# penguins_post

February 10, 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.

[ ]:
```python
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
```

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

[ ]:
```
  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  year
0       3750.0    male  2007
1       3800.0  female  2007
2       3250.0  female  2007
3          NaN     NaN  2007
4       3450.0  female  2007
```

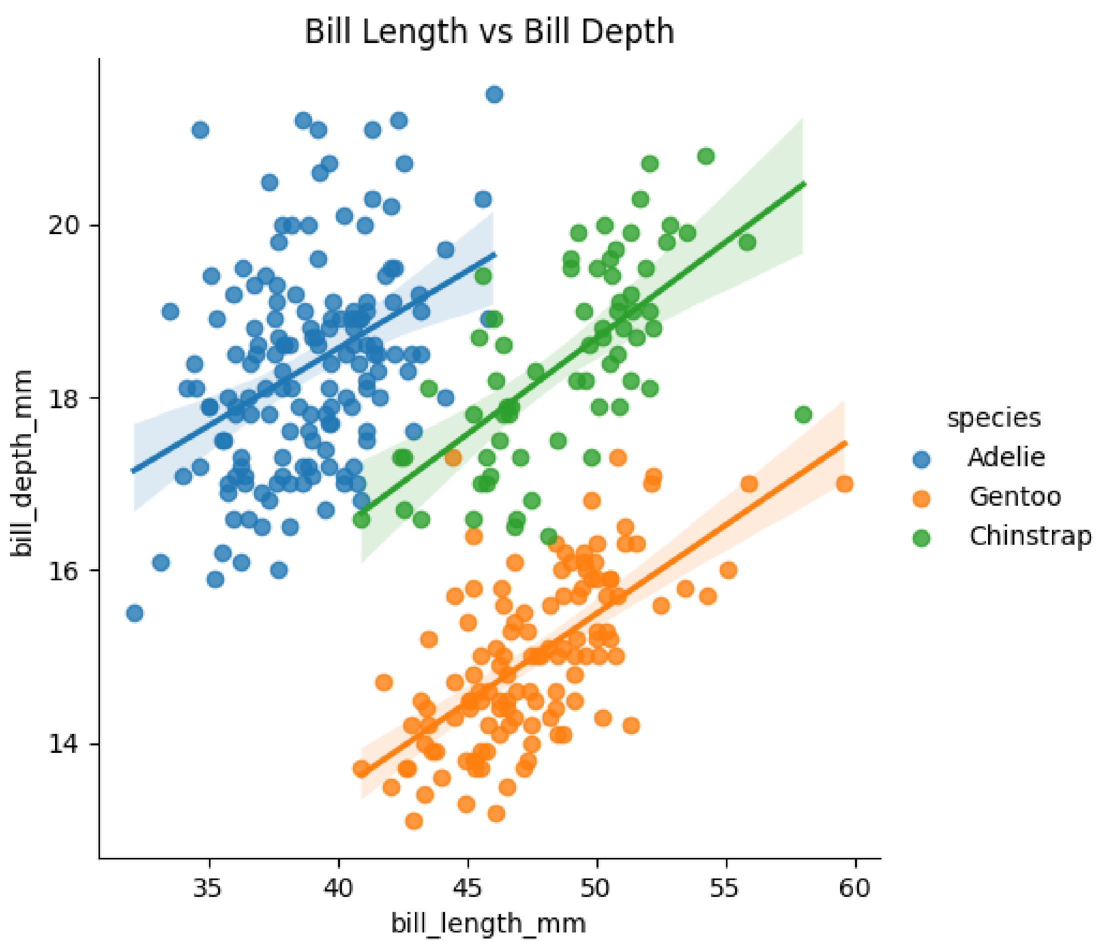
[ ]:
```
# drop the year column, it is not useful for our analysis,
# and it has no adequate explanation in the dataset documentation

penguins=penguins.drop('year',axis=1)
penguins
```

