415700 以上的判定为异常点，在 hours-per-week 中将小于 32 小时和大于 53 小时以上的点判定为异常点，capital-gain 和 capital-loss 两组资产数据由于太过于分散，数据差别较大，无法用箱形图判断。

统计出 hours-per-week 异常值的个数，>53 的有 3467 个，<31 的有 5245 个，由于数据量不小，因此我们处理以均值代替。

```
print(len(data[data['hours-per-week']>53]))
print(len(data[data['hours-per-week']<31]))
```

- 划分测试集和训练集

按照 train_test_split 随机划分测试集，设置样本比例为 40%

```
# 划分测试集和实验集
from sklearn.model_selection import train_test_split
X=np.array(data[['age','workclass','fnlwg','education','education-num','marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week','native-country']])
Y=np.array(data['50K'])
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4, random_state=0)
```

2. 模型构建

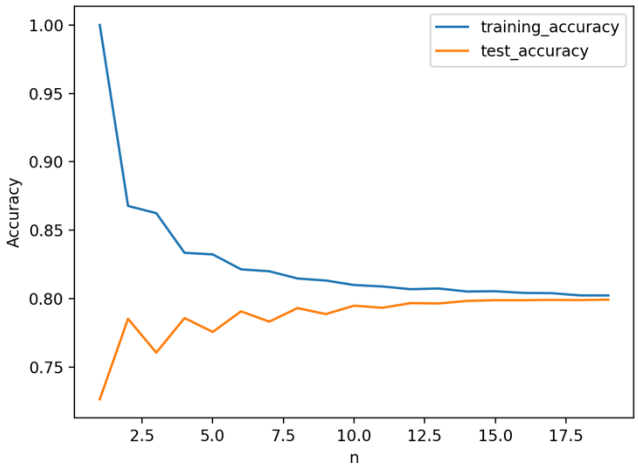
2.1KNN 算法

```
# KNN 算法
from sklearn.neighbors import KNeighborsClassifier
training_accuracy =[]
test_accuracy =[]
# try n_neighbors from 1-10
neighbors_settings=range(1,20)
for n_neighbors in neighbors_settings:
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train,Y_train)
    training_accuracy.append(knn.score(X_train,Y_train))
    test_accuracy.append(knn.score(X_test,Y_test))
```

确定 n 的个数

```
# 确定 n 的个数
plt.plot(neighbors_settings,training_accuracy,label="training_accuracy")
plt.plot(neighbors_settings,test_accuracy,label="test_accuracy")
plt.xlabel("n")
plt.ylabel("Accuracy")
plt.legend()
plt.savefig('knn_compare_model')
plt.show()
```

通过图像确定 n 的个数



通过观察，n=17 时效果最好，开始使用 knn 进行聚类

```
# 选取 n=17
knn =KNeighborsClassifier(n_neighbors=17)
knn.fit(X_train,Y_train)
print('Accuracy of KNN classifier on train set:{:.3f}'.format(knn.score(X_train,Y_train)))
print('Accuracy of KNN classifier on test set:{:.3f}'.format(knn.score(X_test,Y_test)))
from sklearn.metrics import classification_report
print(knn.predict(X_test))
target_names = ['losing', 'active']
print(classification_report(Y_test,knn.predict(X_test), target_names=target_names))
```

n=17

Accuracy of K-NN classifier on train set:0.804

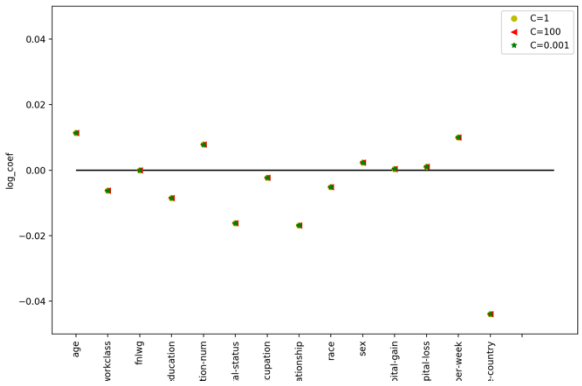
Accuracy of K-NN classifier on test set:0.799

