# Machine Learning - Explain models with SHAP

## Classifier

https://www.kaggle.com/parulpandey/palmer-archipelago-antarctica-penguin-data

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, plot_confusion_matrix
import shap
```

In [ ]:
```
penguins = pd.read_csv('penguins_size.csv')
penguins.head()
```

Out[ ]:

| | species | island | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 3750.0 | MALE |
| **1** | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 3800.0 | FEMALE |
| **2** | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 3250.0 | FEMALE |
| **3** | Adelie | Torgersen | NaN | NaN | NaN | NaN | NaN |
| **4** | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 3450.0 | FEMALE |

In [ ]:
```
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

In [ ]:
```
penguins.describe(include='all')
```

Out[ ]:

| | species | island | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **count** | 344 | 344 | 342.000000 | 342.000000 | 342.000000 | 342.000000 | 334 |
| **unique** | 3 | 3 | NaN | NaN | NaN | NaN | 3 |
| **top** | Adelie | Biscoe | NaN | NaN | NaN | NaN | MALE |
| **freq** | 152 | 168 | NaN | NaN | NaN | NaN | 168 |
| **mean** | NaN | NaN | 43.921930 | 17.151170 | 200.915205 | 4201.754386 | NaN |
| **std** | NaN | NaN | 5.459584 | 1.974793 | 14.061714 | 801.954536 | NaN |
| **min** | NaN | NaN | 32.100000 | 13.100000 | 172.000000 | 2700.000000 | NaN |
| **25%** | NaN | NaN | 39.225000 | 15.600000 | 190.000000 | 3550.000000 | NaN |
| **50%** | NaN | NaN | 44.450000 | 17.300000 | 197.000000 | 4050.000000 | NaN |
| **75%** | NaN | NaN | 48.500000 | 18.700000 | 213.000000 | 4750.000000 | NaN |
| **max** | NaN | NaN | 59.600000 | 21.500000 | 231.000000 | 6300.000000 | NaN |

In [ ]:
```
penguins.isna().sum()
```

Out[ ]:
```
species              0
island               0
culmen_length_mm     2
culmen_depth_mm      2
```

```
flipper_length_mm     2
body_mass_g           2
sex                  10
dtype: int64
```

In [ ]:

```
penguins = penguins.dropna()
penguins = penguins[penguins['sex'] != '.']
penguins = penguins.reset_index(drop=True)
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            333 non-null    object 
 1   island             333 non-null    object 
 2   culmen_length_mm   333 non-null    float64
 3   culmen_depth_mm    333 non-null    float64
 4   flipper_length_mm  333 non-null    float64
 5   body_mass_g        333 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.3+ KB
```

In [ ]:

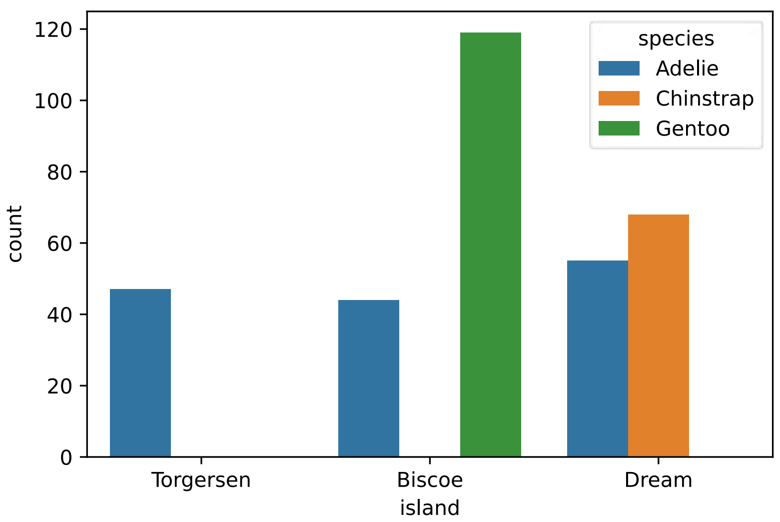
```
val_count_cols = ['species', 'island', 'sex']
for col in val_count_cols:
    print(penguins[col].value_counts())
    print()
```

```
Adelie       146
Gentoo       119
Chinstrap     68
Name: species, dtype: int64

Biscoe       163
Dream        123
Torgersen     47
Name: island, dtype: int64

MALE      168
FEMALE    165
Name: sex, dtype: int64
```

In [ ]:

```
sns.countplot(data=penguins, x='island', hue='species')
```

Out[ ]: <AxesSubplot:xlabel='island', ylabel='count'>




In [ ]:

```
penguins.groupby(['island', 'species']).count()
```

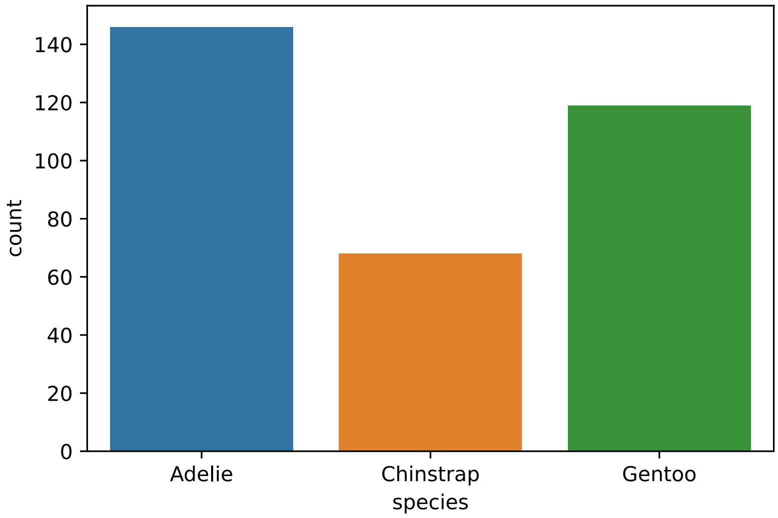
Out[ ]:

| | | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|
| **island** | **species** | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Biscoe** | **Adelie** | 44 | 44 | 44 | 44 | 44 |
| | **Gentoo** | 119 | 119 | 119 | 119 | 119 |
| **Dream** | **Adelie** | 55 | 55 | 55 | 55 | 55 |
| | **Chinstrap** | 68 | 68 | 68 | 68 | 68 |
| **Torgersen** | **Adelie** | 47 | 47 | 47 | 47 | 47 |

In [ ]:
```
sns.countplot(data=penguins, x='species')
```

Out[ ]: <AxesSubplot:xlabel='species', ylabel='count'>

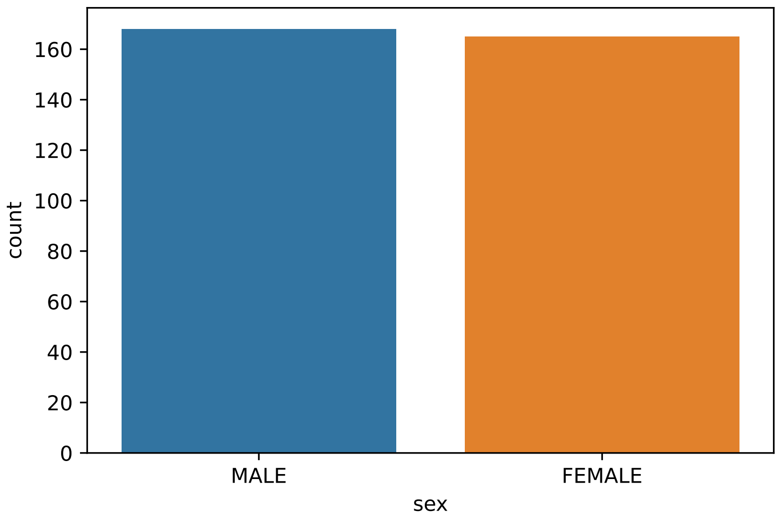



In [ ]:
```
penguins['species'].value_counts()
```

Out[ ]:
```
Adelie       146
Gentoo       119
Chinstrap     68
Name: species, dtype: int64
```

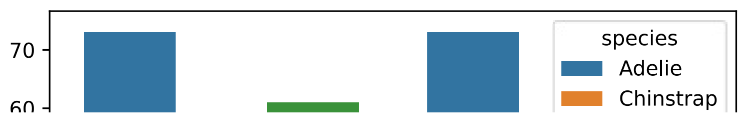
In [ ]:
```
sns.countplot(data=penguins, x='sex')
```

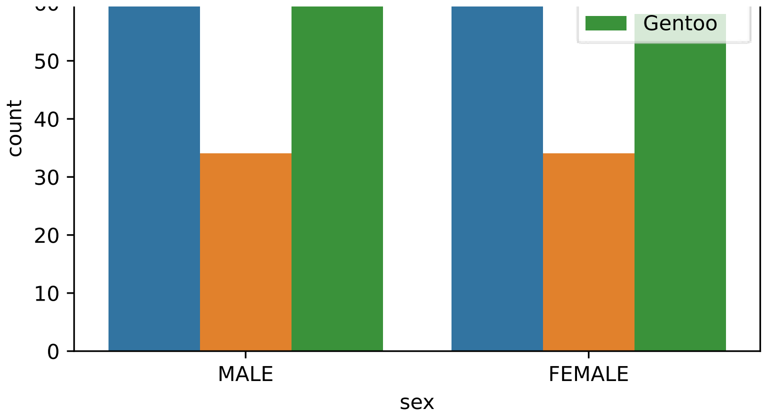
Out[ ]: <AxesSubplot:xlabel='sex', ylabel='count'>




In [ ]:
```
sns.countplot(data=penguins, x='sex', hue='species')
```

Out[ ]: <AxesSubplot:xlabel='sex', ylabel='count'>






In [ ]:
```
penguins.sex.value_counts()
```

Out[ ]:
```
MALE      168
FEMALE    165
Name: sex, dtype: int64
```

In [ ]:
```
penguins.groupby(['sex', 'species'])['sex'].count()
```

Out[ ]:
```
sex     species
FEMALE  Adelie       73
        Chinstrap    34
        Gentoo       58
MALE    Adelie       73
        Chinstrap    34
        Gentoo       61
Name: sex, dtype: int64
```

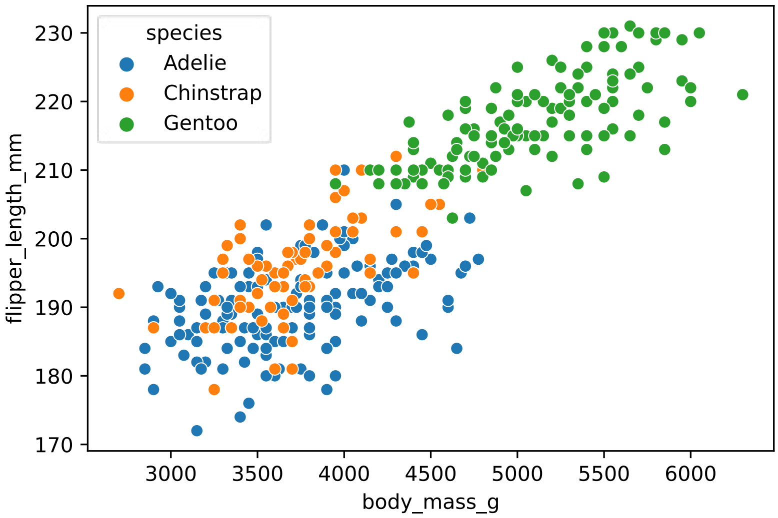
In [ ]:
```
penguins.corr()
```

Out[ ]:

| | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g |
|---|---|---|---|---|
| **culmen_length_mm** | 1.000000 | -0.228626 | 0.653096 | 0.589451 |
| **culmen_depth_mm** | -0.228626 | 1.000000 | -0.577792 | -0.472016 |
| **flipper_length_mm** | 0.653096 | -0.577792 | 1.000000 | 0.872979 |
| **body_mass_g** | 0.589451 | -0.472016 | 0.872979 | 1.000000 |

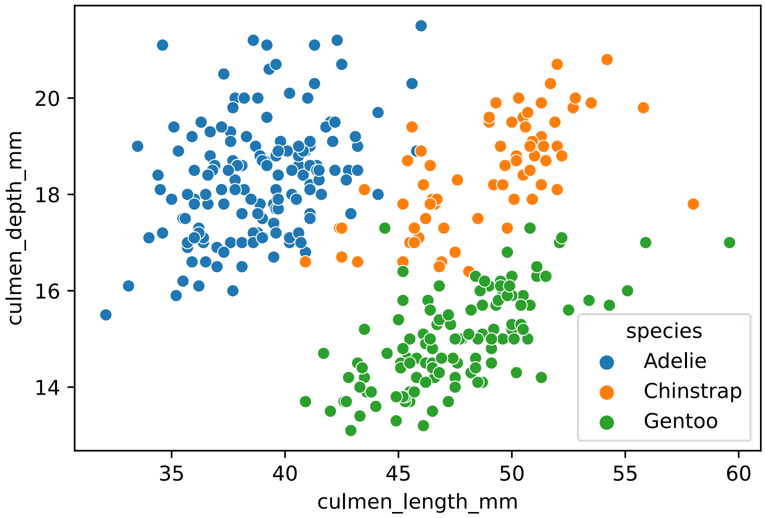
In [ ]:
```
sns.scatterplot(data=penguins, x='body_mass_g', y='flipper_length_mm', hue='species')
```

