DecisionTree (1)

August 19, 2022

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

data = pd.read_csv('horse.csv')
data.head()
```

[1]:

	surgery	age	hospital_number	rectal_temp	pulse	respiratory_rate	temp_of_extremities	peripheral_pulse	mucous_membrane	capillary_refill_time	…	packed_cell_volume	total_protein	abdomo_appearance	abdomo_protein	outcome	surgical_lesion	lesion_1	lesion_2	lesion_3	cp_data
0	no	adult	530101	38.5	66.0	28.0	cool	reduced	NaN	more_3_sec	…	45.0	8.4	NaN	NaN	died	no	11300	0	0	no
1	yes	adult	534817	39.2	88.0	20.0	NaN	NaN	pale_cyanotic	less_3_sec	…	50.0	85.0	cloudy	2.0	euthanized	no	2208	0	0	no
2	no	adult	530334	38.3	40.0	24.0	normal	normal	pale_pink	less_3_sec	…	33.0	6.7	NaN	NaN	lived	no	0	0	0	yes
3	yes	young	5290409	39.1	164.0	84.0	cold	normal	dark_cyanotic	more_3_sec	…	48.0	7.2	serosanguious	5.3	died	yes	2208	0	0	yes
4	no	adult	530255	37.3	104.0	35.0	NaN	NaN	dark_cyanotic	more_3_sec	…	74.0	7.4	NaN	NaN	died	no	4300	0	0	no

[5 rows x 28 columns]

[2]:
```
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
```

[3]:
```
data.isna().sum()
```

[3]:
```
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
```

[4]:
```
#Target Class
data.outcome.value_counts()
```

[4]:
```
lived         178
died           77
euthanized     44
Name: outcome, dtype: int64
```

[5]:
```
features = data.drop(['outcome'], axis = 1)
target = data[['outcome']]
```

[6]:
```
features.shape,target.shape
```

[6]:
```
((299, 27), (299, 1))
```

[7]:
```
features.dtypes
```

[7]:
```
surgery                  object
age                      object
hospital_number           int64
rectal_temp             float64
pulse                   float64
respiratory_rate        float64
temp_of_extremities      object
peripheral_pulse         object
mucous_membrane          object
```

```
capillary_refill_time     object
pain                      object
peristalsis               object
abdominal_distention      object
nasogastric_tube          object
nasogastric_reflux        object
nasogastric_reflux_ph    float64
rectal_exam_feces         object
abdomen                   object
packed_cell_volume       float64
total_protein            float64
abdomo_appearance         object
abdomo_protein           float64
surgical_lesion           object
lesion_1                   int64
lesion_2                   int64
lesion_3                   int64
cp_data                   object
dtype: object
```

[8]:
```
features_transformed = pd.get_dummies(features)
```

[9]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[10]:
```
X_train , X_test, y_train, y_test = train_test_split(features_transformed,␣
↪target, random_state = 10)
```

[11]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(224, 67)
(75, 67)
(224, 1)
(75, 1)
```

[12]:
```
from sklearn.impute import SimpleImputer
```

[13]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
```

[14]:
```
X_train = imputer.fit_transform(X_train)
X_test = imputer.fit_transform(X_test)
```

[15]:
```
my_DT_model = DecisionTreeClassifier(criterion='entropy', random_state=2,␣
↪max_depth = 1)
```

[16]:
```
my_DT_model.fit(X_train,y_train)
```

[16]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=1, random_state=2)
```

0.0.1 Using GridSearchCV to find best params

[17]:
```
from sklearn.model_selection import GridSearchCV
```

[18]:
```
params = {'criterion':['gini', 'entropy'], 'max_depth':[1,2,3,10], 'splitter' :
↪['best', 'random']}
```

[19]:
```
grid_search = GridSearchCV(my_DT_model, params, cv = 3, n_jobs = -1)
```

[20]:
```
grid_search.fit(X_train, y_train)
```

[20]:
```
GridSearchCV(cv=3,
             estimator=DecisionTreeClassifier(criterion='entropy', max_depth=1,
                                              random_state=2),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 10],
                         'splitter': ['best', 'random']})
```

[21]:
```
grid_search.best_params_
```

[21]:
```
{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}
```

[22]:
```
my_DT_model = DecisionTreeClassifier(criterion='gini', random_state=2,␣
↪max_depth = 3, splitter = 'best')
```

[23]:
```
my_DT_model.fit(X_train, y_train)
```

[23]:
```
DecisionTreeClassifier(max_depth=3, random_state=2)
```

[24]:
```
my_preds = my_DT_model.predict(X_test)
```

[25]:
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[26]:
```
accuracy_score(y_test, my_preds)
```

[26]:
```
0.6666666666666666
```

[27]:
```
print(confusion_matrix(y_test, my_preds, ))
```

```
[[ 3  0 12]
 [ 1  1  9]
 [ 3  0 46]]
```

[28]:
```
print(classification_report(y_test, my_preds))
```

```
              precision    recall  f1-score   support

        died       0.43      0.20      0.27        15
  euthanized       1.00      0.09      0.17        11
       lived       0.69      0.94      0.79        49

    accuracy                           0.67        75
   macro avg       0.71      0.41      0.41        75
weighted avg       0.68      0.67      0.60        75
```

1 Voting Classifiers

[29]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

[30]:
```
rf_clf = RandomForestClassifier()
log_clf = LogisticRegression()
svm_clf = SVC()
```

[31]:
```
voting_clf = VotingClassifier(estimators=[('lr', log_clf), ('rf', rf_clf),␣
↪('svc', svm_clf)])
```

[32]:
```
voting_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765:
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)
```

[32]:
```
VotingClassifier(estimators=[('lr', LogisticRegression()),
                             ('rf', RandomForestClassifier()), ('svc', SVC())])
```

[33]:
```
from sklearn.metrics import accuracy_score
```

[34]:
```
for clf in (log_clf, rf_clf, svm_clf, voting_clf):
    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/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765:
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)
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().


LogisticRegression 0.6266666666666667
RandomForestClassifier 0.7333333333333333
SVC 0.6533333333333333

/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765:
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)
```

```
VotingClassifier 0.64
```

[35]:
```
from sklearn.ensemble import BaggingClassifier
```

[36]:
```
bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=100)
```

[37]:
```
bag_clf.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

[37]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

[38]:
```
y_pred = bag_clf.predict(X_test)
accuracy_score(y_pred, y_test)
```

[38]:
```
0.7333333333333333
```

[39]:
```
my_rf_classifier = RandomForestClassifier()
```

[40]:
```
my_rf_classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:1:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  """Entry point for launching an IPython kernel.
```

[40]:
```
RandomForestClassifier()
```

[41]:
```
my_predictions = my_rf_classifier.predict(X_test)
```

[42]:
```
print(accuracy_score(y_test, my_predictions))
```

```
0.7466666666666667
```

[43]:
```
print(confusion_matrix(y_test, my_predictions))
```

```
[[ 8  0  7]
 [ 1  2  8]
 [ 3  0 46]]
```

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

```
              precision    recall  f1-score   support

        died       0.67      0.53      0.59        15
  euthanized       1.00      0.18      0.31        11
       lived       0.75      0.94      0.84        49

    accuracy                           0.75        75
   macro avg       0.81      0.55      0.58        75
weighted avg       0.77      0.75      0.71        75
```

[45]:
```
from sklearn.ensemble import VotingClassifier, BaggingClassifier
from sklearn.linear_model import LogisticRegression
```

[46]:
```
my_logreg_clf = LogisticRegression()
```

[47]:
```
my_vt_clf = VotingClassifier(estimators=[('lr', my_logreg_clf), ('rf',␣
↪my_rf_classifier)],
                             voting = 'hard')
```

[48]:
```
my_vt_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765:
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)
```

[48]:
```
VotingClassifier(estimators=[('lr', LogisticRegression()),
                             ('rf', RandomForestClassifier())])
```

[49]:
```
my_bagging_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=100,
                                   max_samples=100, bootstrap=True)

my_bagging_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

[49]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), max_samples=100,
                  n_estimators=100)
```

[50]:
```
from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier(DecisionTreeClassifier(), n_estimators= 100)
```

[51]:
```
ada_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

[51]:
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

[52]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

[53]:
```
#learning_rate = 0.3, max_depth=5, n_estimators=1100, n_iter_no_change=10
gbc_clf = GradientBoostingClassifier()
```

[54]:
```
gbc_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

[54]:
```
GradientBoostingClassifier()
```

[55]:
```
gbc_clf.n_estimators_
```

[55]: 100

[56]:
```
import xgboost, time
```

[57]:
```
xgb_clf = xgboost.XGBClassifier()
```

[58]:
```
start = time.time()
xgb_clf.fit(X_train, y_train)
end = time.time()

time_elapsed = end - start
print(time_elapsed)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

```
88.5109612941742
```

[59]:
```
y_pred = xgb_clf.predict(X_test)
```

[60]:
```
accuracy_score(y_pred, y_test)
```

[60]: 0.7466666666666667

[ ]:
```
params = {'n_estimators':[100, 200, 400, 800], 'max_depth':[1,2,3,6,10],␣
↪'learning_rate' :[0.1, 0.2, 0.3, 0.5], 'min_child_weight' : [1, 2, 3, 4, 5],␣
↪'subsample' : [0.5, 0.6, 0.7, 0.8, 1.0]}
grid_search = GridSearchCV(xgb_clf, params, cv = 3, n_jobs = -1)
grid_search.fit(X_train, y_train)
```

[ ]:
```
grid_search.best_params_
```

[ ]:
```
xgb_clf = xgboost.XGBClassifier(learning_rate = 0.1, max_depth = 3,␣
↪min_child_weight = 5, n_estimators = 200, subsample = 0.6)
```

[ ]:
```
xgb_clf.fit(X_train, y_train)
```

[ ]:
```
y_pred = xgb_clf.predict(X_test)
accuracy_score(y_pred, y_test)
```

[ ]: