第三次机器学习内容

> 第三次开始取的是之前两次清洗后的数据，
>
> 所以没有数据清洗步骤，将 只对Lab重点和新出现内容总结。

Lab06 - 基于概率的学习

朴素贝叶斯算法：

- 训练数据要求：数值特征
- 预测目标：分类特征

```
# 引入朴素贝叶斯算法
from sklearn.naive_bayes import GaussianNB

# 生成分类器（建模）
gnb = GaussianNB()
# 训练分类器（训练）
gnb.fit(X_train, y_train)
# 预测和评估
pred = gnb.predict(X_test)

print(classification_report(y_pred=pred, y_true=y_test))
```

Lab07 - 线性回归和逻辑回归

1、线性回归

- 训练数据要求：数值特征
- 预测目标：数值特征

```python
# 引入线性回归算法
from sklearn.linear_model import LinearRegression

# 生成线性回归模型（建模）
lin_reg = LinearRegression(normalize=True)

# 训练线性回归模型（训练）
lin_reg.fit(X=X_train, y=y_train)

# 预测和评估
pred = lin_reg.predict(X=X_test)

print("Mean squared error:", mean_squared_error(y_pred=pred,
y_true=y_test))
```

2、逻辑回归

- 训练数据要求：数值特征
- 预测目标：分类特征

```python
# 引入逻辑回归算法
from sklearn.linear_model import LinearRegression,
LogisticRegression

# 生成逻辑回归模型（建模）
log_reg = LogisticRegression(max_iter=5)

# 训练逻辑回归模型（训练）
log_reg.fit(X_train, y_train)

# 使用逻辑回归模型进行预测（预测和评估）
pred = log_reg.predict(X_test)

# 输出预测结果的均方误差
print(metrics.accuracy_score(y_test, pred))
```

Lab08 - 机器学习模型评估与性能分析

> 评估维度：1、定时性能；2、预测性能；3、ROC曲线；4、交叉验证

举例评估逻辑回归和knn

```
# 引入依赖
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
classification_report, plot_roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

# 加载、划分数据
data = datasets.load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.DataFrame(data.target, columns=['cancer_type'])

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y)
X_train, X_val, y_train, y_val = train_test_split(X_train_val,
y_train_val)

# 生成模型
log_reg = LogisticRegression(max_iter=5000)
knn = KNeighborsClassifier(n_neighbors=5)
```

1. 定时性能

```
%%time
log_reg.fit(X_train, y_train.values.ravel())
```

```
CPU times: user 2.28 s, sys: 603 ms, total: 2.88 s
Wall time: 493 ms

LogisticRegression(max_iter=5000)
```

```
%%time
knn.fit(X_train, y_train.values.ravel())
```

```
CPU times: user 10.3 ms, sys: 2.69 ms, total: 13 ms
Wall time: 2.2 ms
```

2. 预测性能

```
# 先为模型生成混淆矩阵和分类报告
confusion_matrix(y_true=y_val, y_pred=pred_log_reg)
print(classification_report(y_true=y_val, y_pred=pred_log_reg))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.97      0.99      0.98        68

    accuracy                           0.97       107
   macro avg       0.97      0.97      0.97       107
weighted avg       0.97      0.97      0.97       107
```

```
# 先为模型生成混淆矩阵和分类报告
confusion_matrix(y_true=y_val, y_pred=pred_knn)
print(classification_report(y_true=y_val, y_pred=pred_knn))
```

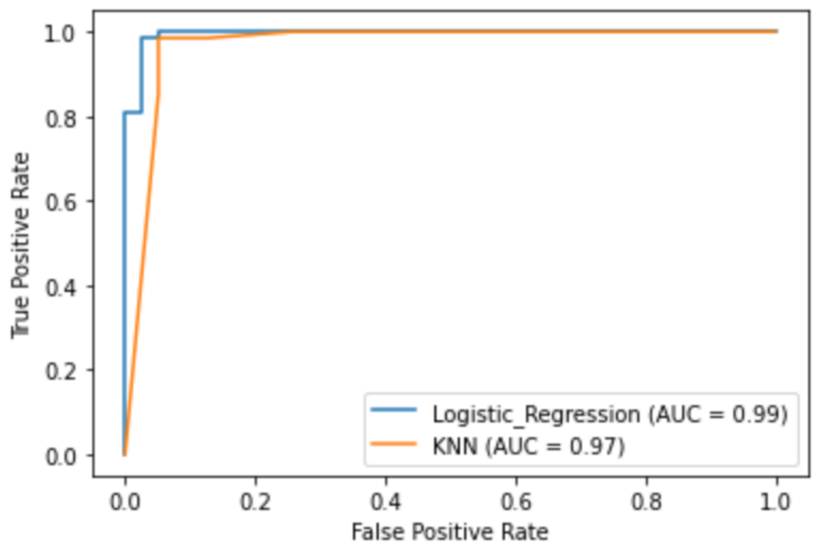
```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.97      0.99      0.98        68

    accuracy                           0.97       107
   macro avg       0.97      0.97      0.97       107
weighted avg       0.97      0.97      0.97       107
```

3. ROC曲线

```
# 绘制两个模型的ROC曲线
ax = plt.gca()
plot_log_reg = plot_roc_curve(estimator=log_reg, X=X_val,
y=y_val, ax=ax, name='Logistic_Regression')
plot_knn = plot_roc_curve(estimator=knn, X=X_val, y=y_val,
ax=ax, name='KNN')
```




```
# 输出roc曲线下更精确的面积
# Area under the curve for logistic regression model:
0.9947209653092006
pred_scores = log_reg.predict_proba(X_val)
print('Area under the curve for logistic regression model:
{}'.format(roc_auc_score(y_true=y_val,

                                      y_score=pred_scores[:, 1])))
```

```
# 输出roc曲线下更精确的面积
# Area under the curve for KNearestNeighbours model:
0.9685143288084465
pred_scores = knn.predict_proba(X_val)
print('Area under the curve for KNearestNeighbours model:
{}'.format(roc_auc_score(y_true=y_val,

                                    y_score=pred_scores[:, 1])))
```

4. 交叉验证

> 分别利用两个模型的所有训练和验证数据进行10倍交叉验证。使用宏观平均f1分数作为评分方法。

```
# F1_score: 0.95 std: 0.05
scores = cross_val_score(estimator=log_reg, X=X_train_val,
y=y_train_val.values.ravel(), cv=10, scoring='f1_macro')
print('F1_score: %0.2f std: %0.2f'%(scores.mean(),
scores.std()))
```

```
# F1_score: 0.92 std: 0.05
scores = cross_val_score(estimator=knn, X=X_train_val,
y=y_train_val.values.ravel(), cv=10, scoring='f1_macro')
print('F1_score: %0.2f std: %0.2f'%(scores.mean(),
scores.std()))
```