Out[ ]:
```
<AxesSubplot:xlabel='body_mass_g', ylabel='flipper_length_mm'>
```



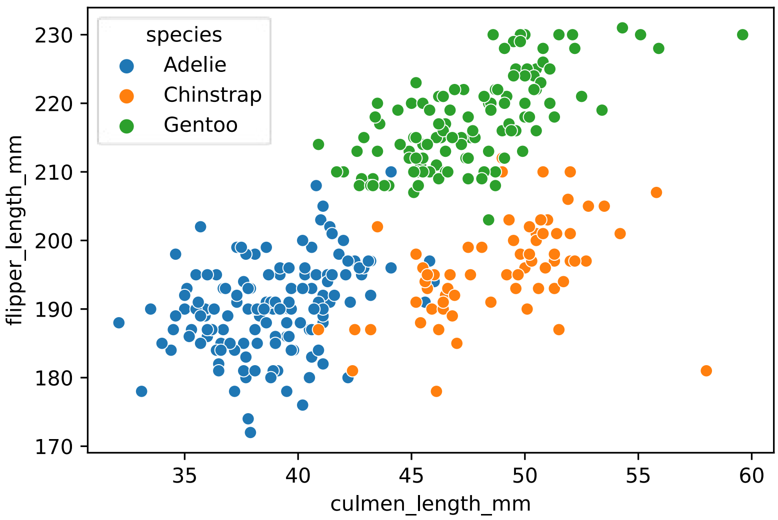

In [ ]:
```
sns.scatterplot(data=penguins, x='culmen_length_mm', y='culmen_depth_mm', hue='species')
```

Out[ ]: <AxesSubplot:xlabel='culmen_length_mm', ylabel='culmen_depth_mm'>



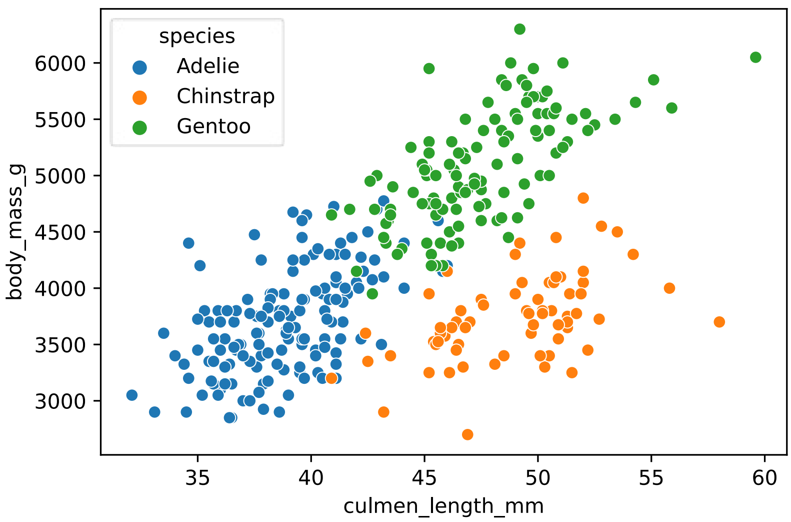

In [ ]:
```
sns.scatterplot(data=penguins, x='culmen_length_mm', y='flipper_length_mm', hue='species')
```

Out[ ]: <AxesSubplot:xlabel='culmen_length_mm', ylabel='flipper_length_mm'>




In [ ]:
```
sns.scatterplot(data=penguins, x='culmen_length_mm', y='body_mass_g', hue='species')
```

Out[ ]: <AxesSubplot:xlabel='culmen_length_mm', ylabel='body_mass_g'>




In [ ]:
```
px.scatter_3d(penguins, x='culmen_length_mm', y='culmen_depth_mm', z='flipper_length_mm', color='species', title=
```

In [ ]:
```
px.scatter_3d(penguins, x='culmen_length_mm', y='culmen_depth_mm', z='body_mass_g', color='species', title='Pengu
```

In [ ]:
```
px.scatter_3d(penguins, x='culmen_depth_mm', y='flipper_length_mm', z='body_mass_g', color='species', title='Peng
```

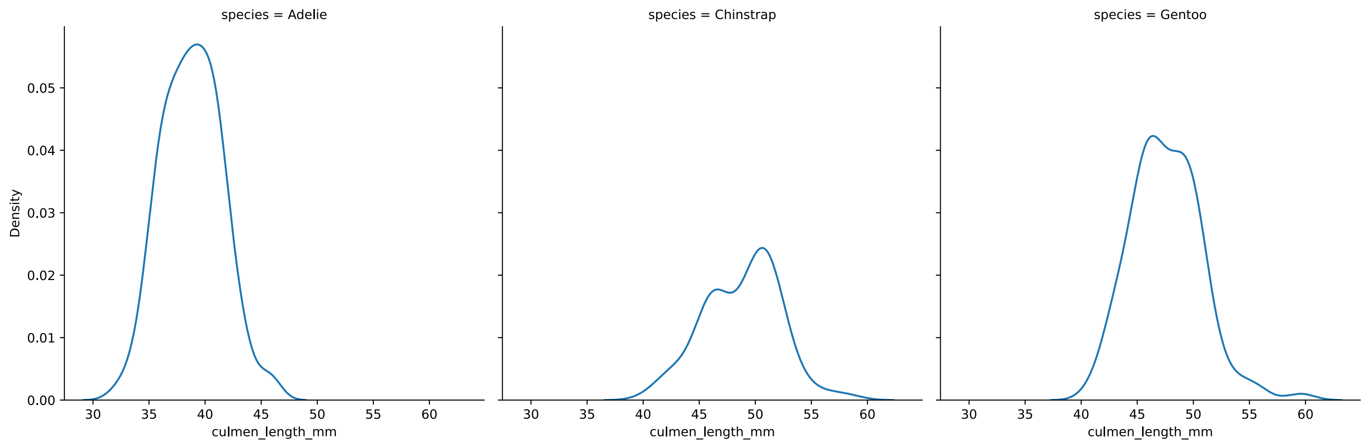
In [ ]:
```
px.scatter_3d(penguins, x='culmen_length_mm', y='flipper_length_mm', z='body_mass_g', color='species', title='Per
```

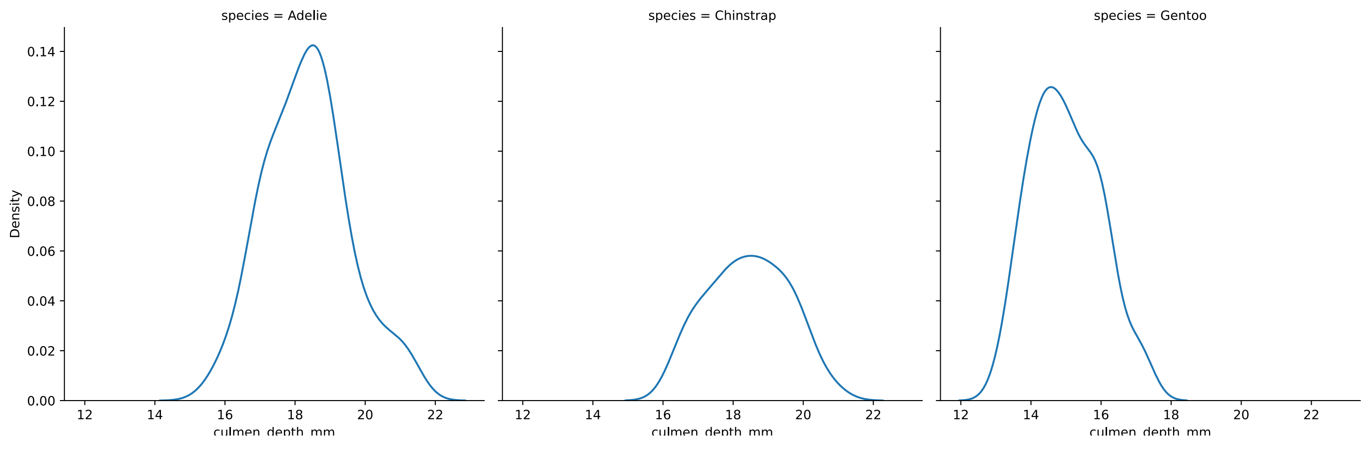
In [ ]:
```
penguins = penguins[['island', 'species', 'culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_
penguins.head()
```

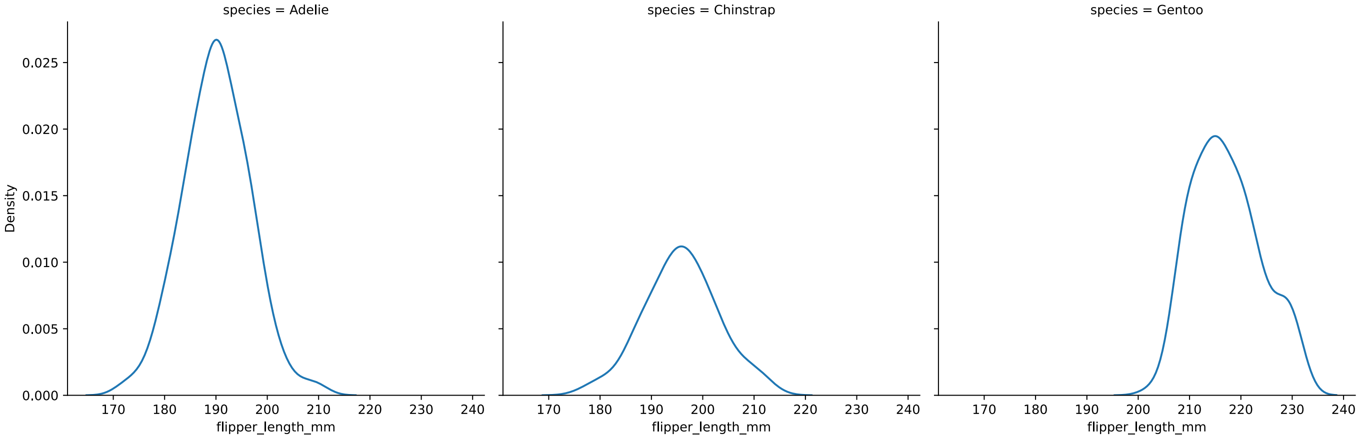
Out[ ]:

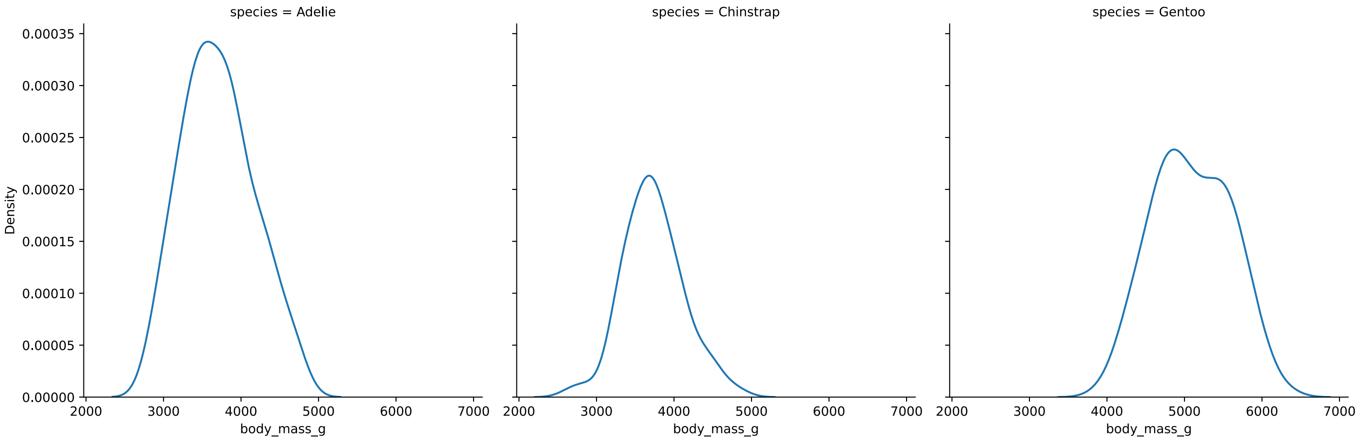
| | island | species | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Torgersen | Adelie | 39.1 | 18.7 | 181.0 | 3750.0 | MALE |
| **1** | Torgersen | Adelie | 39.5 | 17.4 | 186.0 | 3800.0 | FEMALE |
| **2** | Torgersen | Adelie | 40.3 | 18.0 | 195.0 | 3250.0 | FEMALE |
| **3** | Torgersen | Adelie | 36.7 | 19.3 | 193.0 | 3450.0 | FEMALE |
| **4** | Torgersen | Adelie | 39.3 | 20.6 | 190.0 | 3650.0 | MALE |