2.2逻辑回归

```
# 逻辑回归
from sklearn.linear_model import LogisticRegression
logreg =LogisticRegression(C=1).fit(X_train,Y_train)
print('Accuracy of LR classifier on train set:{:.3f}'.format(logreg.score(X_train,Y_train)))
print('Accuracy of LR classifier on test set:{:.3f}'.format(logreg.score(X_test,Y_test)))
```

调整正则化参数 C，分别设置 C=0.001，C=1，C=100，用可视化的方式来看一下用三种不同正则化参数 C 所得模型的系数，根据图，可以看出在三种不同的参数设置下，点的位置几乎不变，说明改变 C 不影响预测结果，因此我们选择默认值 C=1。

```
# 修改参数
adult_features=[x for i ,x in enumerate(data.columns) if i !=14]
plt.plot(logreg.coef_.T,'yo',label='C=1')
plt.plot(logreg.coef_.T,'r<',label='C=100')
plt.plot(logreg.coef_.T,'g*',label='C=0.001')
plt.legend()
plt.xticks(range(data.shape[1]),adult_features,rotation=90)
plt.hlines(0,0,data.shape[1])
plt.ylim(-0.05,0.05)
plt.xlabel('Features')
plt.ylabel('log_coef')
plt.show()
```



Accuracy of LR classifier on train set:0.797

Accuracy of LR classifier on test set:0.798

2.3决策树

不减枝前

```
# 决策树
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train,Y_train)
```

```
print('Accuracy of tree classifier on train set:{:.3f}'.format(tree.score(X_train,Y_train)))
print('Accuracy of tree classifier on test set:{:.3f}'.format(tree.score(X_test,Y_test)))
```

Accuracy of tree classifier on train set:1.000

Accuracy of tree classifier on test set:0.807

训练集的准确度可以高达 100%，而测试集的准确度相对就差了很多。这表明决策树是过度拟合的，不能对新数据产生好的效果。因此，我们需要对树进行预剪枝。

剪后

设置 max_depth=7，限制树的深度以减少过拟合。这会使训练集的准确度降低，但测试集准确度提高。

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=7,random_state=0)
```

Accuracy of tree classifier on train set:0.857

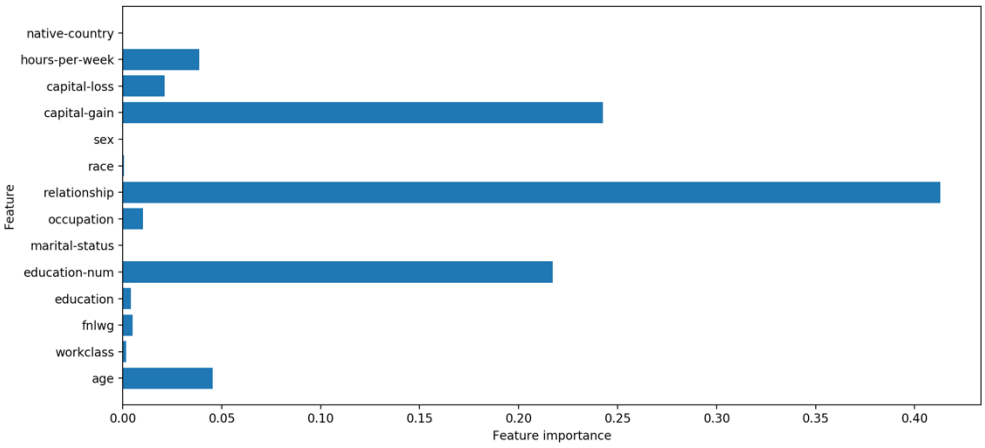
Accuracy of tree classifier on test set:0.856

特征重要程度排序

决策树中的特征重要度是用来衡量每个特征对于预测结果的重要性的。对每个特征有一个从 0 到 1 的打分，0 表示“一点也没用”，1 表示“完美预测”。各特征的重要度加和一定是为 1 的。

```
# 特征重要程度
print("特征重要度：\n{}".format(tree.feature_importances_))
diabetes_features=[x for i,x in enumerate(data.columns) if i !=14]
print(diabetes_features)
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(13,6))
    n_features=14
    plt.barh(range(n_features), model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features), diabetes_features)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1,n_features)
plot_feature_importances_diabetes(tree)
plt.show()
```

可视化特征重要度



2.4支持向量机

```
# 支持向量机
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
scaler =MinMaxScaler()
svc=SVC ()
svc.fit(X_train,Y_train)
print('Accuracy of SVM classifier on train set:{:.3f}'.format(svc.score(X_train,Y_train)))
print('Accuracy of SVM classifier on test set:{:.3f}'.format(svc.score(X_test,Y_test)))
```

Accuracy of SVM classifier on train set:0.999

Accuracy of SVM classifier on test set:0.757

这个模型过拟合比较明显，虽然在训练集中有一个完美的表现，但是在测试集中仅仅有 75%的准确度。

SVM 要求所有的特征要在相似的度量范围内变化。我们需要重新调整各特征值尺度使其基本上在同一量表上。 数据的度量标准化后效果提升很多。

```
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
scaler =MinMaxScaler()
X_train_scaled =scaler.fit_transform(X_train)
X_test_scaled =scaler.fit_transform(X_test)
```

```
svc=SVC()
svc.fit(X_train_scaled,Y_train)
print('Accuracy of SVC classifier on train set:{:.3f}'.format(svc.score(X_train_scaled,Y_train)))
print('Accuracy of SVC classifier on test set:{:.3f}'.format(svc.score(X_test_scaled,Y_test)))
```

Accuracy of SVC classifier on train set:0.828

Accuracy of SVC classifier on test set:0.827

我们还可以试着提高 C 值或者提高了 C 值后，模型效果有一定提升。

```
svc=SVC(C=10000)
```

Accuracy of SVC classifier on train set:0.852

Accuracy of SVC classifier on test set:0.851

3.模型评估与分析

3.1 混淆矩阵

```
#构建混淆矩阵
from sklearn.metrics import confusion_matrix
y_predict=model.predict(X_test)

cm=confusion_matrix(Y_test,y_predict)

col_labels=[0,1]
row_labels=[0,1]
table_vals=cm
table=plt.table(cellText=table_vals,colWidths = [0.1]*3,rowLabels=row_labels,colLabels=col_labels,loc='center')
plt.axis('off')
plt.show(table)
```

逻辑回归

	0	1
0	9575	298
1	2332	820

决策树

	0	1
0	9438	435
1	1422	1730

KNN

	0	1
0	9723	150
1	2467	685

SVM

	0	1
0	9120	753
1	1226	1926

通过混淆矩阵，可以反映不同分类器的不同分类结果，其中行代表真实类，列代表预测类，通过行与列的计算，可以得出不同的评价指标，这里我们通过计算准确率来衡量分类器的效果。

分类算法	准确率 ACCURACY
KNN	Accuracy of K-NN classifier on train set:0.804 Accuracy of K-NN classifier on test set:0.799
逻辑回归	Accuracy of LR classifier on train set:0.797 Accuracy of LR classifier on test set:0.798
决策树	Accuracy of tree classifier on train set:0.857 Accuracy of tree classifier on test set:0.856
SVM	Accuracy of SVC classifier on train set:0.852 Accuracy of SVC classifier on test set:0.851

根据准确率来分析以上四种算法，可以看出决策树的效果最好，逻辑回归有点欠拟合，决策树>SVM>KNN。

二、 实验小结

本实验通过对一组美国居民收入统计的数据集进行分析，分别利用了四种分类算法：KNN、逻辑回归、决策树和 SVM 来对数据进行预测，预测一个人的年收入是否能超过 50k，其中通过计算算法的准确率来评估算法的效果，在准确率的比较中，效果最优的是决策树，准确率在测试集上达到了 0.856，其次是 SVM 达到了 0.851，KNN 的效果一般，在聚类数最优的情况下仍旧不如 SVM 和决策树，但是 KNN 的运行效率很快。