[ ]:
```
       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
..         …          …             …              …                  …
339  Chinstrap      Dream            55.8           19.8              207.0
340  Chinstrap      Dream            43.5           18.1              202.0
341  Chinstrap      Dream            49.6           18.2              193.0
342  Chinstrap      Dream            50.8           19.0              210.0
343  Chinstrap      Dream            50.2           18.7              198.0

     body_mass_g     sex
0         3750.0    male
1         3800.0  female
```

```
2          3250.0  female
3             NaN     NaN
4          3450.0  female
..            …       …
339        4000.0    male
340        3400.0  female
341        3775.0    male
342        4100.0    male
343        3775.0  female

[344 rows x 7 columns]
```

[ ]:
```
# Create a scatterplot of bill length vs bill depth using seaborn, hue by␣
 ↪species.
# Add a title.
#penguins.plot(kind='scatter',x='bill_length_mm',y='bill_depth_mm',␣
 ↪hue='species',alpha=0.4)

sns.lmplot(data=penguins,x='bill_length_mm', y='bill_depth_mm',hue='species').
 ↪set(title='Bill Length vs Bill Depth')
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x28eb06db020>
```



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

```
[ ]: from sklearn.compose import ColumnTransformer
     from sklearn.impute import SimpleImputer
     from sklearn.preprocessing import StandardScaler
     from sklearn.preprocessing import OneHotEncoder
     from sklearn.pipeline import make_pipeline


     # create a pipeline to impute missing values with the mean and scale numeric␣
      ↪features
     num_pipeline = make_pipeline(
         SimpleImputer(strategy='mean'),
         StandardScaler()
     )
```

```
# create a pipeline to impute missing values with the most frequent value and␣
 ↪one-hot encode categorical features
cat_pipeline = make_pipeline(
    SimpleImputer(strategy='most_frequent'),
    OneHotEncoder(handle_unknown='ignore')
)

# 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
preprocessing = ColumnTransformer([
    ("num",num_pipeline,numeric_features),
    ("cat",cat_pipeline,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 = preprocessing.fit_transform(penguins) #transforming␣
 ↪dataframe
print(penguins_prepared.shape) #dont need this
print(preprocessing.get_feature_names_out()) #column names

penguins_prepared_df = pd.DataFrame(penguins_prepared,columns=preprocessing.
 ↪get_feature_names_out())

penguins_prepared_df.head()

# display the first 5 rows of the preprocessed dataframe
```

```
(344, 10)
['num__bill_length_mm' 'num__bill_depth_mm' 'num__flipper_length_mm'
 'num__body_mass_g' 'cat__island_Biscoe' 'cat__island_Dream'
 'cat__island_Torgersen' 'cat__sex_female' 'cat__sex_male'
 'remainder__species']
```

[ ]:
```
  num__bill_length_mm num__bill_depth_mm num__flipper_length_mm  \
0           -0.887081           0.787743              -1.422488
1           -0.813494           0.126556              -1.065352
2            -0.66632           0.431719              -0.422507
3                -0.0                0.0                    0.0
4           -1.328605           1.092905              -0.565361

  num__body_mass_g cat__island_Biscoe cat__island_Dream cat__island_Torgersen  \
0        -0.565789                0.0               0.0                   1.0
```

```
1       -0.503168               0.0               0.0                   1.0
2       -1.192003               0.0               0.0                   1.0
3             0.0               0.0               0.0                   1.0
4       -0.941517               0.0               0.0                   1.0

  cat__sex_female cat__sex_male remainder__species
0             0.0           1.0             Adelie
1             1.0           0.0             Adelie
2             1.0           0.0             Adelie
3             0.0           1.0             Adelie
4             1.0           0.0             Adelie
```

[ ]:
```
# separate the features from the target
# call the features X and the target y
#X = penguins_prepared_df.drop(columns=[penguins_prepared_df.columns[0]])
#i think that features are correct
#y = penguins_prepared_df[penguins_prepared_df.columns[0]]
#maybe assume target would be index 0?

X = penguins_prepared_df.drop('remainder__species',axis=1)
y = penguins_prepared_df['remainder__species']

y
```

[ ]:
```
0         Adelie
1         Adelie
2         Adelie
3         Adelie
4         Adelie
         …
339    Chinstrap
340    Chinstrap
341    Chinstrap
342    Chinstrap
343    Chinstrap
Name: remainder__species, Length: 344, dtype: object
```

[ ]:
```
# setup binary classification for Adelie vs. rest of species
# use the Adelie species as the positive class
# create a new target called y_adelie
# we need to split the data into train and split sets ??
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.2,␣
 ↪random_state=42)
y_adelie = (y == 'Adelie')
y_adelie
```

```
[ ]: 0       True
     1       True
     2       True
     3       True
     4       True
            …
     339    False
     340    False
     341    False
     342    False
     343    False
     Name: remainder__species, Length: 344, dtype: bool
```

```
[ ]: # build an SGDClassifier model using X and y
     # use random_state=42 for reproducibility
     from sklearn.linear_model import SGDClassifier
     sdg_clf =SGDClassifier(random_state=42)
     sdg_clf.fit(X,y_adelie)
```

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

```
[ ]: # compute the accuracy using cross_val_score with cv=10
     arr1 = cross_val_score(sdg_clf, X, y, cv=10, scoring='accuracy')
     arr1
```

```
[ ]: array([1.        , 1.        , 0.97142857, 1.        , 0.97058824,
            1.        , 1.        , 1.        , 1.        , 0.94117647])
```

```
[ ]: # compute the mean accuracy
     sum(arr1)/10
```

```
[ ]: 0.9883193277310924
```

```
[ ]: # predict the target using cross_val_predict with cv=10
     # call the result y_train_pred

     y_train_pred = cross_val_predict(sdg_clf, X, y_adelie, cv = 10)
     y_train_pred
```

```
[ ]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True, False,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True,  True,
            True,  True,  True,  True,  True, False,  True,  True,  True,
            True,  True,  True,  True,  True,  True,  True,  True, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False])
```

[ ]:
```
# compute the confusion matrix
cm=confusion_matrix(y_adelie,y_train_pred)
cm
```

[ ]:
```
array([[192,   0],
       [  2, 150]], dtype=int64)
```

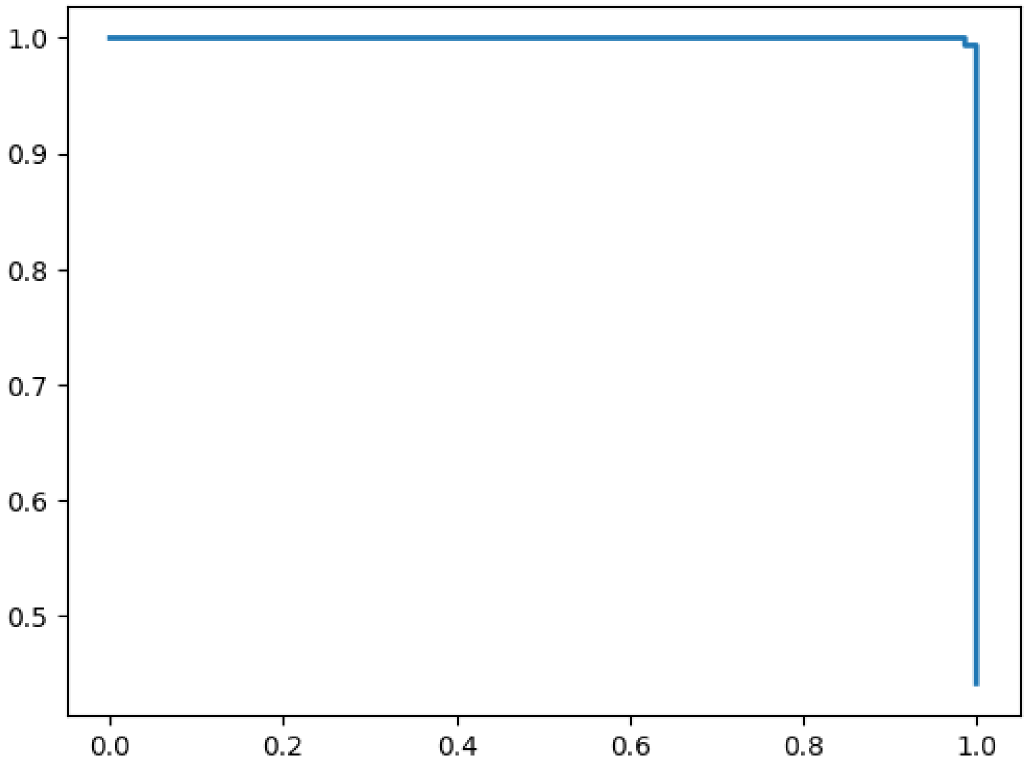
[ ]:
```
# compute the precision score using precision_score()
ps = precision_score(y_adelie,y_train_pred)
ps #that's kinda high
```

[ ]:
```
1.0
```

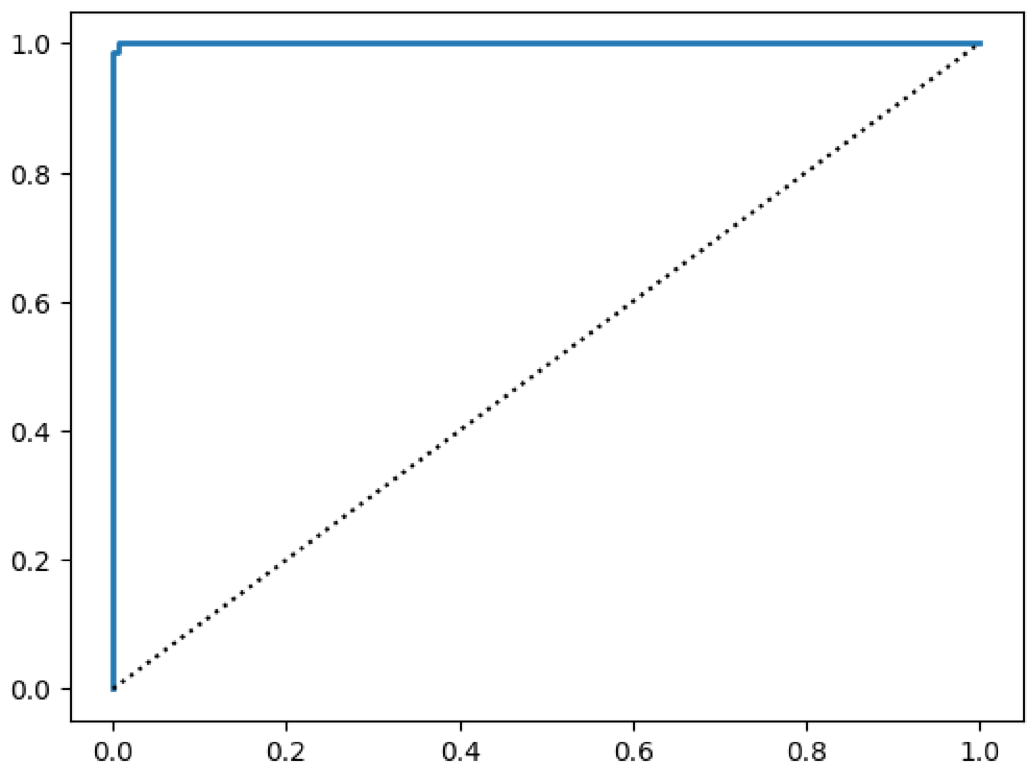
[ ]:
```
# compute the recall score using recall_score()
recall_score(y_adelie,y_train_pred)
```

[ ]: 0.9868421052631579

[ ]:
```
# draw the precision-recall curve
# call the result precisions, recalls, thresholds

y_scores = cross_val_predict(sdg_clf, X, y_adelie, cv=10,␣
 ↪method='decision_function')
y_scores

precisions, recalls, thresholds = precision_recall_curve(y_adelie, y_scores)

plt.plot(recalls, precisions, linewidth=2, label="Precision/Recall curve")
plt.show()
```



[ ]:
```
# call the result fpr, tpr, thresholds
# plot the roc curve

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



```
[ ]: # now let's do multiclass classification
     # build an SGDClassifier model using X and y
     # use random_state=42 for reproducibility

     sgd_c = SGDClassifier(random_state=42)
     sgd_c.fit(X, y)
```

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

```
[ ]: # show the mean accuracy using cross_val_score with cv=10
     arr2 = cross_val_score(sgd_c, X, y, cv=10, scoring='accuracy')
     sum(arr2)/10
```

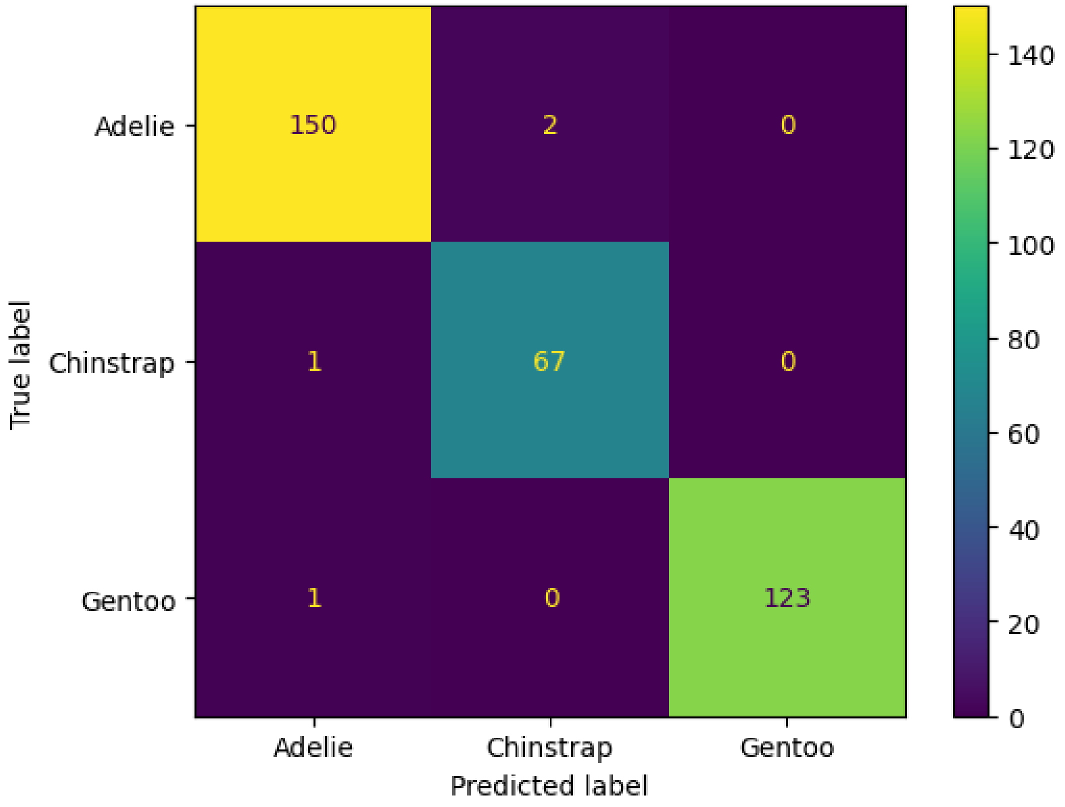
```
[ ]: 0.9883193277310924
```

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

     #multiclass
```

```
y_train_pred = cross_val_predict(sgd_c, X, y, cv = 10)
y_train_pred
```

```
[ ]: array(['Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Chinstrap', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Chinstrap', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
            'Adelie', 'Adelie', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Gentoo', 'Adelie', 'Gentoo', 'Gentoo', 'Gentoo', 'Gentoo',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Adelie', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap', 'Chinstrap',
            'Chinstrap', 'Chinstrap', 'Chinstrap'], dtype='<U9')
```

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


Adelie
Chinstrap
Gentoo
True label
Adelie
Chinstrap
Gentoo
Predicted label