In [ ]:
```
plot_cols = ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']

for col in plot_cols:
    sns.displot(data=penguins, x=col, col='species', kind='kde')
```






In [ ]:
```
penguins.groupby('species').agg({'culmen_length_mm': ['count', 'min', 'max', 'mean', 'median'], 'culmen_depth_mm'
```

Out[ ]:

| | culmen_length_mm | | | | | culmen_depth_mm | | | | | flipper_length_mm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | count | min | max | mean | median | count | min | max | mean | median | count | min | max | mean | median | count | min |
| **species** | | | | | | | | | | | | | | | | | |
| **Adelie** | 146 | 32.1 | 46.0 | 38.823973 | 38.85 | 146 | 15.5 | 21.5 | 18.347260 | 18.40 | 146 | 172.0 | 210.0 | 190.102740 | 190.0 | 146 | 2850.0 |
| **Chinstrap** | 68 | 40.9 | 58.0 | 48.833824 | 49.55 | 68 | 16.4 | 20.8 | 18.420588 | 18.45 | 68 | 178.0 | 212.0 | 195.823529 | 196.0 | 68 | 2700.0 |
| **Gentoo** | 119 | 40.9 | 59.6 | 47.568067 | 47.40 | 119 | 13.1 | 17.3 | 14.996639 | 15.00 | 119 | 203.0 | 231.0 | 217.235294 | 216.0 | 119 | 3950.0 |

In [ ]:
```
X = penguins.drop(columns='sex')
y = penguins['sex']
```

```
print(X.shape, y.shape)
```

(333, 6) (333,)

In [ ]:
```
print(y[0])
print(y[120])

y = LabelEncoder().fit_transform(y)

print(y[0])
print(y[120])
```

```
MALE
FEMALE
1
0
```

In [ ]:
```
X = pd.get_dummies(X)
X.head()
```

Out[ ]:

| | culmen_length_mm | culmen_depth_mm | flipper_length_mm | body_mass_g | island_Biscoe | island_Dream | island_Torgersen | species_Adelie | spe |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 39.1 | 18.7 | 181.0 | 3750.0 | 0 | 0 | 1 | 1 | |
| **1** | 39.5 | 17.4 | 186.0 | 3800.0 | 0 | 0 | 1 | 1 | |
| **2** | 40.3 | 18.0 | 195.0 | 3250.0 | 0 | 0 | 1 | 1 | |
| **3** | 36.7 | 19.3 | 193.0 | 3450.0 | 0 | 0 | 1 | 1 | |
| **4** | 39.3 | 20.6 | 190.0 | 3650.0 | 0 | 0 | 1 | 1 | |

# Random Forest Classifier -> predict sex

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

(266, 10) (266,) (67, 10) (67,)

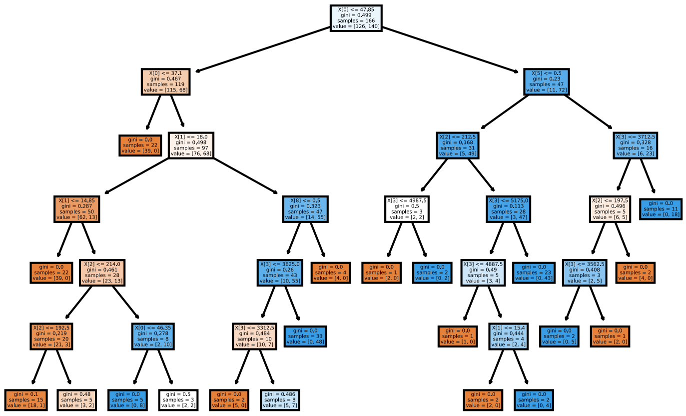
In [ ]:
```
model = RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=6, random_state=7)
model.fit(X_train, y_train)
```

Out[ ]: RandomForestClassifier(max_depth=6, random_state=7)

In [ ]:
```
estimator = model.estimators_[5]

plot_tree(estimator, filled = True)
```

Out[ ]:
```
[Text(173.6, 201.90857142857143, 'X[0] <= 47.85\ngini = 0.499\nsamples = 166\nvalue = [126, 140]'),
 Text(80.60000000000001, 170.84571428571428, 'X[0] <= 37.1\ngini = 0.467\nsamples = 119\nvalue = [115, 68]'),
 Text(68.2, 139.78285714285715, 'gini = 0.0\nsamples = 22\nvalue = [39, 0]'),
 Text(93.0, 139.78285714285715, 'X[1] <= 18.0\ngini = 0.498\nsamples = 97\nvalue = [76, 68]'),
 Text(37.2, 108.72, 'X[1] <= 14.85\ngini = 0.287\nsamples = 50\nvalue = [62, 13]'),
 Text(24.8, 77.65714285714284, 'gini = 0.0\nsamples = 22\nvalue = [39, 0]'),
 Text(49.6, 77.65714285714284, 'X[2] <= 214.0\ngini = 0.461\nsamples = 28\nvalue = [23, 13]'),
 Text(24.8, 46.59428571428572, 'X[2] <= 192.5\ngini = 0.219\nsamples = 20\nvalue = [21, 3]'),
 Text(12.4, 15.531428571428563, 'gini = 0.1\nsamples = 15\nvalue = [18, 1]'),
 Text(37.2, 15.531428571428563, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]'),
 Text(74.4, 46.59428571428572, 'X[0] <= 46.35\ngini = 0.278\nsamples = 8\nvalue = [2, 10]'),
 Text(62.0, 15.531428571428563, 'gini = 0.0\nsamples = 5\nvalue = [0, 8]'),
 Text(86.8, 15.531428571428563, 'gini = 0.5\nsamples = 3\nvalue = [2, 2]'),
 Text(148.8, 108.72, 'X[8] <= 0.5\ngini = 0.323\nsamples = 47\nvalue = [14, 55]'),
 Text(136.4, 77.65714285714284, 'X[3] <= 3625.0\ngini = 0.26\nsamples = 43\nvalue = [10, 55]'),
 Text(124.0, 46.59428571428572, 'X[3] <= 3312.5\ngini = 0.484\nsamples = 10\nvalue = [10, 7]'),
 Text(111.60000000000001, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [5, 0]'),
 Text(136.4, 15.531428571428563, 'gini = 0.486\nsamples = 8\nvalue = [5, 7]'),
 Text(148.8, 46.59428571428572, 'gini = 0.0\nsamples = 33\nvalue = [0, 48]'),
 Text(161.20000000000002, 77.65714285714284, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(266.6, 170.84571428571428, 'X[5] <= 0.5\ngini = 0.23\nsamples = 47\nvalue = [11, 72]'),
 Text(223.20000000000002, 139.78285714285715, 'X[2] <= 212.5\ngini = 0.168\nsamples = 31\nvalue = [5, 49]'),
 Text(198.4, 108.72, 'X[3] <= 4987.5\ngini = 0.5\nsamples = 3\nvalue = [2, 2]'),
 Text(186.0, 77.65714285714284, 'gini = 0.0\nsamples = 1\nvalue = [2, 0]'),
 Text(210.8, 77.65714285714284, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(248.0, 108.72, 'X[3] <= 5175.0\ngini = 0.113\nsamples = 28\nvalue = [3, 47]'),
```

```
Text(235.6, 77.65714285714284, 'X[3] <= 4887.5\ngini = 0.49\nsamples = 5\nvalue = [3, 4]'),
Text(223.20000000000002, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(248.0, 46.59428571428572, 'X[1] <= 15.4\ngini = 0.444\nsamples = 4\nvalue = [2, 4]'),
Text(235.6, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
Text(260.40000000000003, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [0, 4]'),
Text(260.40000000000003, 77.65714285714284, 'gini = 0.0\nsamples = 23\nvalue = [0, 43]'),
Text(310.0, 139.78285714285715, 'X[3] <= 3712.5\ngini = 0.328\nsamples = 16\nvalue = [6, 23]'),
Text(297.6, 108.72, 'X[2] <= 197.5\ngini = 0.496\nsamples = 5\nvalue = [6, 5]'),
Text(285.2, 77.65714285714284, 'X[3] <= 3562.5\ngini = 0.408\nsamples = 3\nvalue = [2, 5]'),
Text(272.8, 46.59428571428572, 'gini = 0.0\nsamples = 2\nvalue = [0, 5]'),
Text(297.6, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [2, 0]'),
Text(310.0, 77.65714285714284, 'gini = 0.0\nsamples = 2\nvalue = [4, 0]'),
Text(322.40000000000003, 108.72, 'gini = 0.0\nsamples = 11\nvalue = [0, 18]')]
```



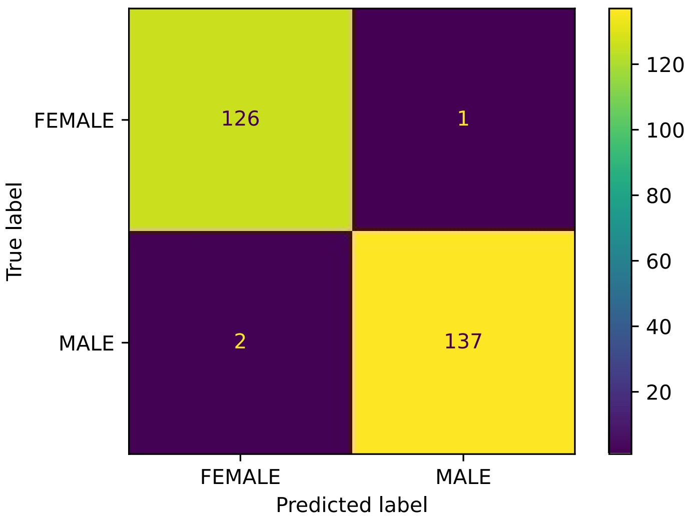
In [ ]:
```
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)
```

In [ ]:
```
print(classification_report(y_train, y_pred_train))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       127
           1       0.99      0.99      0.99       139

    accuracy                           0.99       266
   macro avg       0.99      0.99      0.99       266
weighted avg       0.99      0.99      0.99       266
```

In [ ]:
```
plot_confusion_matrix(model, X_train, y_train, display_labels=['FEMALE', 'MALE'])
```

Out[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1fbf2756520>



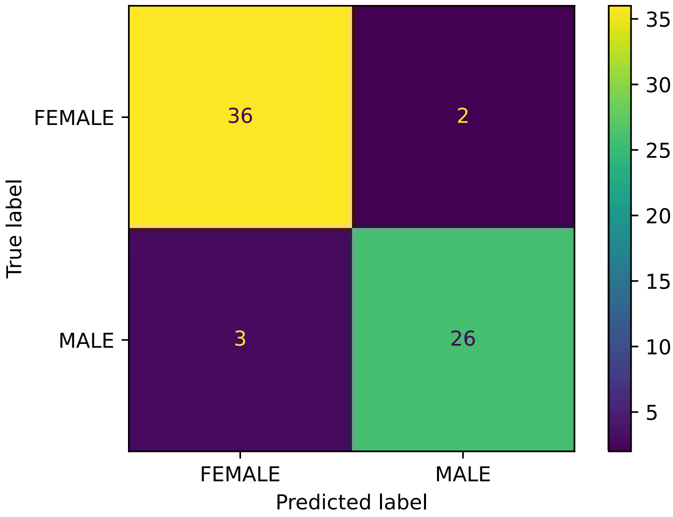

In [ ]:
```
print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support
```

```
           0       0.92      0.95      0.94        38
           1       0.93      0.90      0.91        29

    accuracy                           0.93        67
   macro avg       0.93      0.92      0.92        67
weighted avg       0.93      0.93      0.93        67
```

In [ ]:
```
plot_confusion_matrix(model, X_test, y_test, display_labels=['FEMALE', 'MALE'])
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1fbf14ede50>
```



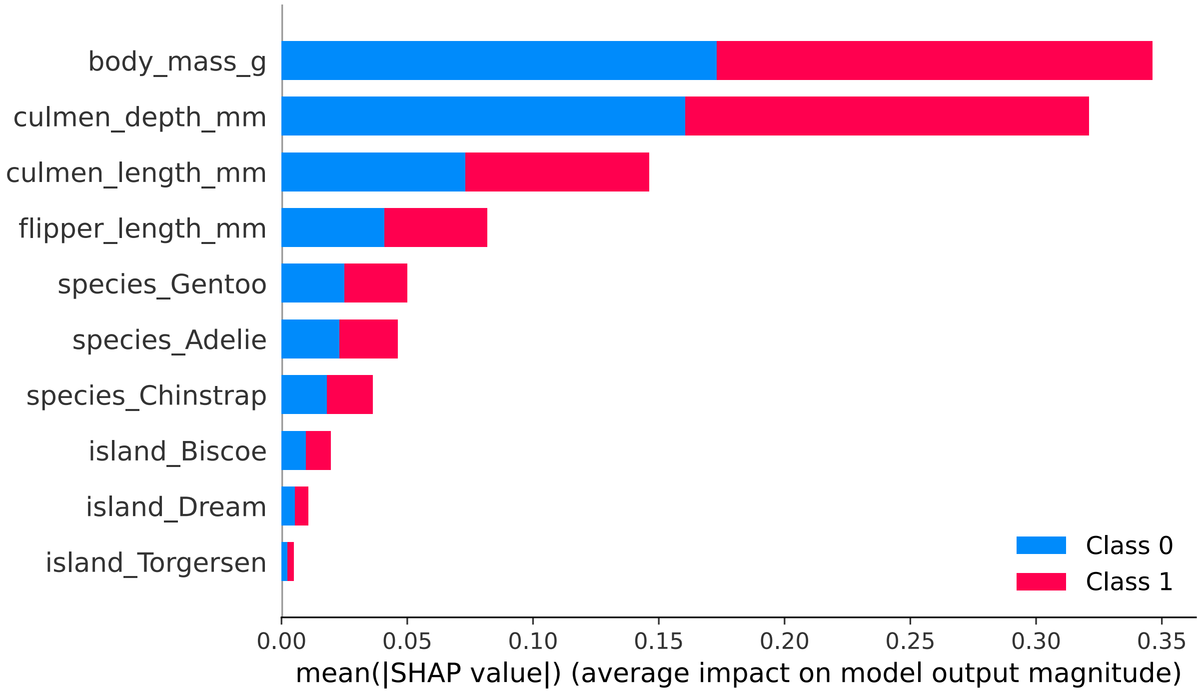

In [ ]:
```
y_pred_proba = model.predict_proba(X_test)
y_pred_proba
```

Out[ ]:
```
array([[9.89259329e-01, 1.07406711e-02],
       [7.58275514e-02, 9.24172449e-01],
       [9.16221287e-02, 9.08377871e-01],
       [2.82487259e-02, 9.71751274e-01],
       [2.32558140e-04, 9.99767442e-01],
       [9.32253144e-01, 6.77468562e-02],
       [9.73028779e-01, 2.69712211e-02],
       [1.99836185e-02, 9.80016381e-01],
       [9.65358230e-01, 3.46417700e-02],
       [9.31472444e-01, 6.85275562e-02],
       [9.92937595e-01, 7.06240495e-03],
       [2.32558140e-04, 9.99767442e-01],
       [3.27363366e-01, 6.72636634e-01],
       [9.74020928e-01, 2.59790716e-02],
       [5.63459880e-02, 9.43654012e-01],
       [9.91662801e-01, 8.33719887e-03],
       [1.04634463e-01, 8.95365537e-01],
       [7.55851968e-01, 2.44148032e-01],
       [2.32558140e-04, 9.99767442e-01],
       [3.71320084e-02, 9.62867992e-01],
       [2.41802044e-01, 7.58197956e-01],
       [9.90936874e-01, 9.06312645e-03],
       [7.42169531e-01, 2.57830469e-01],
       [9.74910912e-01, 2.50890877e-02],
       [6.90924400e-01, 3.09075600e-01],
       [3.31674499e-01, 6.68325501e-01],
       [9.81214906e-01, 1.87850940e-02],
       [2.32558140e-04, 9.99767442e-01],
       [8.68116075e-01, 1.31883925e-01],
       [9.31979856e-01, 6.80201439e-02],
       [4.69294647e-01, 5.30705353e-01],
       [7.99055667e-01, 2.00944333e-01],
       [9.52142851e-01, 4.78571490e-02],
       [9.30987469e-01, 6.90125312e-02],
       [5.94129565e-01, 4.05870435e-01],
       [3.95260586e-01, 6.04739414e-01],
       [9.89500095e-01, 1.04999050e-02],
       [4.05742284e-02, 9.59425772e-01],
       [5.03897251e-01, 4.96102749e-01],
       [4.01636754e-01, 5.98363246e-01],
       [9.47727368e-01, 5.22726323e-02],
```

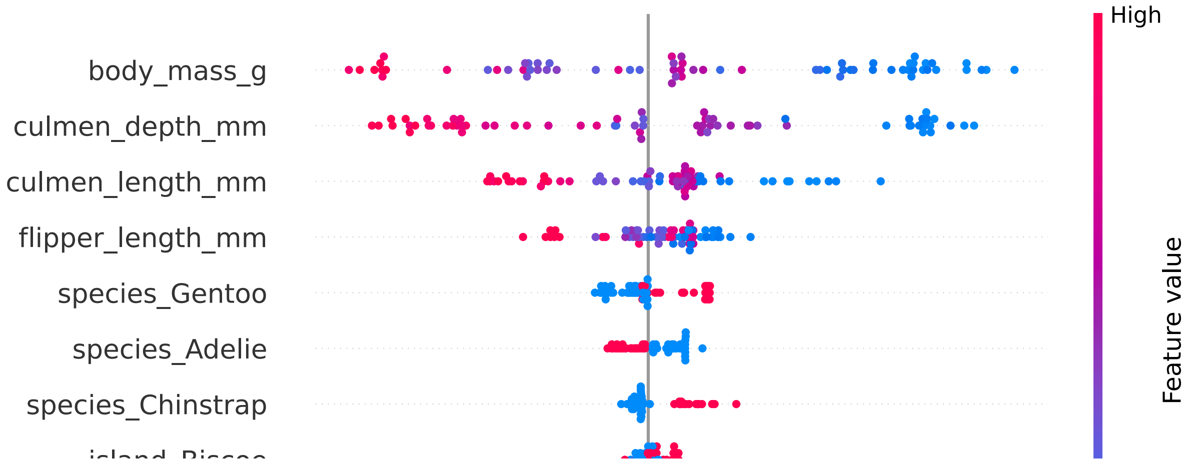
```
 [9.73248110e-01, 2.67518899e-02],
 [8.49187832e-01, 1.50812168e-01],
 [7.92965954e-01, 2.07034046e-01],
 [1.70951253e-02, 9.82904875e-01],
 [9.45799971e-01, 5.42000285e-02],
 [9.80412801e-01, 1.95871989e-02],
 [9.55511968e-01, 4.44880318e-02],
 [7.41874325e-01, 2.58125675e-01],
 [9.86893257e-01, 1.31067426e-02],
 [3.53769841e-02, 9.64623016e-01],
 [2.75530303e-01, 7.24469697e-01],
 [3.95363170e-01, 6.04636830e-01],
 [6.34111888e-01, 3.65888112e-01],
 [9.91866166e-01, 8.13383352e-03],
 [9.84092662e-01, 1.59073378e-02],
 [7.72590208e-01, 2.27409792e-01],
 [8.14668978e-01, 1.85331022e-01],
 [5.24670855e-02, 9.47532914e-01],
 [3.76105305e-01, 6.23894695e-01],
 [2.32558140e-04, 9.99767442e-01],
 [1.87646663e-01, 8.12353337e-01],
 [3.50015813e-01, 6.49984187e-01],
 [9.57614689e-01, 4.23853107e-02],
 [1.22232046e-01, 8.77767954e-01],
 [9.78012878e-01, 2.19871224e-02],
 [5.34451167e-01, 4.65548833e-01]])
```

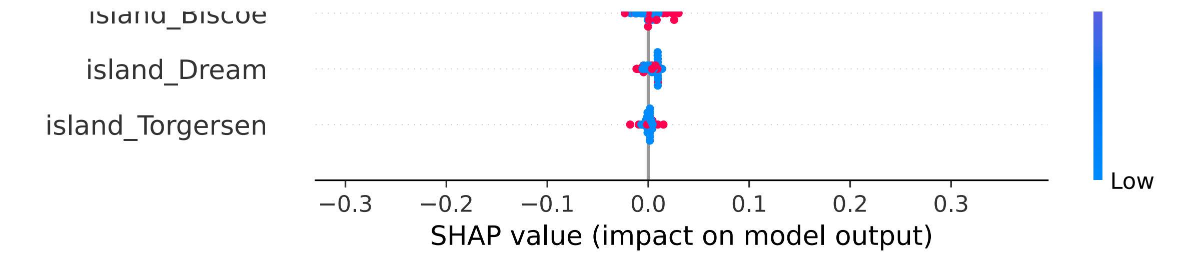
In [ ]:
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
```

In [ ]:
```
shap.summary_plot(shap_values, X_test)
```

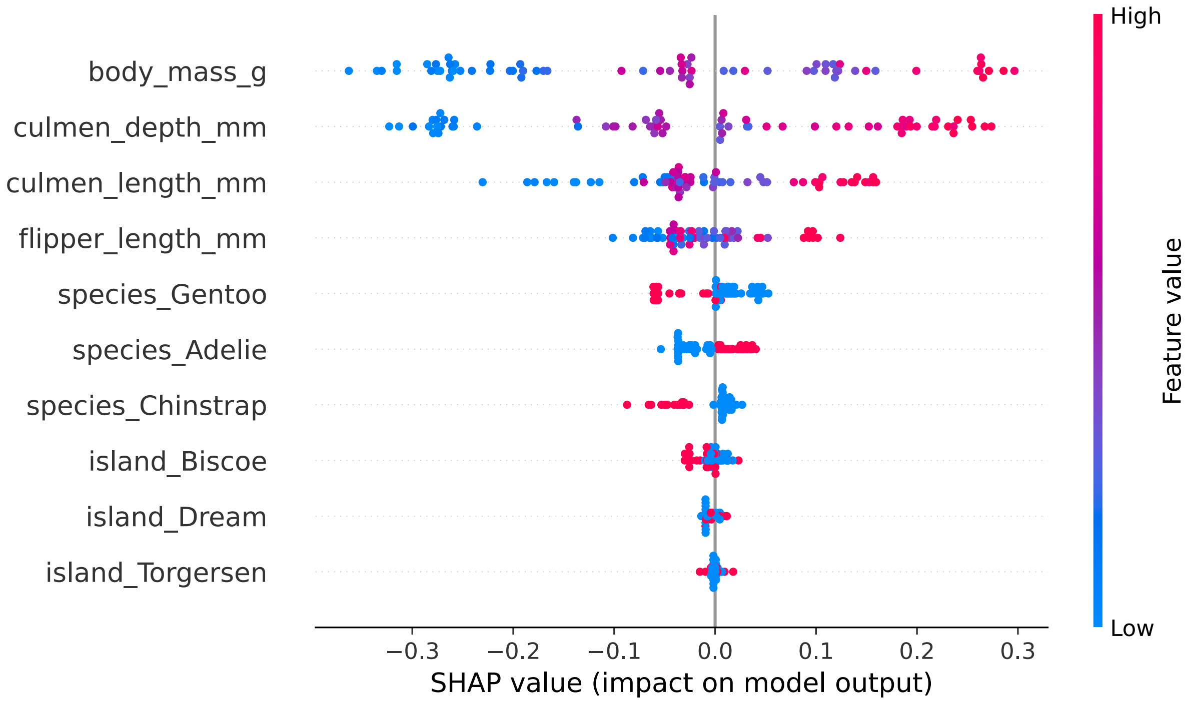



In [ ]:
```
shap.summary_plot(shap_values[0], X_test)
```

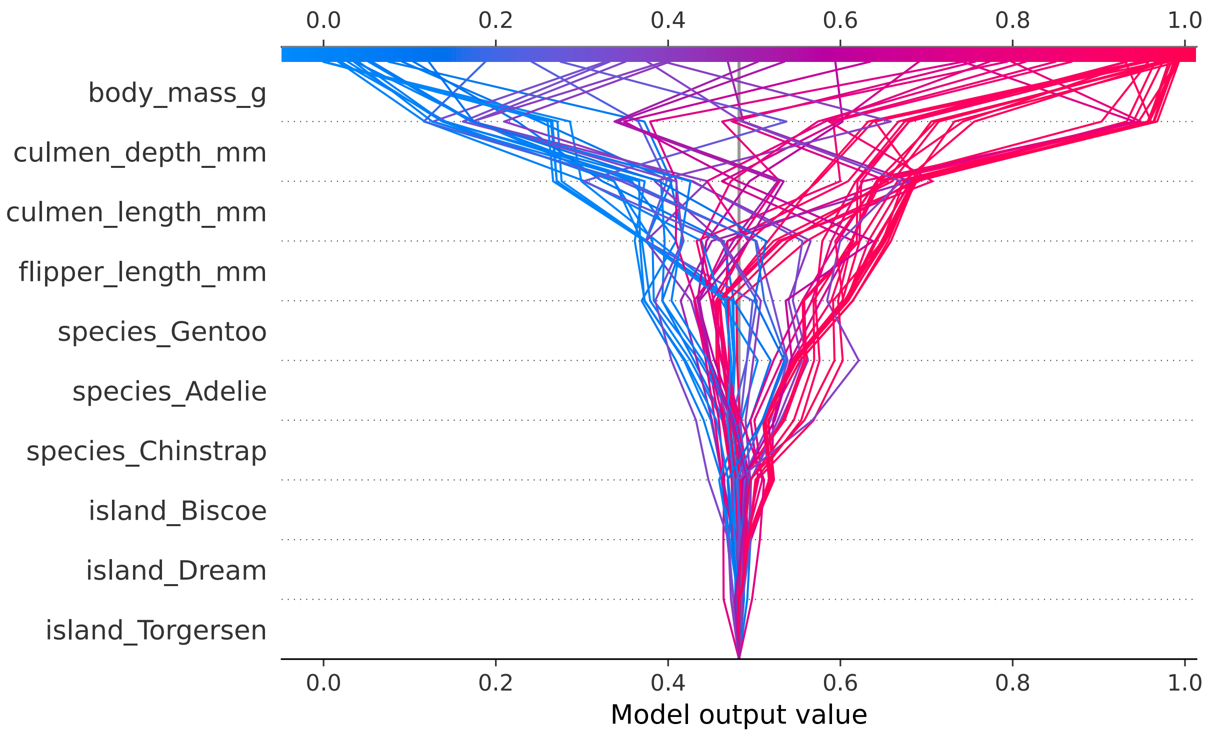




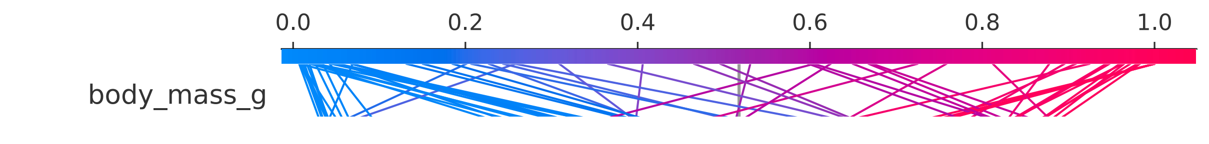

In [ ]:
```
shap.summary_plot(shap_values[1], X_test)
```

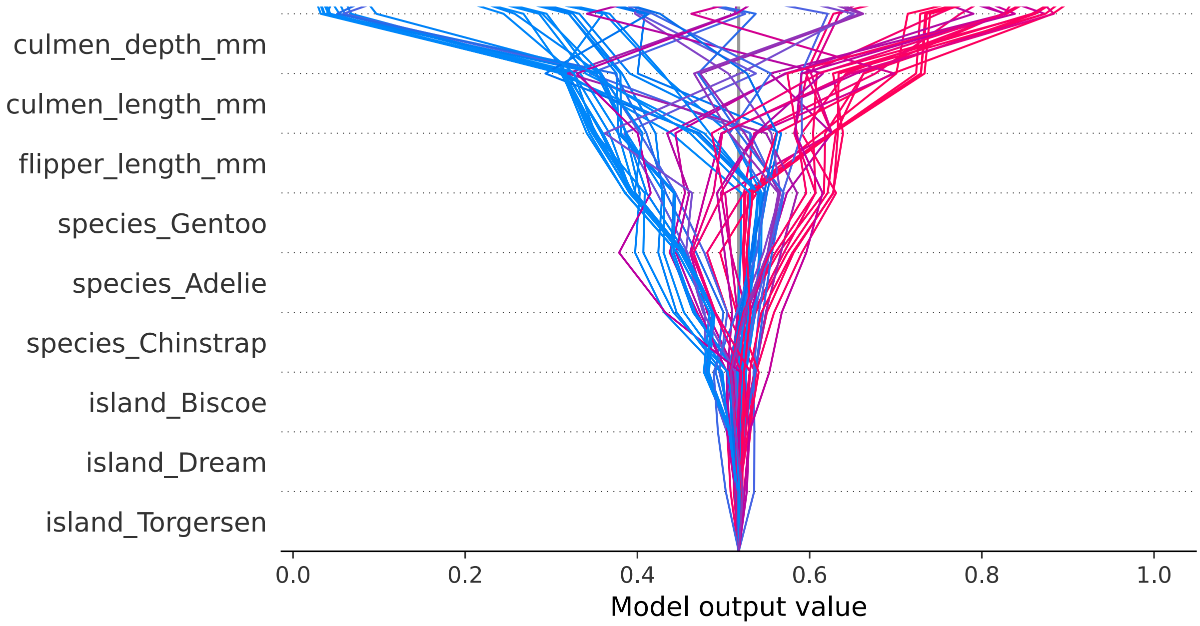



In [ ]:
```
shap.decision_plot(explainer.expected_value[0], shap_values[0], X_test.iloc[0, :])
```

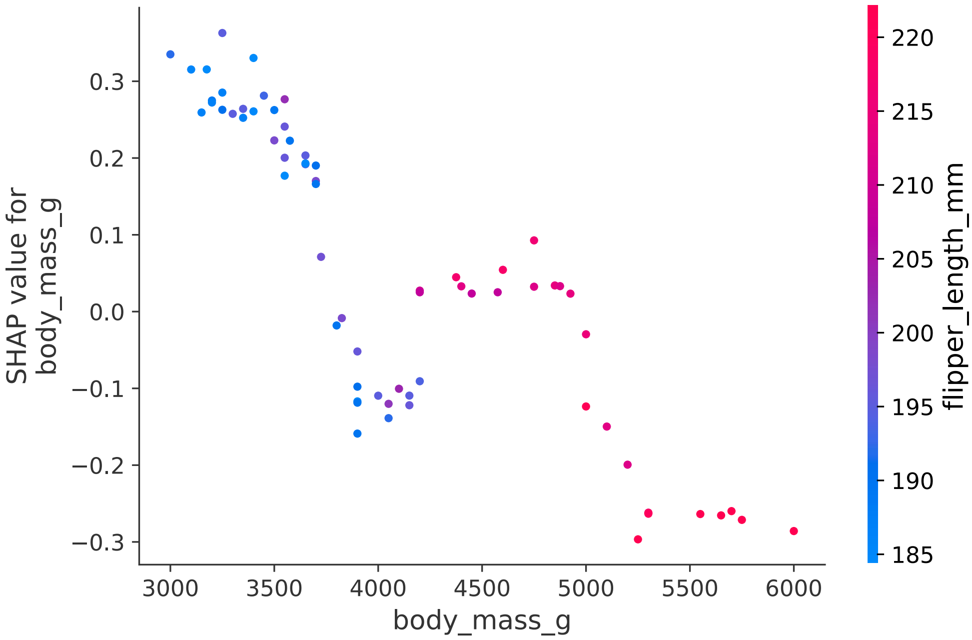



In [ ]:
```
shap.decision_plot(explainer.expected_value[1], shap_values[1], X_test.iloc[1, :])
```

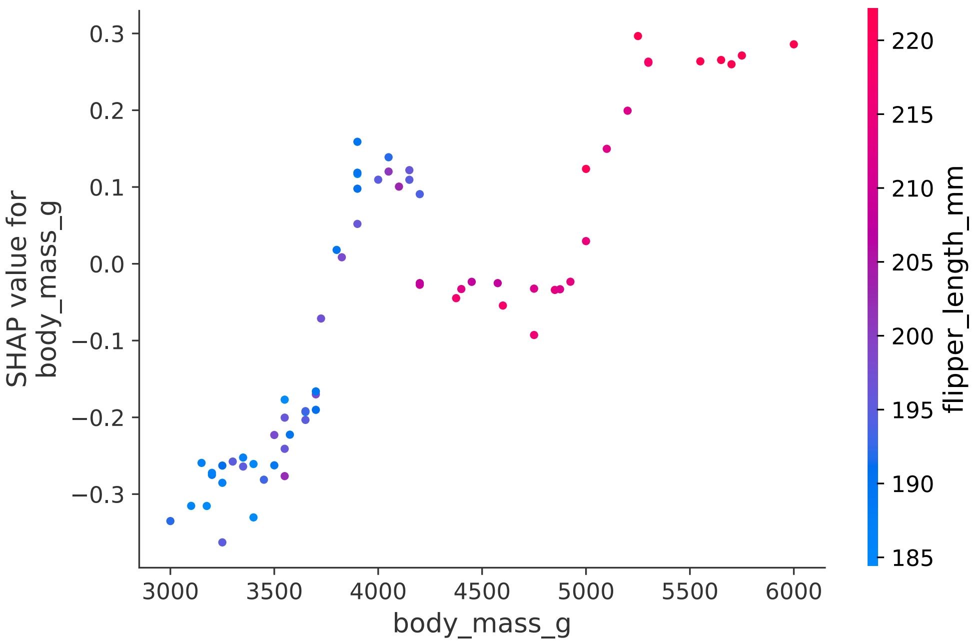




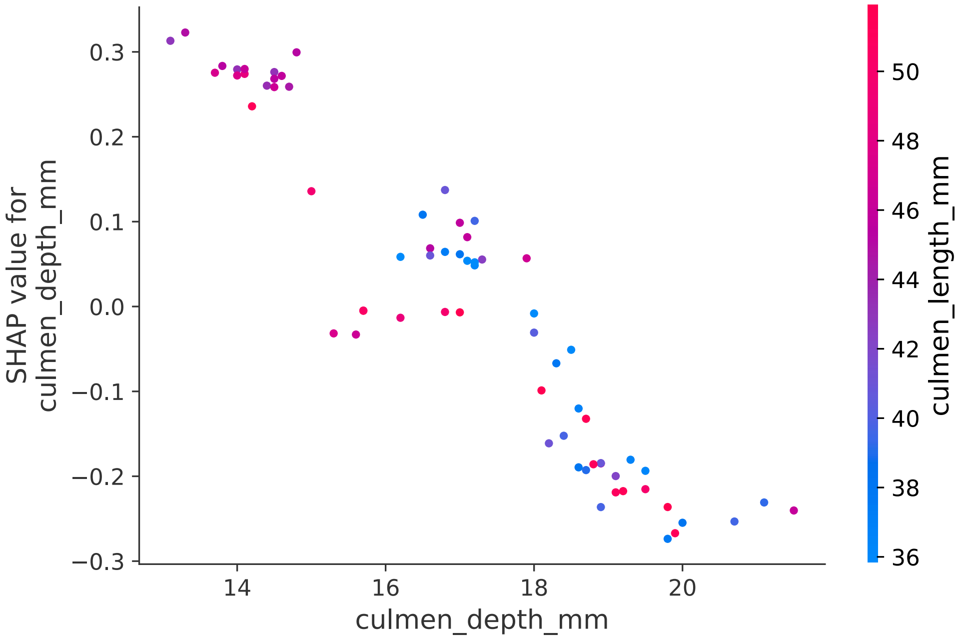

In [ ]:
```
shap.dependence_plot('body_mass_g', shap_values[0], X_test)
```



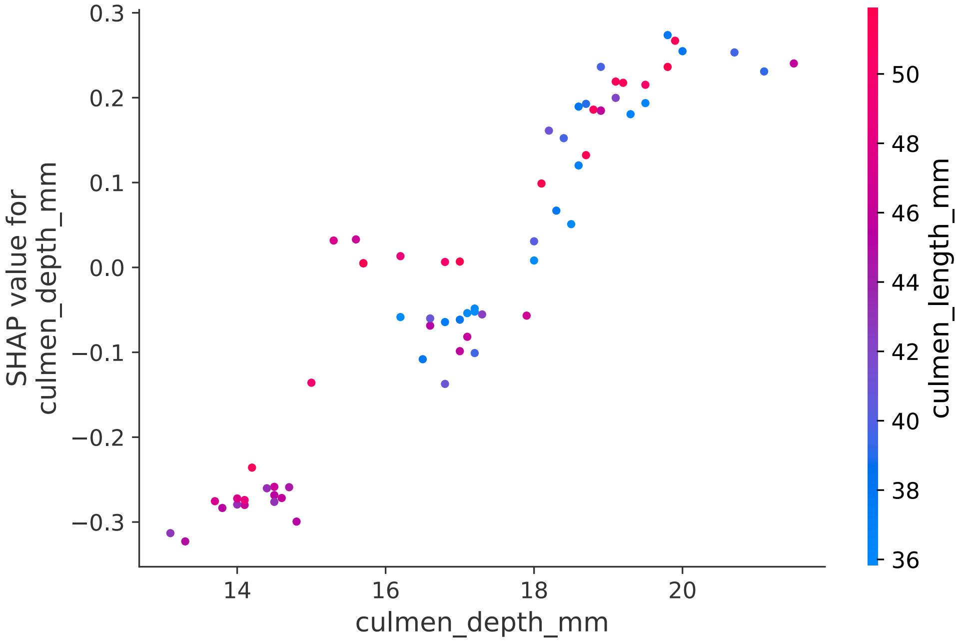

In [ ]:
```
shap.dependence_plot('body_mass_g', shap_values[1], X_test)
```



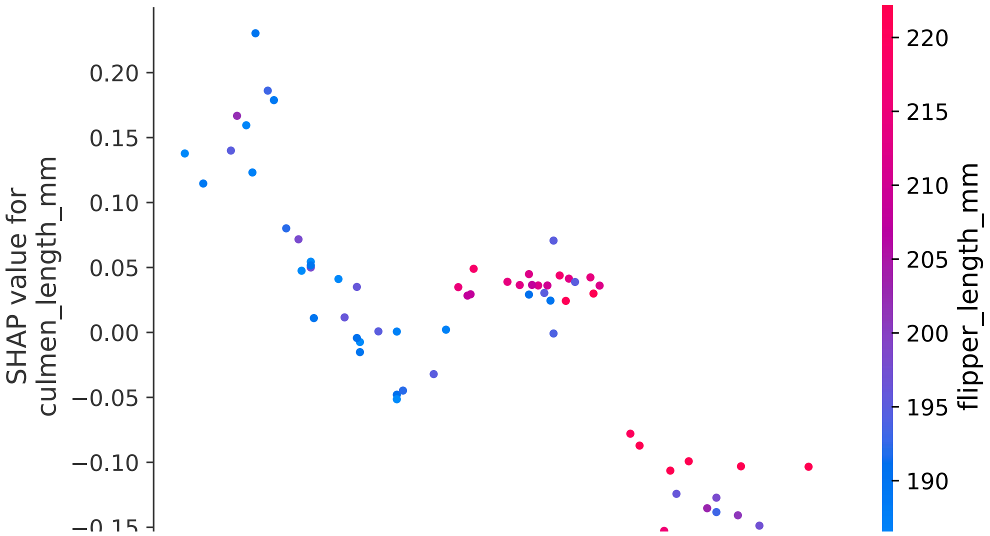
In [ ]:
```
shap.dependence_plot('culmen_depth_mm', shap_values[0], X_test)
```

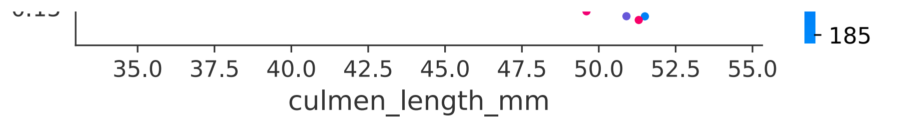



In [ ]:
```
shap.dependence_plot('culmen_depth_mm', shap_values[1], X_test)
```



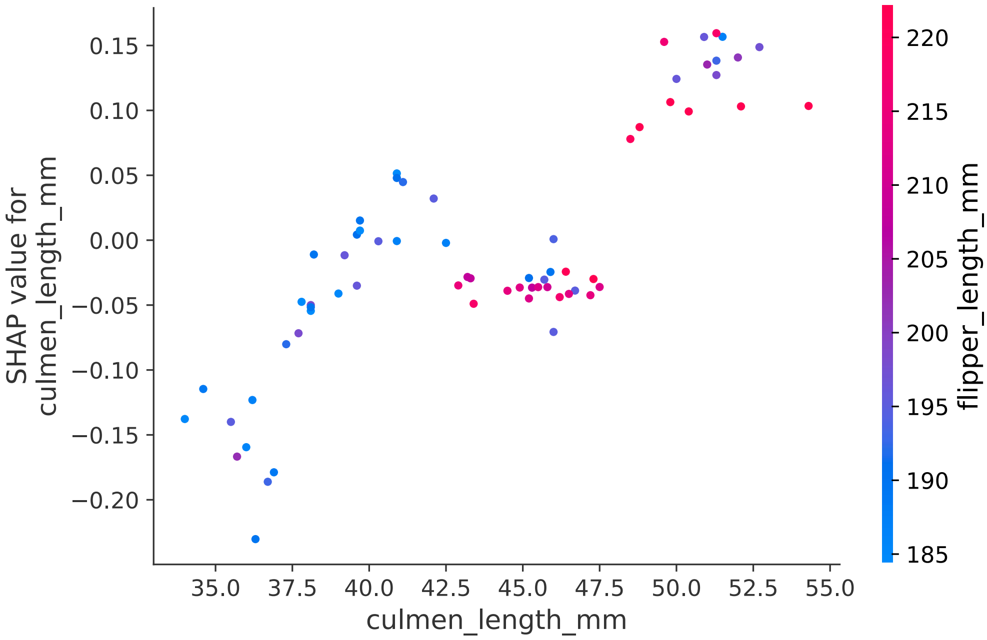

In [ ]:
```
shap.dependence_plot('culmen_length_mm', shap_values[0], X_test)
```





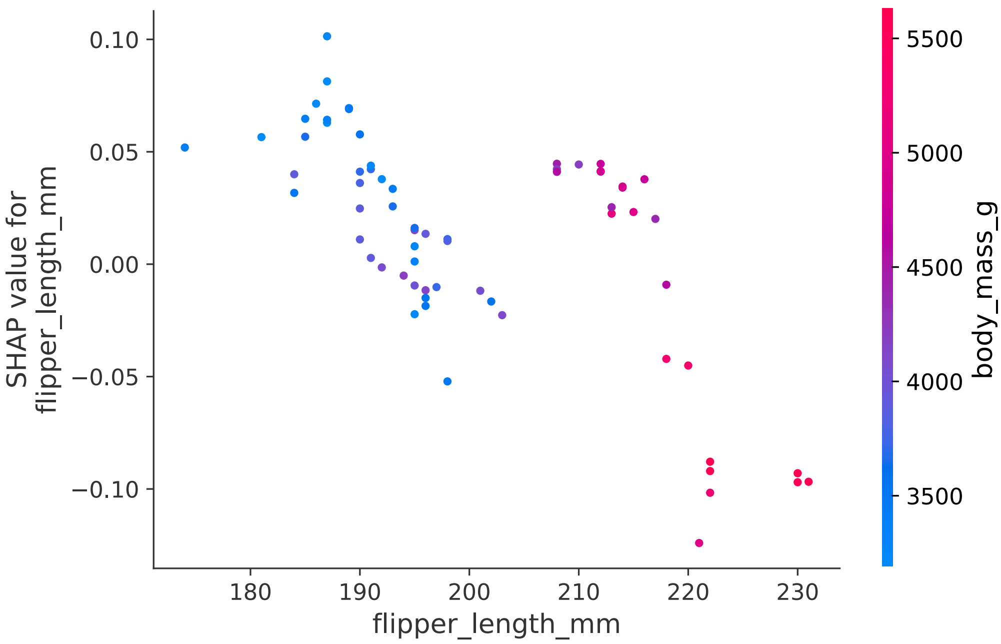

In [ ]:

