penguins_post

September 29, 2024

1 Classification

The Palmer Penguins dataset is a common resource for data exploration and demonstration of data analysis techniques. It was brought into the limelight by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, which is a member of the Long Term Ecological Research Network.

The dataset includes data for 344 penguins from three different species found on three islands in the Palmer Archipelago, Antarctica. The measured attributes in the dataset include:

1. **Species**: The species of the penguin, which can be Adelie, Gentoo, or Chinstrap.
2. **Island**: The island in the Palmer Archipelago, Antarctica, where the penguin observation was made. The options are Torgersen, Biscoe, or Dream.
3. **Culmen Length (mm)**: The length of the penguin's culmen (bill).
4. **Culmen Depth (mm)**: The depth of the penguin's culmen (bill).
5. **Flipper Length (mm)**: The length of the penguin's flipper.
6. **Body Mass (g)**: The body mass of the penguin.
7. **Sex**: The sex of the penguin.

The Palmer Penguins dataset is excellent for practicing data cleaning, exploration, and visualization.

You can find more information about the dataset, including a more detailed explanation of the variables, in this repository: allisonhorst/palmerpenguins.

For more in-depth studies or referencing, you might also consider checking out the publications from Palmer Station LTER: pal.lternet.edu/bibliography.

[47]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
```

```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import ConfusionMatrixDisplay
```

[48]:
```
# read penquins dataset from github
penguins = pd.read_csv('https://raw.githubusercontent.com/allisonhorst/
↪palmerpenguins/master/inst/extdata/penguins.csv')
penguins.describe()
```

[48]:
```
       bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g  \
count      342.000000     342.000000         342.000000   342.000000
mean        43.921930      17.151170         200.915205  4201.754386
std          5.459584       1.974793          14.061714   801.954536
min         32.100000      13.100000         172.000000  2700.000000
25%         39.225000      15.600000         190.000000  3550.000000
50%         44.450000      17.300000         197.000000  4050.000000
75%         48.500000      18.700000         213.000000  4750.000000
max         59.600000      21.500000         231.000000  6300.000000

              year
count   344.000000
mean   2008.029070
std       0.818356
min    2007.000000
25%    2007.000000
50%    2008.000000
75%    2009.000000
max    2009.000000
```

[49]:
```
# drop the year column, it is not useful for our analysis,
# and it has no adequate explanation in the dataset documentation
penguins = penguins.drop("year", axis=1)
penguins.head()
```

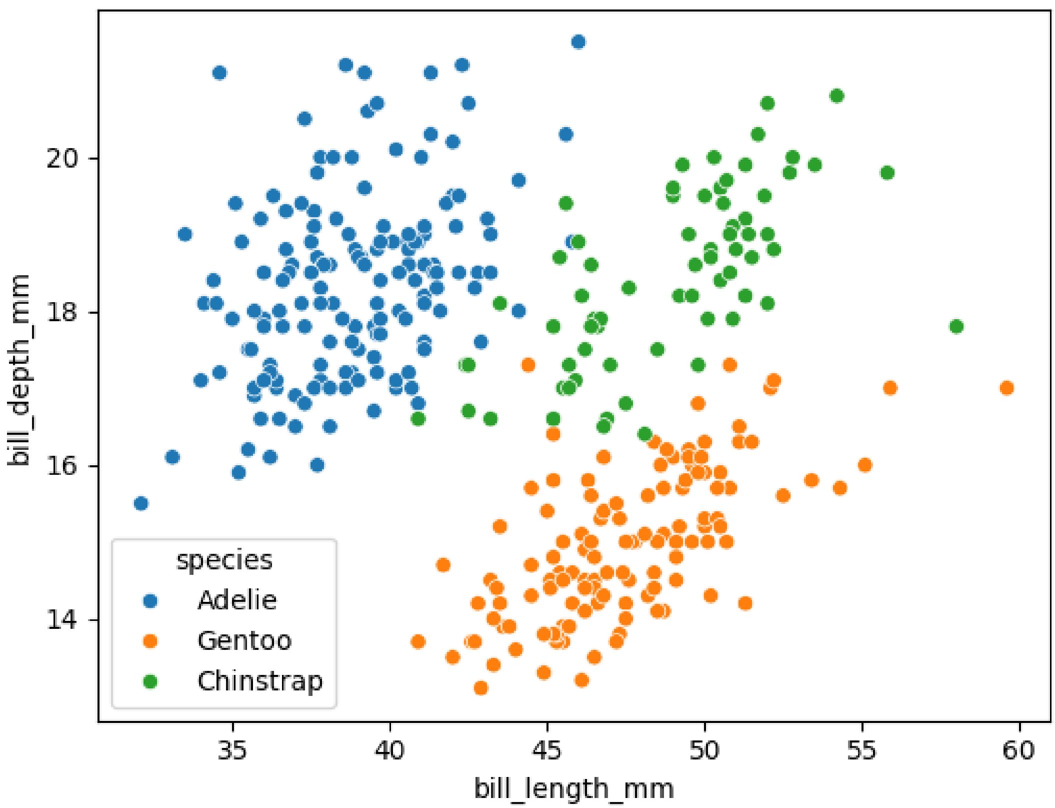
[49]:
```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    male
1       3800.0  female
2       3250.0  female
```

```
3          NaN     NaN
4       3450.0  female
```

[50]:
```
# Create a scatterplot of bill length vs bill depth using seaborn, hue by␣
 ↪species.
# Add a title.
sns.scatterplot(x=penguins["bill_length_mm"],
                y=penguins["bill_depth_mm"],
                hue=penguins["species"])
```

[50]:
```
<Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



[51]:
```
numeric_features = ['bill_length_mm', 'bill_depth_mm',␣
 ↪'flipper_length_mm',          'body_mass_g']
categorical_features = ['island', 'sex']
```

[52]:
```
penguins_num = penguins[numeric_features]
penguins_cat = penguins[categorical_features]

# create a pipeline to impute missing values with the mean and scale numeric␣
 ↪features
```

```
pipeline_num = Pipeline([
                      ("impute", SimpleImputer(strategy='median')),
                      ("standardize", StandardScaler())
                      ])
# pipeline_num


# create a pipeline to impute missing values with the most frequent value and␣
 ↪one-hot encode categorical features

pipeline_cat = Pipeline([
                      ("impute", SimpleImputer(strategy="most_frequent")),
                      ("encode", OneHotEncoder())
                      ])
# pipeline_cat


# create a column transformer to apply the numeric and categorical pipelines to␣
 ↪the correct features
# use remainder='passthrough' to keep the remaining features in the dataframe
preprossessing = ColumnTransformer([
    ("numerical_pipeline", pipeline_num, numeric_features),
    ("categorical_pipeline", pipeline_cat, categorical_features)],
    remainder='passthrough'
)

# fit_transform the preprocessor on the penguins dataset
# convert the result to a dataframe
# use the preprocessor's get_feature_names_out() method to get the column names
penguins_prepared = preprossessing.fit_transform(penguins)
penguins_prep_df = pd.DataFrame(penguins_prepared, columns=preprossessing.
 ↪get_feature_names_out())

# display the first 5 rows of the preprocessed dataframe
print(penguins_prep_df.head())
```

```
  numerical_pipeline__bill_length_mm numerical_pipeline__bill_depth_mm  \
0                          -0.887622                          0.787289
1                          -0.814037                          0.126114
2                          -0.666866                          0.431272
3                           0.096581                          0.075255
4                          -1.329133                          1.092447

  numerical_pipeline__flipper_length_mm numerical_pipeline__body_mass_g  \
0                             -1.420541                       -0.564625
1                             -1.063485                        -0.50201
```

```
2                         -0.420786                  -1.190773
3                         -0.277964                  -0.188936
4                         -0.563608                  -0.940314

  categorical_pipeline__island_Biscoe categorical_pipeline__island_Dream  \
0                                 0.0                                0.0
1                                 0.0                                0.0
2                                 0.0                                0.0
3                                 0.0                                0.0
4                                 0.0                                0.0

  categorical_pipeline__island_Torgersen categorical_pipeline__sex_female  \
0                                    1.0                              0.0
1                                    1.0                              1.0
2                                    1.0                              1.0
3                                    1.0                              0.0
4                                    1.0                              1.0

  categorical_pipeline__sex_male remainder__species
0                            1.0            Adelie
1                            0.0            Adelie
2                            0.0            Adelie
3                            1.0            Adelie
4                            0.0            Adelie
```

[53]:
```
# separate the features from the target
# call the features X and the target y

X = penguins_prep_df.drop("remainder__species", axis=1)
y = penguins_prep_df["remainder__species"].copy()
```

[54]:
```
# setup binary classification for Adelie vs. rest of species
# use the Adelie species as the positive class
# create a new target called y_adelie

y_adelie = (y == "Adelie")
```

[55]:
```
# build an SGDClassifier model using X and y
# use random_state=42 for reproducibility
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X, y_adelie)
```

[55]:
```
SGDClassifier(random_state=42)
```

[56]:
```
# compute the accuracy using cross_val_score with cv=10
accuracy = cross_val_score(sgd_clf,
                           X,
                           y_adelie,
```

```
                cv=10,
                scoring="accuracy")
accuracy
```

[56]:
```
array([0.97142857, 1.        , 0.97142857, 1.        , 1.        ,
       1.        , 1.        , 0.97058824, 1.        , 0.97058824])
```

[57]:
```
# compute the mean accuracy
mean = sum(accuracy) / len(accuracy)
mean
```

[57]:
```
np.float64(0.9884033613445379)
```

[58]:
```
# predict the target using cross_val_predict with cv=10
# call the result y_train_pred
y_train_pred = cross_val_predict(sgd_clf,
                                 X,
                                 y_adelie,
                                 cv=10,)
```

[59]:
```
# compute the confusion matrix
confusion = confusion_matrix(y_adelie,
                             y_train_pred,
                             labels=[True, False])
print(confusion)
```

```
[[149   3]
 [  1 191]]
```

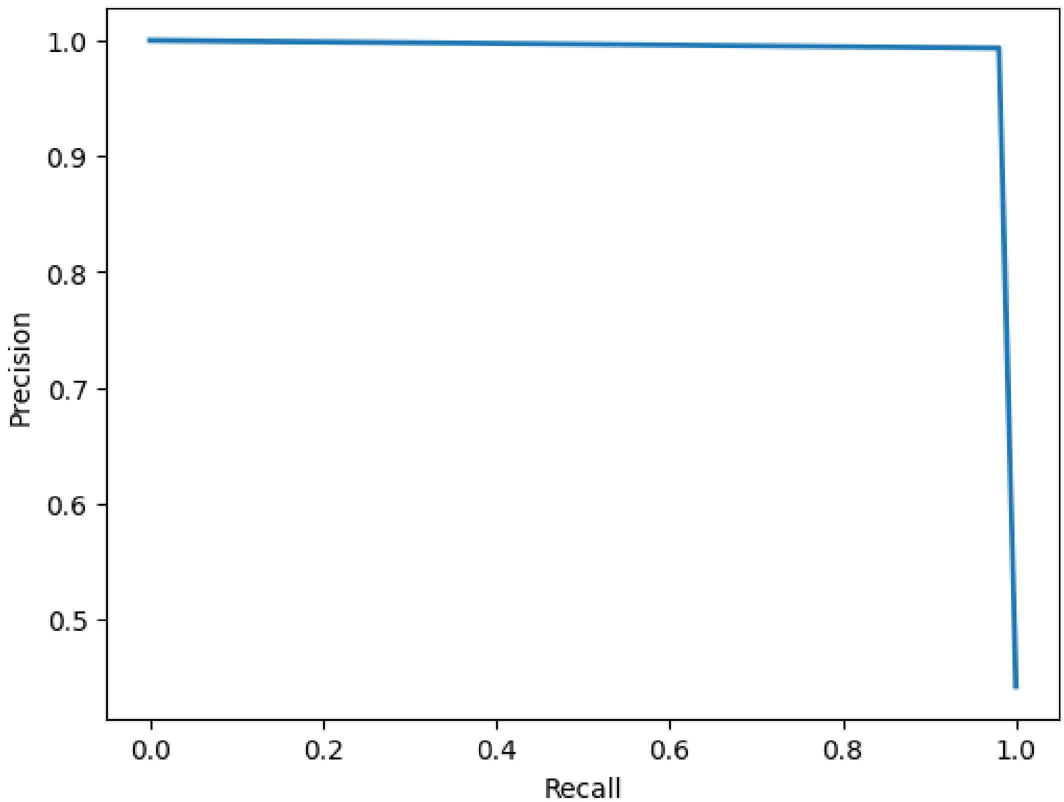
[60]:
```
# compute the precision score using precision_score()
precision_score(y_adelie, y_train_pred)
```

[60]:
```
np.float64(0.9933333333333333)
```

[61]:
```
# compute the recall score using recall_score()
recall_score(y_adelie, y_train_pred)
```

[61]:
```
np.float64(0.9802631578947368)
```

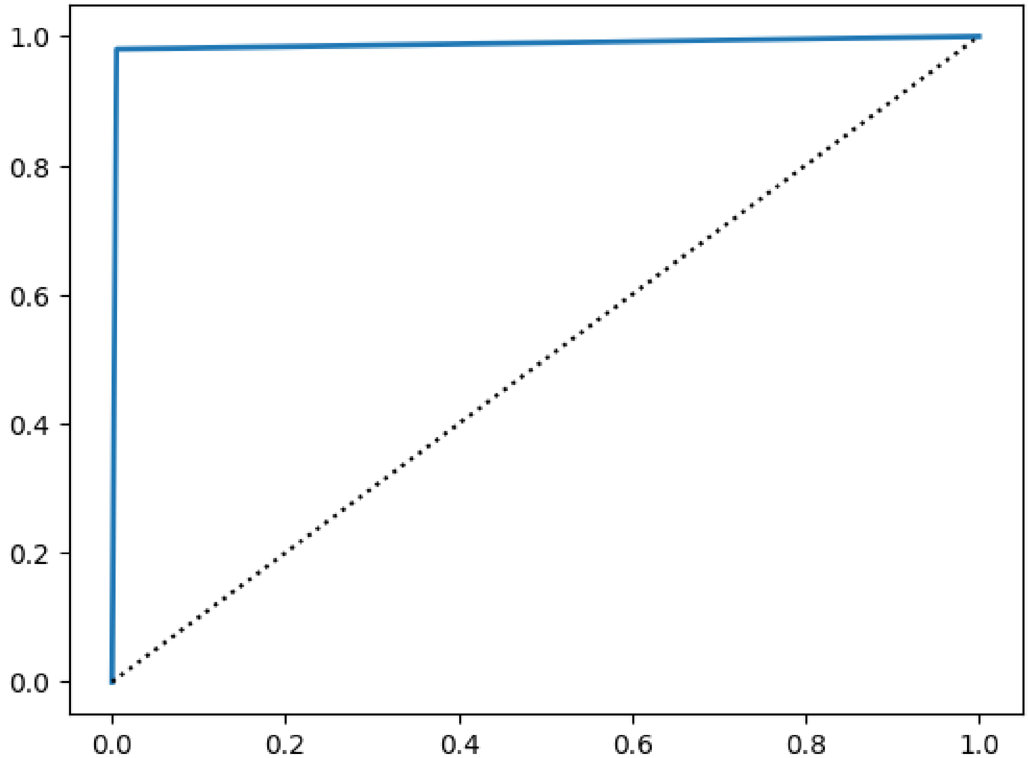
[62]:
```
# draw the precision-recall curve
# call the result precisions, recalls, thresholds
precisions, recalls, thresholds = precision_recall_curve(y_adelie, y_train_pred)
plt.plot(recalls, precisions, linewidth=2, label="Precision/Recall curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.show()
```



[63]:
```
# call the result fpr, tpr, thresholds
# plot the roc curve
fpr, tpr, thresholds = roc_curve(y_adelie, y_train_pred)

plt.plot(fpr, tpr, linewidth=2, label="ROC curve")
plt.plot([0, 1], [0, 1], 'k:', label="Random classifier's ROC curve")
```

[63]:
```
[<matplotlib.lines.Line2D at 0x7dbe7a0d9df0>]
```



[64]:
```
# now let's do multiclass classification
# build an SGDClassifier model using X and y
# use random_state=42 for reproducibility
multi_clf = SGDClassifier(random_state=42)
multi_clf.fit(X, y)
```

[64]:
```
SGDClassifier(random_state=42)
```

[65]:
```
# show the mean accuracy using cross_val_score with cv=10
cross_val_score(multi_clf,
                X,
                y,
                cv=10,
                scoring="accuracy")
```

[65]:
```
array([1.        , 1.        , 0.97142857, 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 0.94117647])
```

[66]:
```
# predict the target using cross_val_predict with cv=10
# call the result y_train_pred
# show the confusion matrix
y_train_pred = cross_val_predict(multi_clf,
```

```
                                     X,
                                     y,
                                     cv=10)

confusion = confusion_matrix(y,
                             y_train_pred,
                             labels=["Adelie", "Gentoo", "Chinstrap"])
confusion
```

[66]:
```
array([[150,   0,   2],
       [  1, 123,   0],
       [  0,   0,  68]])
```

[67]:
```
# use ConfusionMatrixDisplay to display the confusion matrix
ConfusionMatrixDisplay.from_predictions(y, y_train_pred)
plt.show()
```