ensemble learning

October 24, 2022

```
[1]: import numpy as np
     import pandas as pd
```

```
[2]: data= pd.read_csv("horse.csv")
```

```
[3]: data.head()
```

```
[3]:   surgery    age  hospital_number  rectal_temp  pulse  respiratory_rate  \
     0      no  adult           530101         38.5   66.0              28.0
     1     yes  adult           534817         39.2   88.0              20.0
     2      no  adult           530334         38.3   40.0              24.0
     3     yes  young          5290409         39.1  164.0              84.0
     4      no  adult           530255         37.3  104.0              35.0

       temp_of_extremities peripheral_pulse mucous_membrane capillary_refill_time  \
     0                cool          reduced             NaN            more_3_sec
     1                 NaN              NaN   pale_cyanotic            less_3_sec
     2              normal           normal       pale_pink            less_3_sec
     3                cold           normal   dark_cyanotic            more_3_sec
     4                 NaN              NaN   dark_cyanotic            more_3_sec

        … packed_cell_volume total_protein abdomo_appearance abdomo_protein  \
     0  …               45.0           8.4               NaN            NaN
     1  …               50.0          85.0            cloudy            2.0
     2  …               33.0           6.7               NaN            NaN
     3  …               48.0           7.2     serosanguious            5.3
     4  …               74.0           7.4               NaN            NaN

           outcome surgical_lesion lesion_1 lesion_2  lesion_3 cp_data
     0        died              no    11300        0         0      no
     1  euthanized              no     2208        0         0      no
     2       lived              no        0        0         0     yes
     3        died             yes     2208        0         0     yes
     4        died              no     4300        0         0      no

     [5 rows x 28 columns]
```

```
[4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   surgery                299 non-null    object
 1   age                    299 non-null    object
 2   hospital_number        299 non-null    int64
 3   rectal_temp            239 non-null    float64
 4   pulse                  275 non-null    float64
 5   respiratory_rate       241 non-null    float64
 6   temp_of_extremities    243 non-null    object
 7   peripheral_pulse       230 non-null    object
 8   mucous_membrane        252 non-null    object
 9   capillary_refill_time  267 non-null    object
 10  pain                   244 non-null    object
 11  peristalsis            255 non-null    object
 12  abdominal_distention   243 non-null    object
 13  nasogastric_tube       195 non-null    object
 14  nasogastric_reflux     193 non-null    object
 15  nasogastric_reflux_ph  53 non-null     float64
 16  rectal_exam_feces      197 non-null    object
 17  abdomen                181 non-null    object
 18  packed_cell_volume     270 non-null    float64
 19  total_protein          266 non-null    float64
 20  abdomo_appearance      134 non-null    object
 21  abdomo_protein         101 non-null    float64
 22  outcome                299 non-null    object
 23  surgical_lesion        299 non-null    object
 24  lesion_1               299 non-null    int64
 25  lesion_2               299 non-null    int64
 26  lesion_3               299 non-null    int64
 27  cp_data                299 non-null    object
dtypes: float64(7), int64(4), object(17)
memory usage: 65.5+ KB
```

```
[5]: data.shape
```

```
[5]: (299, 28)
```

```
[6]: data.size
```

```
[6]: 8372
```

```
[7]: data.isna().sum()
```

[7]:

```
surgery                    0
age                        0
hospital_number            0
rectal_temp               60
pulse                     24
respiratory_rate          58
temp_of_extremities       56
peripheral_pulse          69
mucous_membrane           47
capillary_refill_time     32
pain                      55
peristalsis               44
abdominal_distention      56
nasogastric_tube         104
nasogastric_reflux       106
nasogastric_reflux_ph    246
rectal_exam_feces        102
abdomen                  118
packed_cell_volume        29
total_protein             33
abdomo_appearance        165
abdomo_protein           198
outcome                    0
surgical_lesion            0
lesion_1                   0
lesion_2                   0
lesion_3                   0
cp_data                    0
dtype: int64
```

[9]:

```python
features= data.drop("outcome" , axis=1)
```

[10]:

```python
target= data['outcome']
```

[11]:

```python
features_transformed= pd.get_dummies(features)
```

[12]:

```python
from sklearn.model_selection import train_test_split
```

[13]:

```python
from sklearn.tree import DecisionTreeClassifier
```

[14]:

```python
from sklearn.ensemble import RandomForestClassifier
```

[15]:

```python
x_train, x_test, y_train, y_test = train_test_split(features_transformed,␣
 ↪target, random_state=2)
```

[16]:

```python
print(features_transformed.shape, x_train.shape, x_test.shape)
```

```
(299, 67) (224, 67) (75, 67)
```

```
[18]: from sklearn.impute import SimpleImputer
```

```
[20]: imputer = SimpleImputer(missing_values= np.nan, strategy = 'most_frequent')
```

```
[21]: x_train= imputer.fit_transform(x_train)
      x_test = imputer.fit_transform(x_test)
```

```
[37]: my_tree = DecisionTreeClassifier(criterion = 'entropy', random_state= 2)
```

```
[38]: my_tree.fit(x_train, y_train)
```

```
[38]: DecisionTreeClassifier(criterion='entropy', random_state=2)
```

```
[39]: ypred= my_tree.predict(x_test)
```

```
[40]: from sklearn.metrics import accuracy_score, confusion_matrix,␣
       ↪classification_report
```

```
[41]: accuracy_score(y_test, ypred)
```

```
[41]: 0.56
```

```
[42]: confusion_matrix(y_test, ypred)
```

```
[42]: array([[ 8,  2, 16],
             [ 1,  2,  5],
             [ 3,  6, 32]])
```

```
[44]: print(classification_report(y_test, ypred))
```

```
              precision    recall  f1-score   support

        died       0.67      0.31      0.42        26
  euthanized       0.20      0.25      0.22         8
       lived       0.60      0.78      0.68        41

    accuracy                           0.56        75
   macro avg       0.49      0.45      0.44        75
weighted avg       0.58      0.56      0.54        75
```

### voting classifier

```
[47]: from sklearn.ensemble import VotingClassifier
      from sklearn.linear_model import LogisticRegression
      from sklearn.svm import SVC
```

[48]:

```python
lr_class= LogisticRegression()
svc_class= SVC()
rf_class= RandomForestClassifier()
```

[50]:

```python
voting_classifier= VotingClassifier(estimators =[('lr',lr_class ), ('svc',␣
 ↪svc_class),('rf',rf_class)])
```

[51]:

```python
voting_classifier.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

[51]:

```
VotingClassifier(estimators=[('lr', LogisticRegression()), ('svc', SVC()),
                             ('rf', RandomForestClassifier())])
```

[57]:

```python
for clf in (lr_class,svc_class,rf_class,voting_classifier):
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
LogisticRegression 0.5466666666666666
SVC 0.5466666666666666
RandomForestClassifier 0.68
VotingClassifier 0.5466666666666666
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

0.1 Bagging and Boosting

[58]:
```
from sklearn.ensemble import BaggingClassifier
```

[60]:
```
bag_clf= BaggingClassifier(DecisionTreeClassifier(), n_estimators= 100)
```

[61]:
```
bag_clf.fit(x_train,y_train)
```

[61]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

[63]:
```
y_pred= bag_clf.predict(x_test)
```

[65]:
```
accuracy_score(y_test, y_pred)
```

[65]:
```
0.6133333333333333
```

[66]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        died       0.61      0.65      0.63        26
  euthanized       0.00      0.00      0.00         8
       lived       0.67      0.71      0.69        41

    accuracy                           0.61        75
   macro avg       0.43      0.45      0.44        75
weighted avg       0.58      0.61      0.60        75
```

[67]:
```
confusion_matrix(y_test, y_pred)
```

[67]:
```
array([[17,  0,  9],
       [ 3,  0,  5],
       [ 8,  4, 29]])
```

[68]:
```
from sklearn.ensemble import AdaBoostClassifier
```

[69]:
```
ada_boost=AdaBoostClassifier(DecisionTreeClassifier(), n_estimators=100)
```

[70]:
```
ada_boost.fit(x_train,y_train)
```

[70]:
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

[71]:
```
y_pred= ada_boost.predict(x_test)
```

[72]:
```
accuracy_score(y_test, y_pred)
```

[72]:
```
0.5466666666666666
```

[73]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        died       0.52      0.58      0.55        26
  euthanized       0.12      0.12      0.12         8
       lived       0.66      0.61      0.63        41

    accuracy                           0.55        75
   macro avg       0.43      0.44      0.43        75
weighted avg       0.55      0.55      0.55        75
```

[74]:
```
confusion_matrix(y_test, y_pred)
```

[74]:
```
array([[15,  2,  9],
       [ 3,  1,  4],
       [11,  5, 25]])
```

[75]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

[77]:
```
grad_boost= GradientBoostingClassifier()
```

[78]:
```
grad_boost.fit(x_train,y_train)
```

[78]:
```
GradientBoostingClassifier()
```

[79]:
```
y_pred=grad_boost.predict(x_test)
```

[80]:
```
accuracy_score(y_test, y_pred)
```

[80]:
```
0.56
```

[81]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        died       0.53      0.69      0.60        26
```

```
  euthanized       0.00      0.00      0.00         8
       lived       0.69      0.59      0.63        41

    accuracy                           0.56        75
   macro avg       0.41      0.43      0.41        75
weighted avg       0.56      0.56      0.55        75
```

```
[82]: confusion_matrix(y_test, y_pred)
```

```
[82]: array([[18,  1,  7],
             [ 4,  0,  4],
             [12,  5, 24]])
```

```
[87]: import xgboost
```

```
[88]: xgb_class= xgboost.XGBClassifier()
```

```
[89]: xgb_class.fit(x_train,y_train)
```

```
[89]: XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
                    colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                    importance_type='gain', interaction_constraints=None,
                    learning_rate=0.300000012, max_delta_step=0, max_depth=6,
                    min_child_weight=1, missing=nan, monotone_constraints=None,
                    n_estimators=100, n_jobs=0, num_parallel_tree=1,
                    objective='multi:softprob', random_state=0, reg_alpha=0,
                    reg_lambda=1, scale_pos_weight=None, subsample=1,
                    tree_method=None, validate_parameters=False, verbosity=None)
```

```
[90]: y_pred=xgb_class.predict(x_test)
```

```
[91]: accuracy_score(y_test, y_pred)
```

```
[91]: 0.6266666666666667
```

```
[92]: confusion_matrix(y_test, y_pred)
```

```
[92]: array([[17,  1,  8],
             [ 3,  1,  4],
             [ 9,  3, 29]])
```

```
[93]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        died       0.59      0.65      0.62        26
  euthanized       0.20      0.12      0.15         8
```

```
       lived       0.71      0.71      0.71        41

    accuracy                           0.63        75
   macro avg       0.50      0.50      0.49        75
weighted avg       0.61      0.63      0.62        75
```

[ ]:

```
```