```
shap.dependence_plot('culmen_length_mm', shap_values[1], X_test)
```



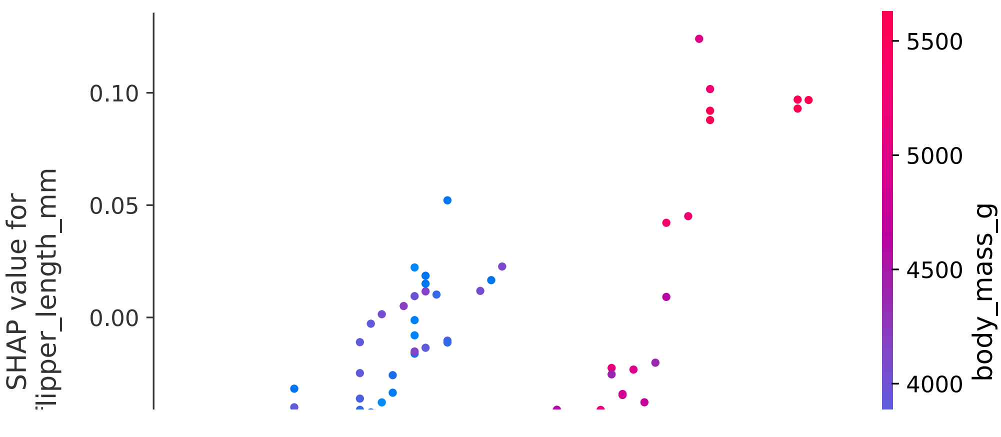

In [ ]:

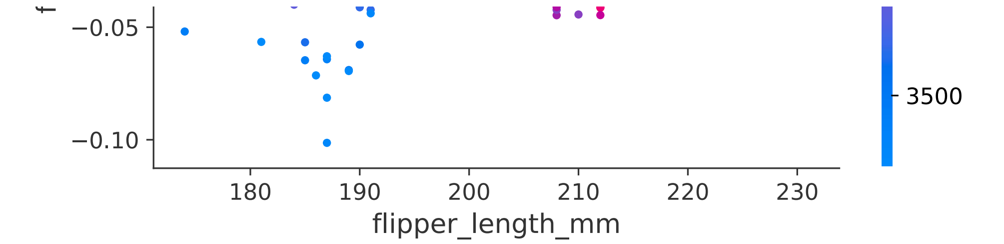
```
shap.dependence_plot('flipper_length_mm', shap_values[0], X_test)
```



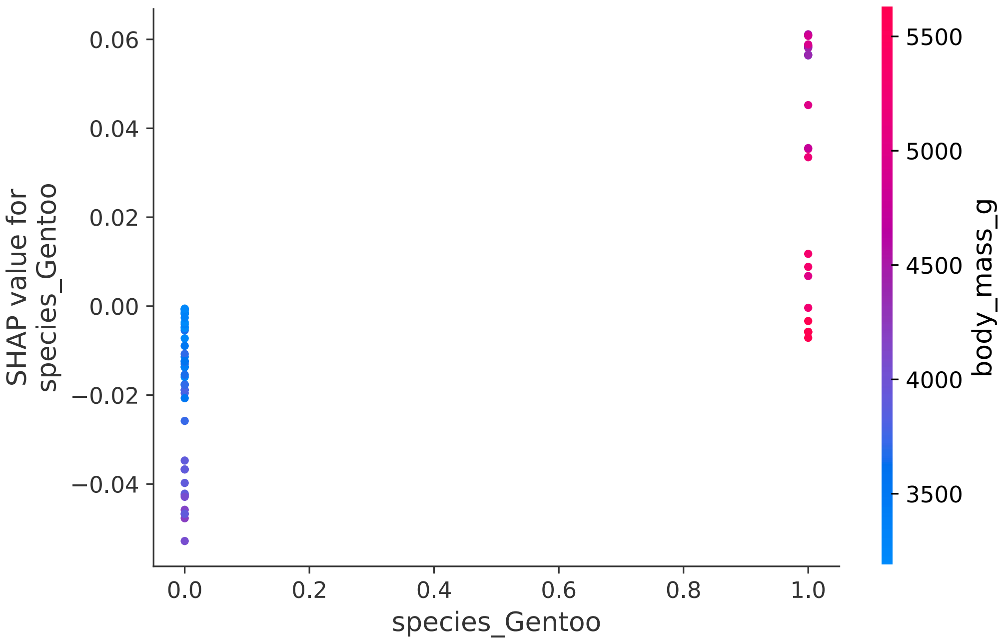

In [ ]:

```
shap.dependence_plot('flipper_length_mm', shap_values[1], X_test)
```

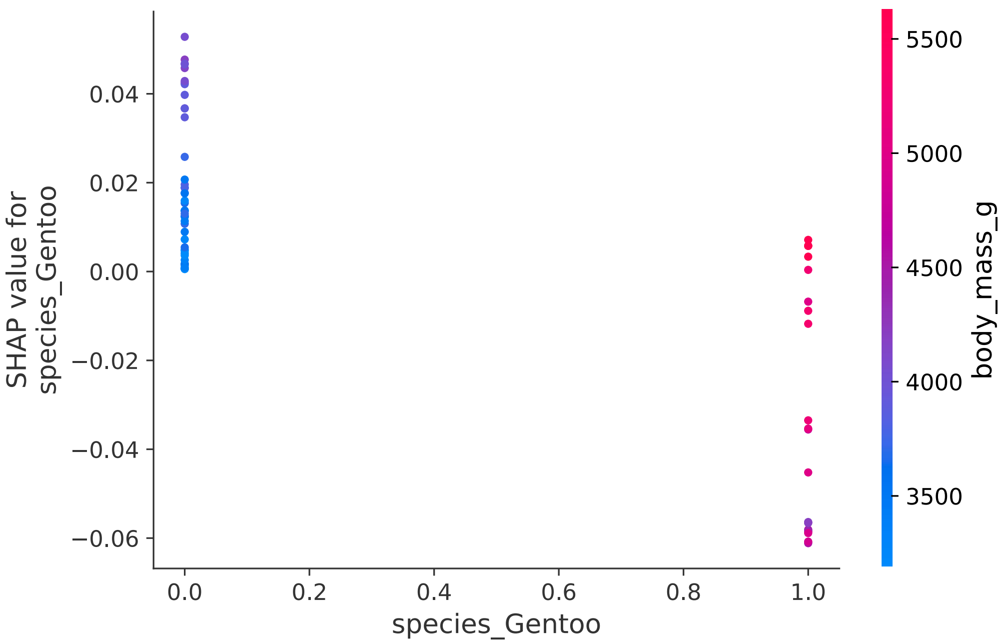




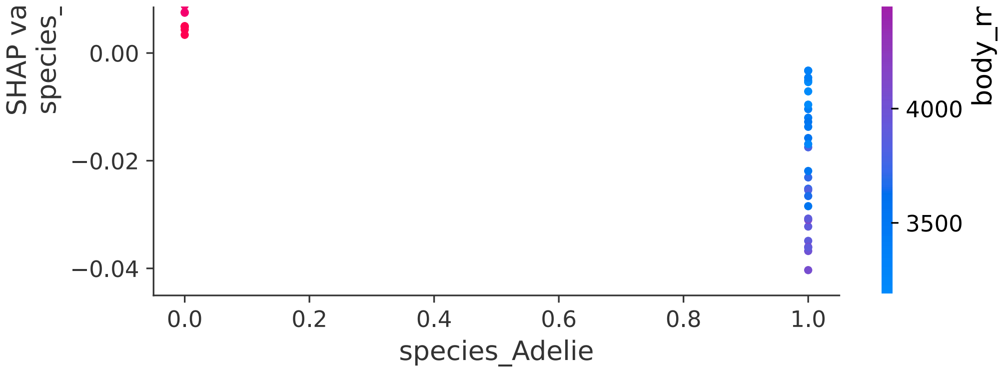

In [ ]:
```
shap.dependence_plot('species_Gentoo', shap_values[0], X_test)
```



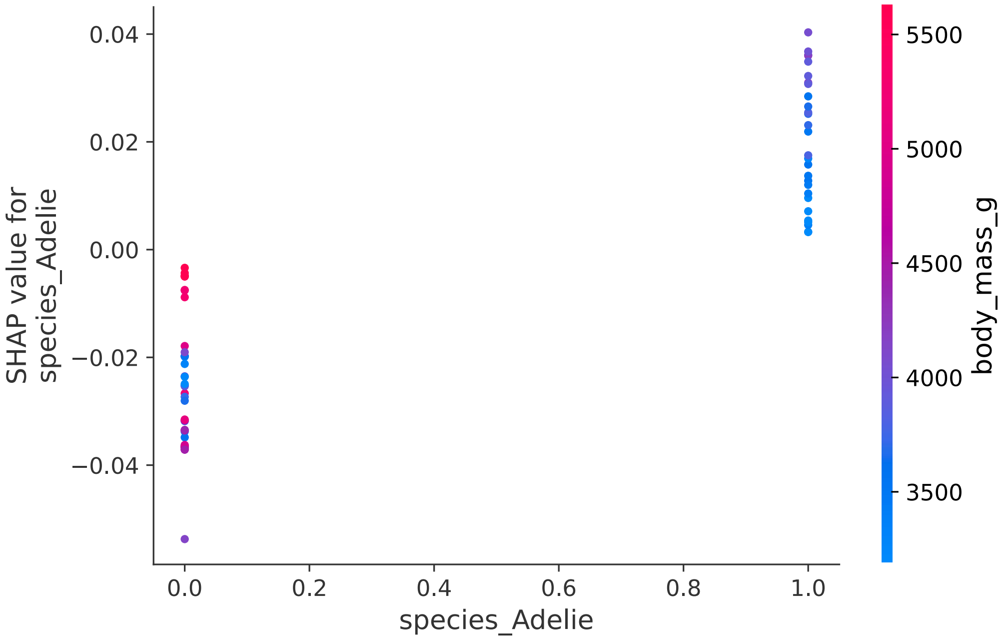

In [ ]:
```
shap.dependence_plot('species_Gentoo', shap_values[1], X_test)
```



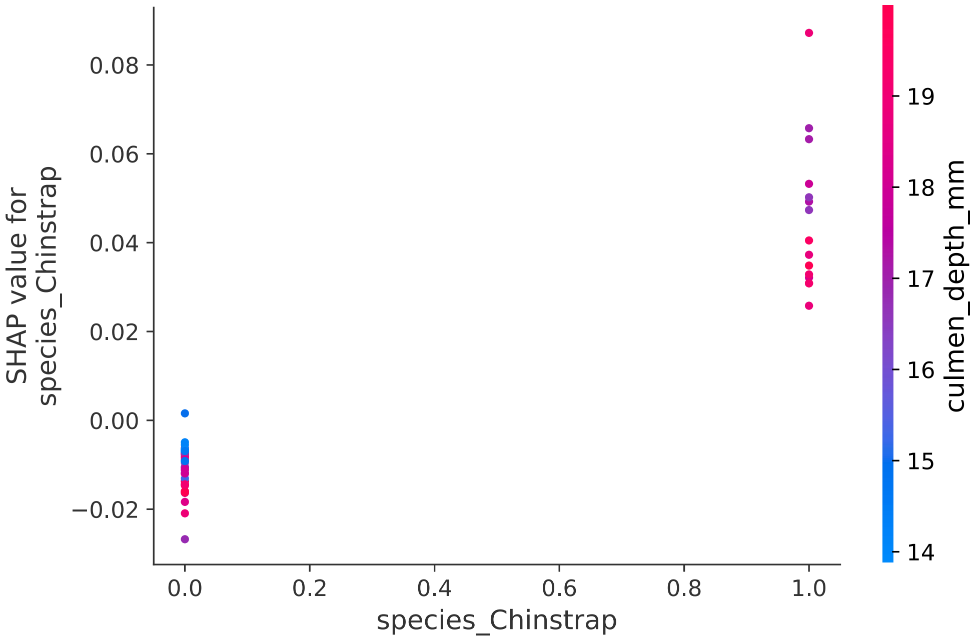

In [ ]:
```
shap.dependence_plot('species_Adelie', shap_values[0], X_test)
```




In [ ]:
```
shap.dependence_plot('species_Adelie', shap_values[1], X_test)
```




In [ ]:
```
shap.dependence_plot('species_Chinstrap', shap_values[0], X_test)
```




In [ ]:
```
shap.initjs()

shap.force_plot(explainer.expected_value[0], shap_values[0], X_test.iloc[:, :])
```

Out[ ]:

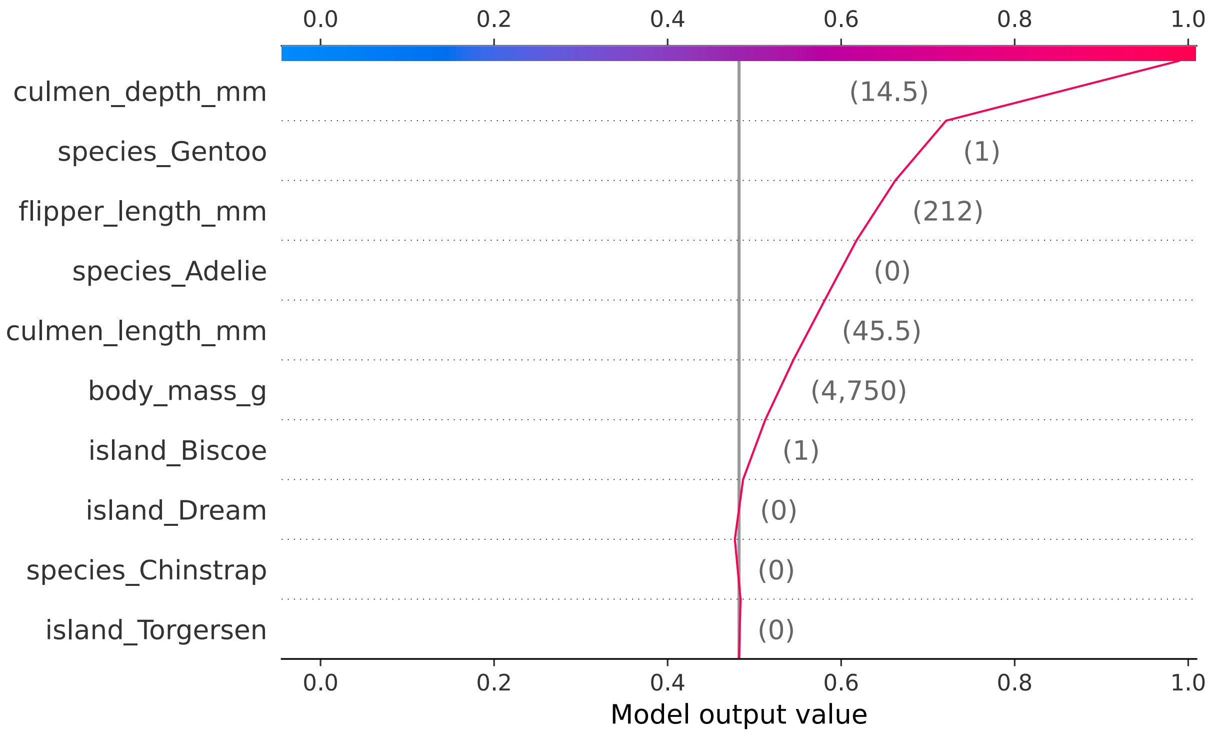
In [ ]:
```
shap.force_plot(explainer.expected_value[1], shap_values[1], X_test.iloc[:, :])
```

Out[ ]:

In [ ]:
```
shap_values_test0 = explainer.shap_values(X_test.iloc[0])
shap.force_plot(explainer.expected_value[0], shap_values_test0[0], X_test.iloc[0])
```

Out[ ]:

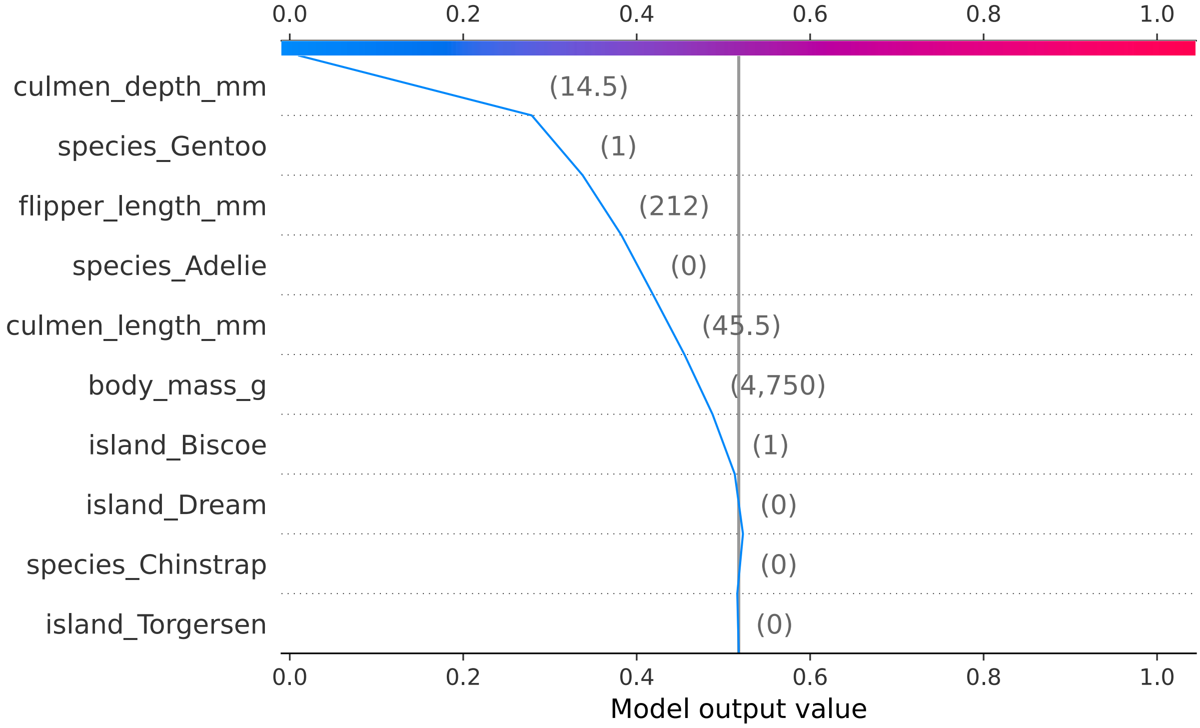
In [ ]:
```
shap.decision_plot(explainer.expected_value[0], shap_values[0][0], X_test.iloc[0, :])
```




In [ ]:
```
shap.force_plot(explainer.expected_value[1], shap_values_test0[1], X_test.iloc[0])
```

Out[ ]:

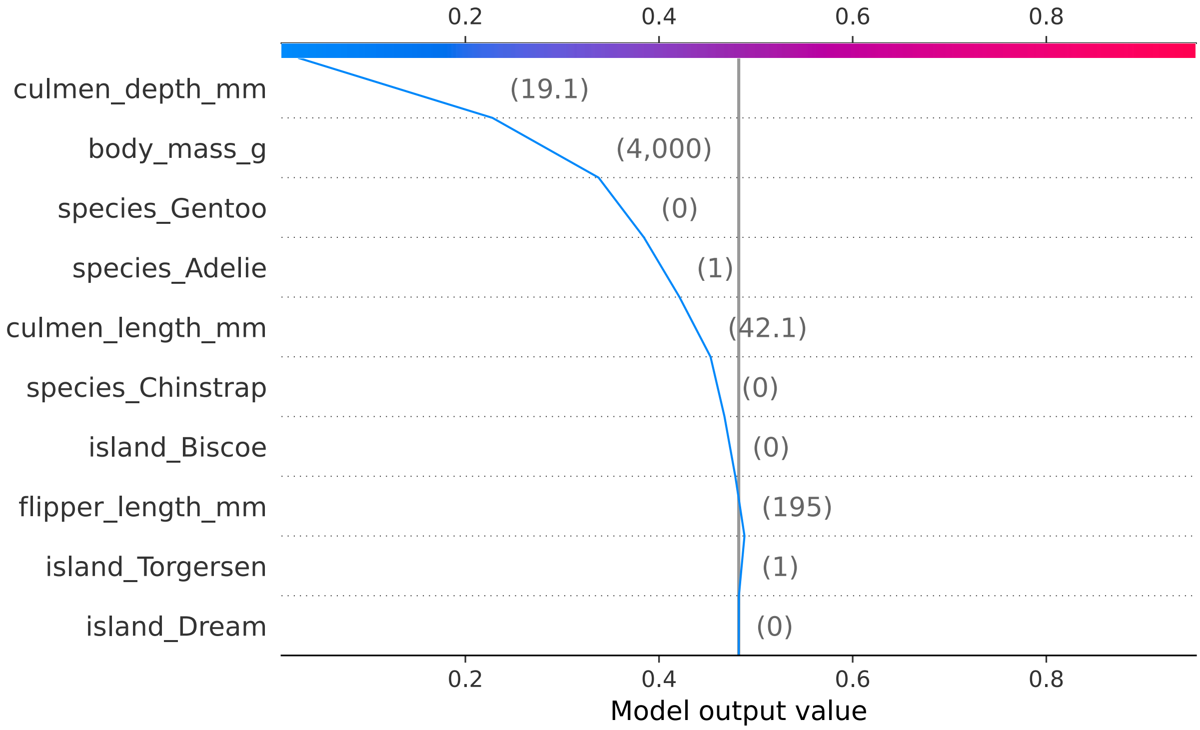
In [ ]:
```
shap.decision_plot(explainer.expected_value[1], shap_values[1][0], X_test.iloc[0, :])
```




In [ ]:
```
shap_values_test3 = explainer.shap_values(X_test.iloc[3])
shap.force_plot(explainer.expected_value[0], shap_values_test3[0], X_test.iloc[3])
```

Out[ ]:

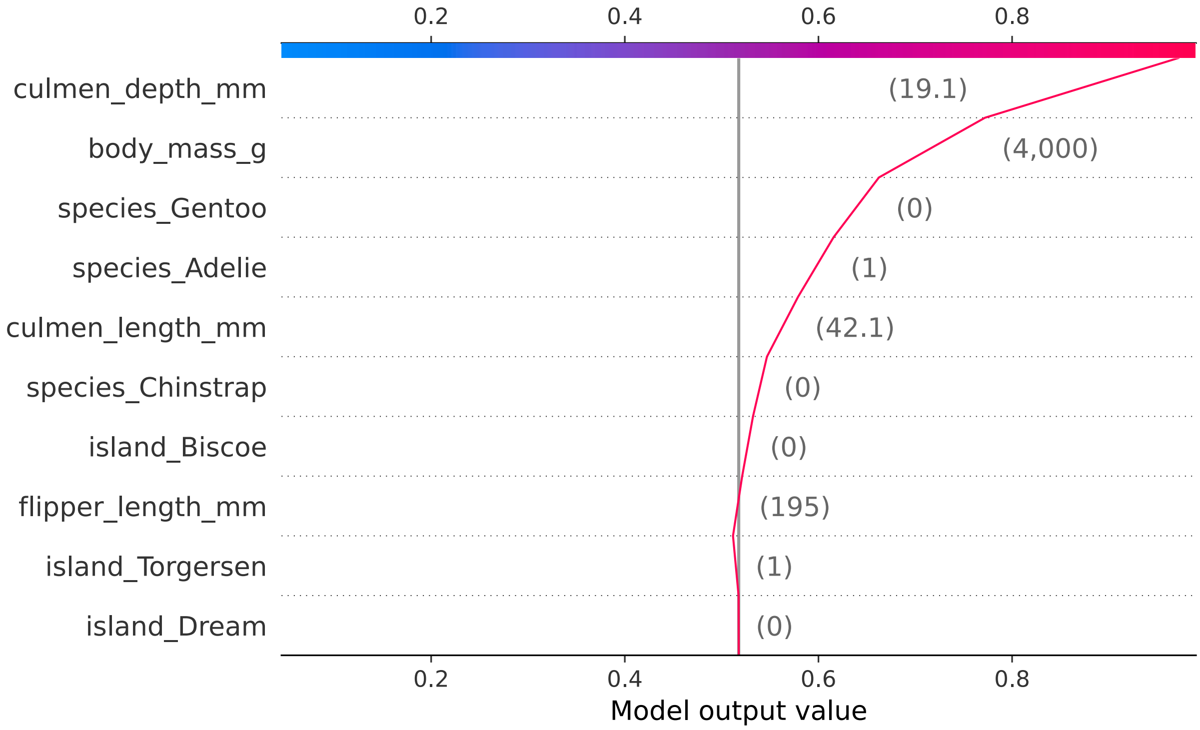
In [ ]:
```
shap.decision_plot(explainer.expected_value[0], shap_values[0][3], X_test.iloc[3, :])
```




In [ ]:
```
shap.force_plot(explainer.expected_value[1], shap_values_test3[1], X_test.iloc[3])
```

Out[ ]:

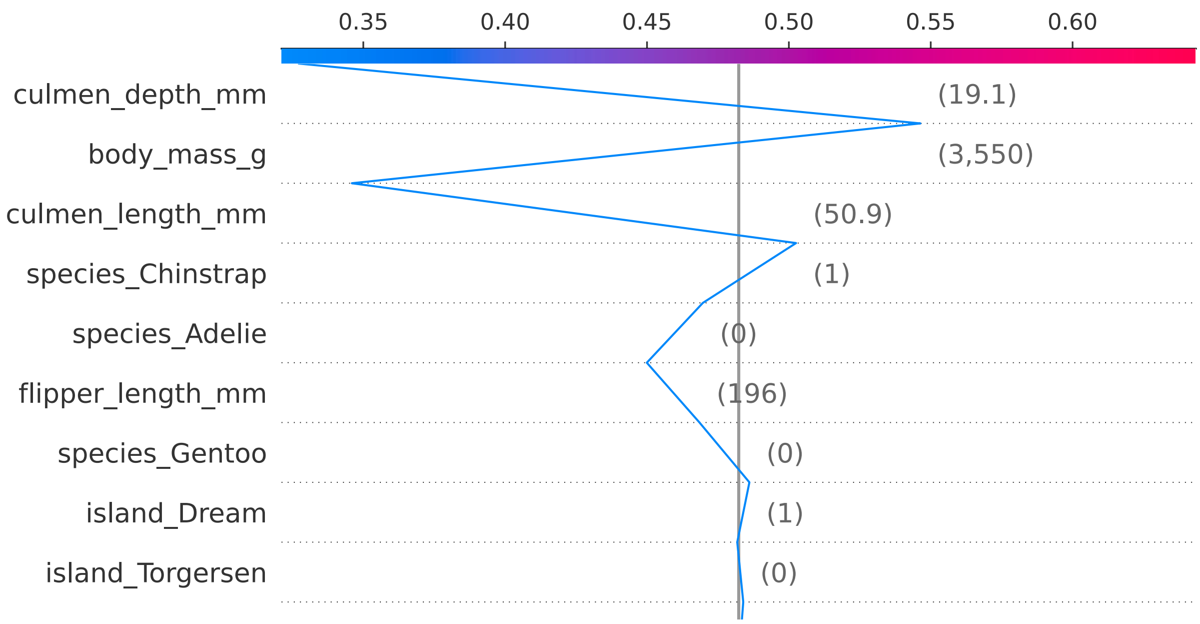
In [ ]:
```
shap.decision_plot(explainer.expected_value[1], shap_values[1][3], X_test.iloc[3, :])
```

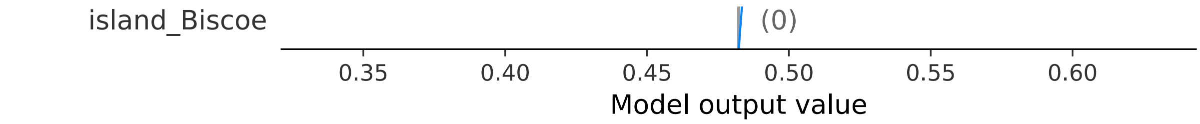



In [ ]:
```
shap_values_test12 = explainer.shap_values(X_test.iloc[12])
shap.force_plot(explainer.expected_value[0], shap_values_test12[0], X_test.iloc[12])
```

Out[ ]:

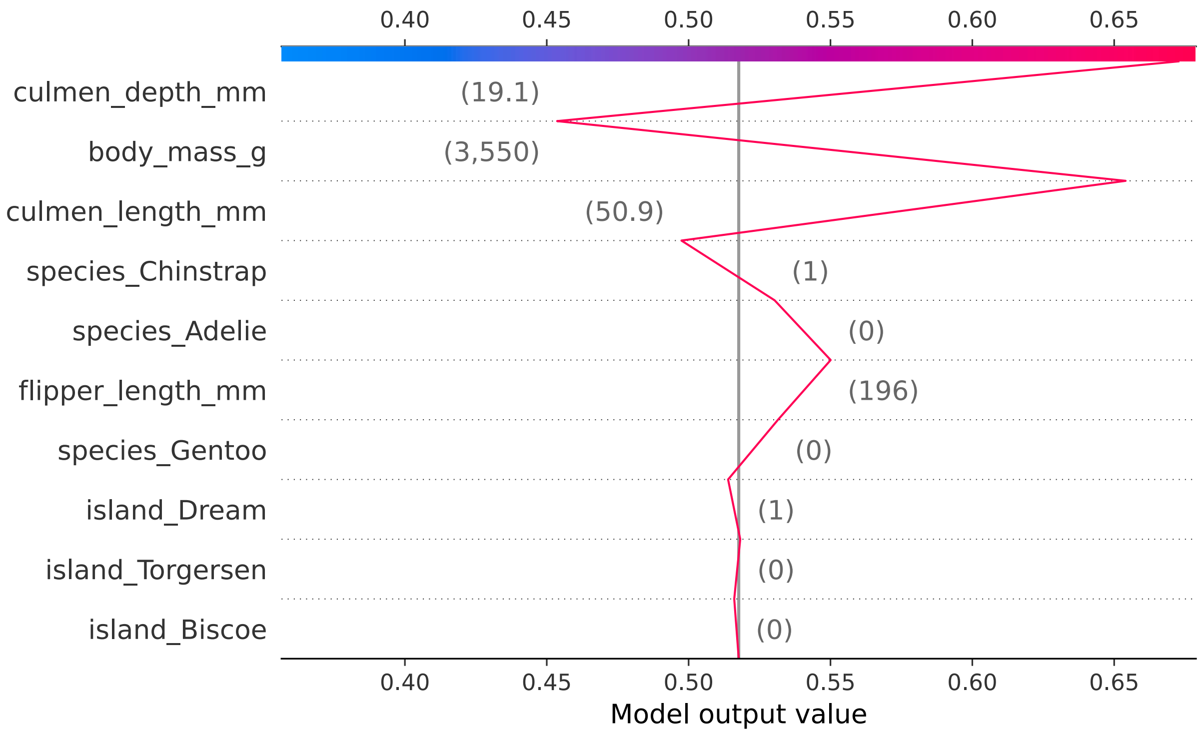
In [ ]:
```
shap.decision_plot(explainer.expected_value[0], shap_values[0][12], X_test.iloc[12, :])
```






In [ ]:
```
shap.force_plot(explainer.expected_value[1], shap_values_test12[1], X_test.iloc[12])
```
Out[ ]:

In [ ]:
```
shap.decision_plot(explainer.expected_value[1], shap_values[1][12], X_test.iloc[12, :])
```




In [ ]: