实战案例4：动物种类识别

作者： Robin
日期： 2018/06
数据集来源： kaggle

1. 案例描述

该项目的主要是根据动物的各项属性预测动物的类型。

2. 数据集描述

- Kaggle提供的数据集包含101条动物记录，每条记录包含18项信息。
- 数据字典
  - **animal_name**: 动物名称，字符串
  - **hair**: 是否有毛发，整型（0, 1）
  - **feathers**: 是否有皮，整型（0, 1）
  - **eggs**: 是否卵生，整型（0, 1）
  - **milk**: 是否泌乳，整型（0, 1）
  - **airborne**: 是否飞行，整型（0, 1）
  - **aquatic**: 是否水生，整型（0, 1）
  - **predator**: 是否食肉，整型（0, 1）
  - **toothed**: 是否有齿，整型（0, 1）
  - **backbone**: 是否有脊椎，整型（0, 1）
  - **breathes**: 是否呼吸，整型（0, 1）
  - **venomous**: 是否有毒，整型（0, 1）
  - **fins**: 是否有鳍，整型（0, 1）
  - **legs**: 腿的个数，整型
  - **tail**: 是否有尾，整型（0, 1）
  - **domestic**: 是否家养，整型（0, 1）
  - **catsize**: 是否和猫的大小一样，整型（0, 1）
  - **class_type**: 类别编码（1-7）

3. 任务描述

- 使用scikit-learn建立不同的机器学习模型进行动物种类识别
- 使用特征处理方法，包括编码、归一化和降维
- 使用集成学习模型

4. 主要代码解释

- 代码结构

```
lect05_proj
├── data
    ├── zoo.csv      # CSV动物样本数据文件
    ├── class.csv    # CSV动物种类对应文件
├── output
    ├── model_comparison.csv    # 模型比较结果的CSV文件(程序的输出)
    ├── pred_results.png    # 模型比较结果的png文件(程序的输出)
├── config.py     # 配置文件
├── utils.py     # 工具类文件
├── main.py      # 主程序
├── lect05_proj_readme.pdf  # 案例讲解文档
```

- **utils.py**

使用OneHotEncoder()进行特征的独热编码，sparse=False表明返回的结果不是按照稀疏矩阵返回的

```
def transform_data(train_data, test_data):
    ...
    # 独热编码处理类别特征
    encoder = OneHotEncoder(sparse=False)
    X_train_cat_feat = encoder.fit_transform(train_data[config.category_cols].values)
    X_test_cat_feat = encoder.transform(test_data[config.category_cols].values)
    ...
```

- **utils.py**

使用MinMaxScaler()进行范围归一化（默认归一化后的范围为[0, 1]）

```
def transform_data(train_data, test_data):
    ...
    # 范围归一化处理数值型特征
    scaler = MinMaxScaler()
    X_train_num_feat = scaler.fit_transform(train_data[config.num_cols].values)
    X_test_num_feat = scaler.transform(test_data[config.num_cols].values)
    ...
```

- **utils.py**

使用PCA()进行特征降维，参数n_components为整数时，表示降维后的主成分个数；参数n_components为浮点数时，表示按照“贡献率”进行主成分选取

```
def transform_data(train_data, test_data):
    ...
    # 使用特征降维
    pca = PCA(n_components=0.99)
    X_train = pca.fit_transform(X_train_raw)
    X_test = pca.transform(X_test_raw)
    ...
```

- **main.py**

使用的模型及相关参数配置。该项目中使用了8个机器学习模型，并为不同的学习模型指定了参数空间。 如：kNN，指定了3个k值用于比较对结果的影响：5, 25, 55

```
def main():
    ...
    model_name_param_dict = {'kNN': (KNeighborsClassifier(),
                                     {'n_neighbors': [5, 25, 55]}),
                             'LR': (LogisticRegression(),
                                    {'C': [0.01, 1, 100]}),
                             'SVM': (SVC(),
                                     {'C': [0.01, 1, 100]}),
                             'DT': (DecisionTreeClassifier(),
                                    {'max_depth': [50, 100, 150]}),
                             'Stacking': (sclf,
                                          {'kneighborsclassifier__n_neighbors': [5, 25, 55],
                                           'svc__C': [0.01, 1, 100],
                                           'decisiontreeclassifier__max_depth': [50, 100, 150],
                                           'meta-logisticregression__C': [0.01, 1, 100]}),
                             'AdaBoost': (AdaBoostClassifier(),
                                          {'n_estimators': [50, 100, 150, 200]}),
                             'GBDT': (GradientBoostingClassifier(),
                                      {'learning_rate': [0.01, 0.1, 1, 10, 100]}),
                             'RF': (RandomForestClassifier(),
                                    {'n_estimators': [100, 150, 200, 250]})}
```

- **utils.py**

GridSearchCV()进行网格搜索和交叉验证

```
def train_test_model(X_train, y_train, X_test, y_test, model_name, model, param_range):
    ...
    clf = GridSearchCV(estimator=model,
                       param_grid=param_range,
                       cv=5,
                       scoring='accuracy',
                       refit=True)
    ...
```

5. 案例总结

- 该项目通过预测动物种类，巩固并搭建了一个机器学习框架，包含了必要的组成模块:
    - 特征处理
    - 特征降维
    - 交叉验证
    - 集成学习

6. 课后练习

- 参考sciki-learn的官方API，实现集成学习中的hard voting和soft voting. VotingClassifier

参考资料

1. scikit-learn特征预处理
2. scikit-learn交叉验证
3. scikit-learn模型调参
4. scikit-learn集成学习
5. mlxtend