```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df= pd.read_csv('/content/penguins_size.csv')

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 344,\n  \"fields\": [\
n    {\n      \"column\": \"species\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Adelie\",\n          \"Chinstrap\",\n
\"Gentoo\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"island\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Torgersen\",\n          \"Biscoe\",\n
\"Dream\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"culmen_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5.4595837139265315,\n        \"min\":
32.1,\n        \"max\": 59.6,\n        \"num_unique_values\": 164,\n
\"samples\": [\n          48.2,\n          49.8,\n          45.1\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"culmen_depth_mm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.9747931568167816,\n        \"min\": 13.1,\n        \"max\": 21.5,\n
\"num_unique_values\": 80,\n        \"samples\": [\n          16.9,\n
18.7,\n          18.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flipper_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.061713679356888,\n        \"min\":
172.0,\n        \"max\": 231.0,\n        \"num_unique_values\": 55,\n
\"samples\": [\n          201.0,\n          180.0,\n          212.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"body_mass_g\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
801.9545356980956,\n        \"min\": 2700.0,\n        \"max\":
6300.0,\n        \"num_unique_values\": 94,\n        \"samples\": [\n
4350.0,\n          4150.0,\n          3525.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"MALE\",\n          \"FEMALE\",\n
\".\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df.shape
```

```
(344, 7)
```

```
df.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
```

```
print(df.isnull().sum())
```

```
species               0
island                0
culmen_length_mm      2
culmen_depth_mm       2
flipper_length_mm     2
body_mass_g           2
sex                  10
dtype: int64
```

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 344,\n  \"fields\": [\n    {\n      \"column\": \"species\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"Adelie\",\n          \"Chinstrap\",\n          \"Gentoo\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"island\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"Torgersen\",\n          \"Biscoe\",\n          \"Dream\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"culmen_length_mm\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 5.4595837139265315,\n        \"min\": 32.1,\n        \"max\": 59.6,\n        \"num_unique_values\": 164,\n        \"samples\": [\n          48.2,\n          49.8,\n          45.1\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"culmen_depth_mm\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1.9747931568167816,\n        \"min\": 13.1,\n        \"max\": 21.5,\n
```

\"num_unique_values\": 80,\n   \"samples\": [\n   16.9,\n
18.7,\n   18.6\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"flipper_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 14.061713679356888,\n   \"min\":
172.0,\n   \"max\": 231.0,\n   \"num_unique_values\": 55,\n
\"samples\": [\n   201.0,\n   180.0,\n   212.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
801.9545356980956,\n   \"min\": 2700.0,\n   \"max\":
6300.0,\n   \"num_unique_values\": 94,\n   \"samples\": [\n
4350.0,\n   4150.0,\n   3525.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"sex\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"MALE\",\n   \"FEMALE\",\n
\".\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df=df.fillna(df.mode().iloc[0])
df.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   344 non-null    float64
 3   culmen_depth_mm    344 non-null    float64
 4   flipper_length_mm  344 non-null    float64
 5   body_mass_g        344 non-null    float64
 6   sex                344 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 344,\n \"fields\": [\
n   {\n   \"column\": \"species\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"Adelie\",\n   \"Chinstrap\",\n
\"Gentoo\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"island\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   \"Torgersen\",\n   \"Biscoe\",\n

\"Dream\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"culmen_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 5.4478817414519325,\n   \"min\":
32.1,\n   \"max\": 59.6,\n   \"num_unique_values\": 164,\n
\"samples\": [\n   48.2,\n   49.8,\n   45.1\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"culmen_depth_mm\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.9690609643759762,\n   \"min\": 13.1,\n   \"max\": 21.5,\n
\"num_unique_values\": 80,\n   \"samples\": [\n   16.9,\n
18.7,\n   18.9\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"flipper_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 14.045266153481164,\n   \"min\":
172.0,\n   \"max\": 231.0,\n   \"num_unique_values\": 55,\n
\"samples\": [\n   201.0,\n   180.0,\n   212.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
800.1979232587096,\n   \"min\": 2700.0,\n   \"max\":
6300.0,\n   \"num_unique_values\": 94,\n   \"samples\": [\n
4350.0,\n   4150.0,\n   3525.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"sex\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"MALE\",\n   \"FEMALE\",\n
\".\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"culmen_length_mm\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 108.97611977833193,\n
\"min\": 5.4478817414519325,\n   \"max\": 344.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
43.90552325581396,\n   44.25,\n   344.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"culmen_depth_mm\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
116.45000441627023,\n   \"min\": 1.9690609643759762,\n
\"max\": 344.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   17.15029069767442,\n   17.3,\n
344.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"flipper_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 90.36229363785291,\n   \"min\":
14.045266153481164,\n   \"max\": 344.0,\n

\"num_unique_values\": 8,\n   \"samples\": [\n
200.85174418604652,\n   197.0,\n   344.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1994.1881948154607,\n   \"min\": 344.0,\n   \"max\":
6300.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n
4199.418604651163,\n   4025.0,\n   344.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
df.select_dtypes(exclude='number')
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 344,\n \"fields\": [\
n   {\n   \"column\": \"species\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"Adelie\",\n   \"Chinstrap\",\n
\"Gentoo\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"island\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   \"Torgersen\",\n   \"Biscoe\",\n
\"Dream\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sex\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n
\"MALE\",\n   \"FEMALE\",\n   \".\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
from sklearn import preprocessing
label_encoder=preprocessing.LabelEncoder()
df['species']=label_encoder.fit_transform(df['species'])
df['species'].unique()
df['island']=label_encoder.fit_transform(df['island'])
df['island'].unique()
df['sex']=label_encoder.fit_transform(df['sex'])
df['sex'].unique()
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 344,\n \"fields\": [\
n   {\n   \"column\": \"species\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   0,\n   1,\n   2\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"island\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 2,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   2,\n   0,\n   1\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"culmen_length_mm\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
5.4478817414519325,\n   \"min\": 32.1,\n   \"max\": 59.6,\n
\"num_unique_values\": 164,\n   \"samples\": [\n   48.2,\n
49.8,\n   45.1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"culmen_depth_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 1.9690609643759762,\n   \"min\":
13.1,\n   \"max\": 21.5,\n   \"num_unique_values\": 80,\n
\"samples\": [\n   16.9,\n   18.7,\n   18.9\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"flipper_length_mm\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
14.045266153481164,\n   \"min\": 172.0,\n   \"max\": 231.0,\
n   \"num_unique_values\": 55,\n   \"samples\": [\n
201.0,\n   180.0,\n   212.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
800.1979232587096,\n   \"min\": 2700.0,\n   \"max\":
6300.0,\n   \"num_unique_values\": 94,\n   \"samples\": [\n
4350.0,\n   4150.0,\n   3525.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"sex\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   2,\n   1,\n   0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df_numeric=df.filter(["species","island","culmen_length_mm","culmen_de
pth_mm","flipper_length_mm","body_mass_g"],axis=1)
df_numeric.head()
```

{"summary":"{\n \"name\": \"df_numeric\",\n \"rows\": 344,\n
\"fields\": [\n   {\n   \"column\": \"species\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 2,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   0,\n
1,\n   2\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"island\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   2,\n
0,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"culmen_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 5.4478817414519325,\n   \"min\":
32.1,\n   \"max\": 59.6,\n   \"num_unique_values\": 164,\n

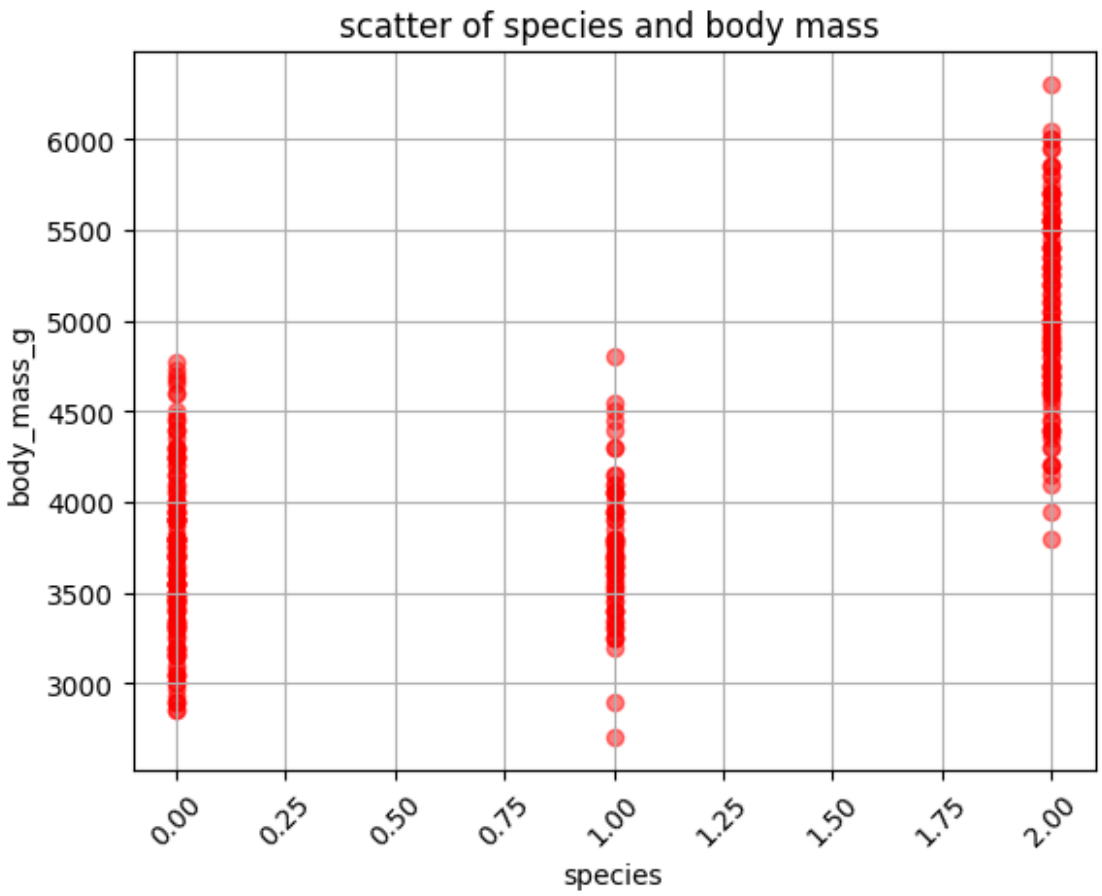
\"samples\": [\n   48.2,\n   49.8,\n   45.1\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"culmen_depth_mm\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.9690609643759762,\n   \"min\": 13.1,\n   \"max\": 21.5,\n
\"num_unique_values\": 80,\n   \"samples\": [\n   16.9,\n
18.7,\n   18.9\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"flipper_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 14.045266153481164,\n   \"min\":
172.0,\n   \"max\": 231.0,\n   \"num_unique_values\": 55,\n
\"samples\": [\n   201.0,\n   180.0,\n   212.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
800.1979232587096,\n   \"min\": 2700.0,\n   \"max\":
6300.0,\n   \"num_unique_values\": 94,\n   \"samples\": [\n
4350.0,\n   4150.0,\n   3525.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df_numeric"}

```
corr_mat=df.corr()
corr_mat
```

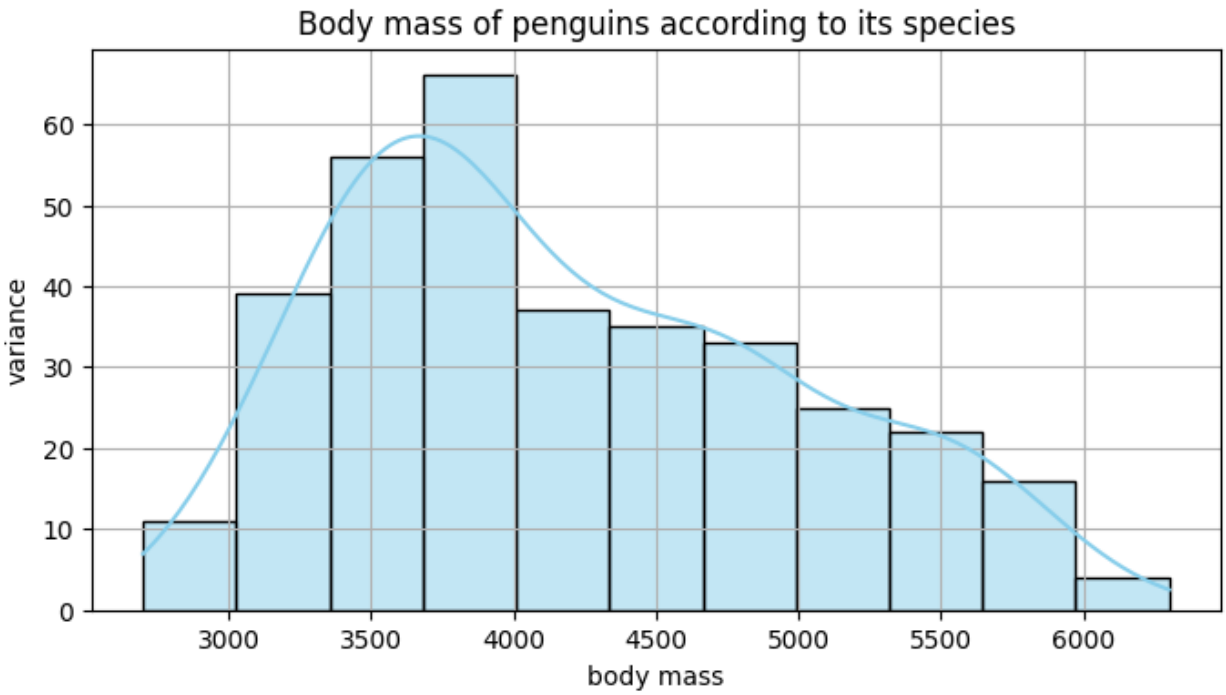
{"summary":"{\n \"name\": \"corr_mat\",\n \"rows\": 7,\n
\"fields\": [\n   {\n   \"column\": \"species\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.7329325973885316,\n   \"min\": -0.7413627636453607,\n
\"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   1.0,\n   -0.6356590214238627,\n
0.746912686249454\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"island\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.6377294432235456,\n   \"min\": -0.6356590214238627,\n
\"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   -0.6356590214238627,\n   1.0,\n   -
0.5589963360317844\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"culmen_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.507781488161546,\n   \"min\": -
0.35259043718023664,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n
0.7278323128854043,\n   -0.35259043718023664,\n
0.5957197979602955\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"culmen_depth_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.6601387114131622,\n   \"min\": -
0.7413627636453607,\n   \"max\": 1.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   -
0.7413627636453607,\n   0.5672882646864881,\n   -

```
0.47133919548221537\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flipper_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.6779578635101543,\n        \"min\": -
0.5824724275889583,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
0.8492555279300408,\n          -0.5634474310232013,\n
0.8713015663447138\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"body_mass_g\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.6340990650559941,\n        \"min\": -
0.5589963360317844,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
0.746912686249454,\n          -0.5589963360317844,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.3358545473976276,\
n        \"min\": -0.0038225730125274323,\n        \"max\": 1.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          -
0.0038225730125274323,\n          0.013369275642612817,\n
0.3905860071863279\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"corr_mat"}
```

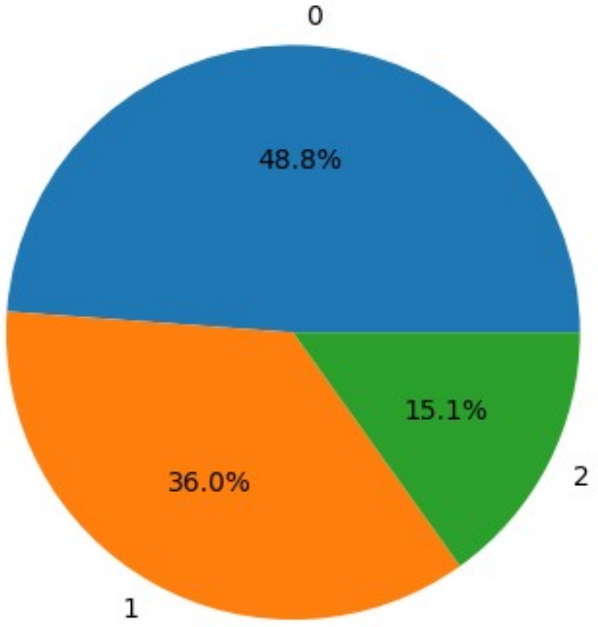

```
plt.scatter(df['species'],df['body_mass_g'],color='red',alpha=0.5)
plt.title('scatter of species and body mass ')
plt.xlabel('species')
plt.ylabel('body_mass_g')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(8,4))
sns.histplot(df['body_mass_g'], kde=True, color='skyblue')
plt.title('Body mass of penguins according to its species')
plt.xlabel('body mass')
plt.ylabel('variance')
plt.grid(True)
plt.show()
```

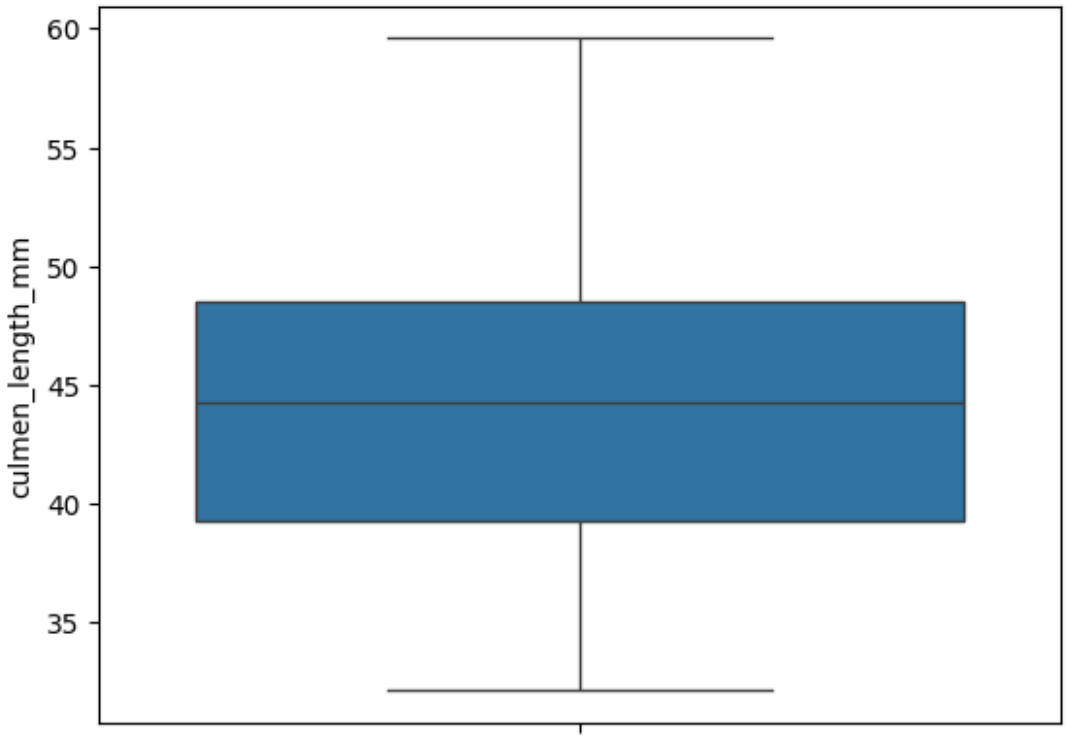



```
x=df['island'].value_counts()
plt.pie(x.values,labels=x.index,autopct='%1.1f%%')
plt.show()
```

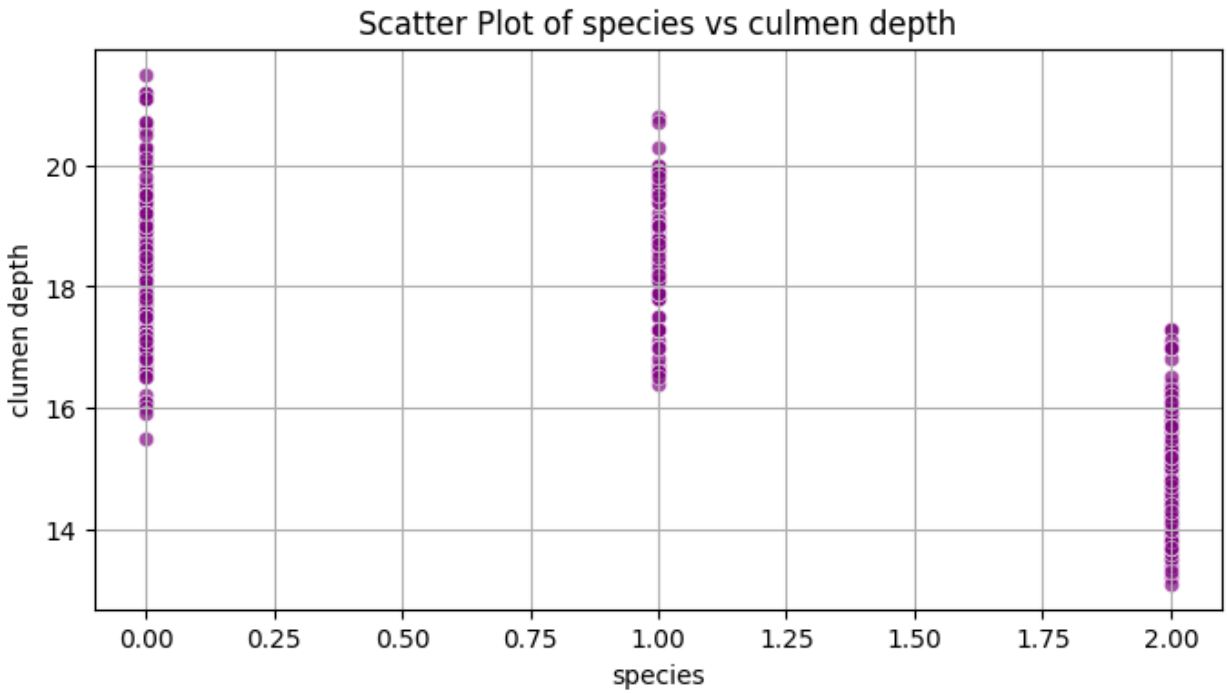


```
sns.boxplot(df.culmen_length_mm)
```

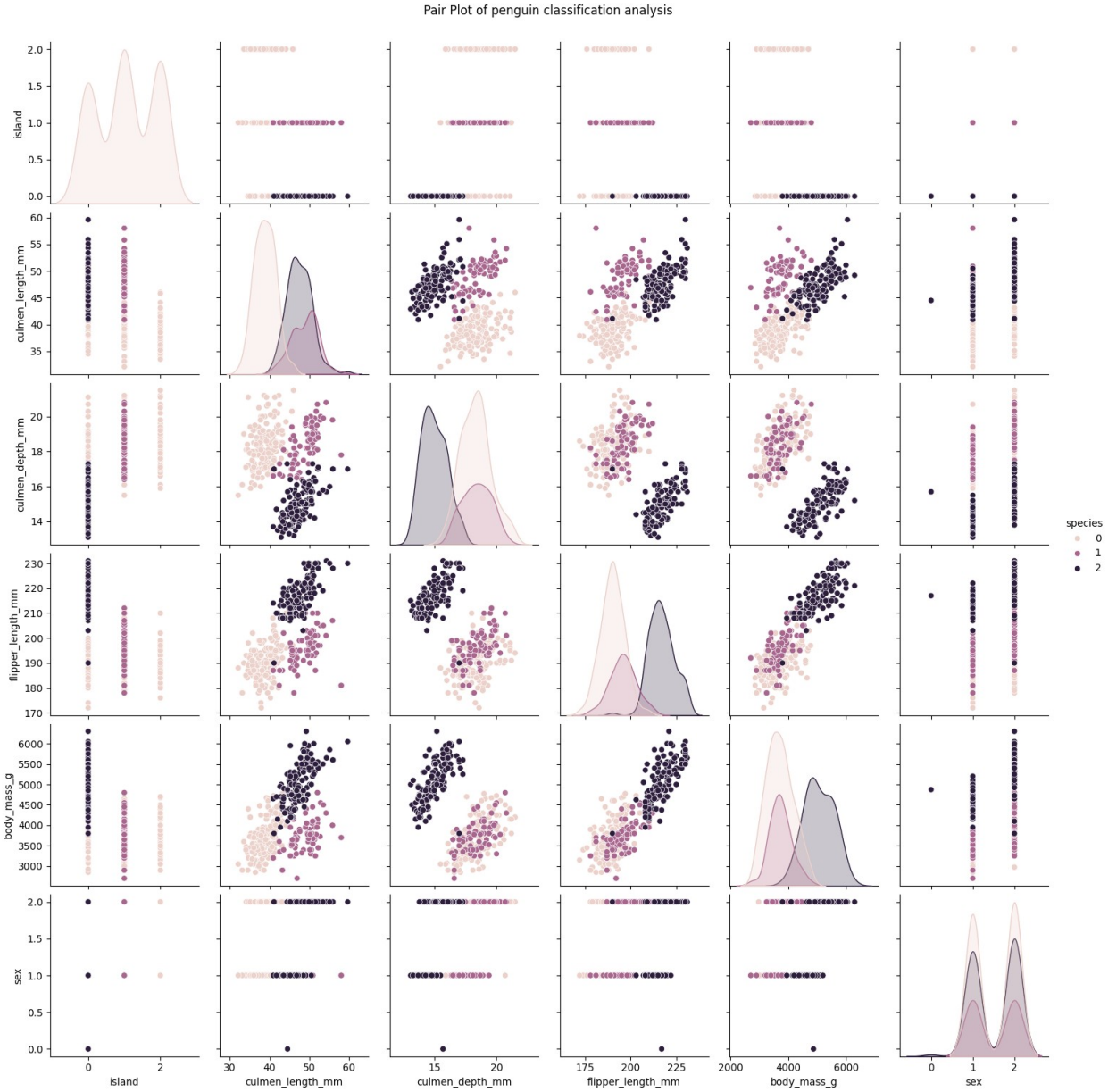
```
<Axes: ylabel='culmen_length_mm'>
```




```
plt.figure(figsize=(8, 4))
sns.scatterplot(x='species', y='culmen_depth_mm', data=df,
color='purple', alpha=0.7)
plt.title('Scatter Plot of species vs culmen depth')
plt.xlabel('species')
plt.ylabel('clumen depth')
plt.grid(True)
plt.show()
```

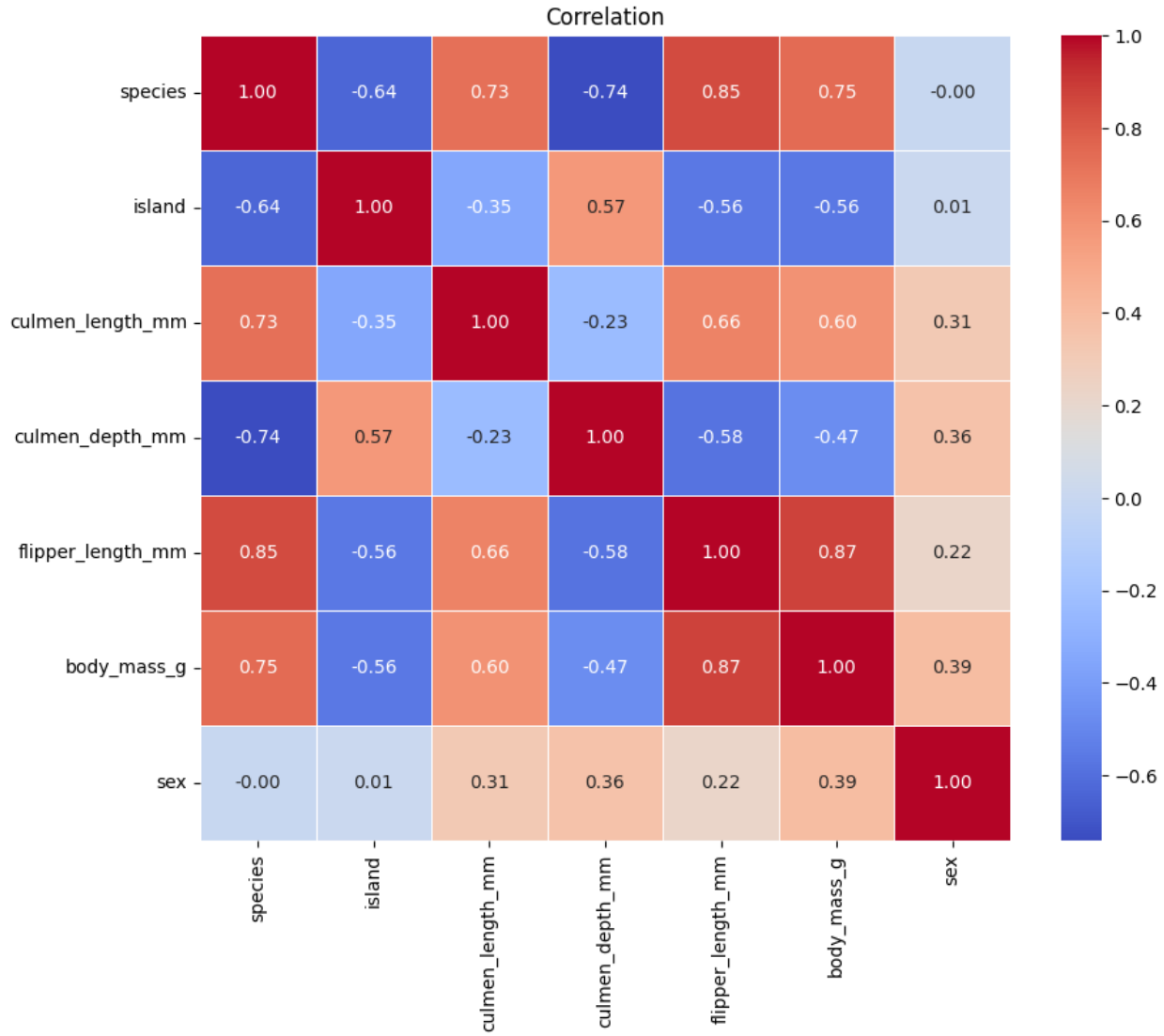



```
sns.pairplot(df, hue='species')
plt.suptitle('Pair Plot of penguin classification analysis', y=1.02)
plt.show()
```




```
corr=df.corr()
plt.rcParams['figure.figsize']=(10,8)
sns.heatmap(corr,cmap='coolwarm',linewidth=0.5,fmt='0.2f',annot=True)
plt.title("Correlation")
```

```
Text(0.5, 1.0, 'Correlation')
```




```
from sklearn.model_selection import train_test_split
X=df.iloc[:,:-1].values #iloc is used for slicing
y=df.iloc[:,-1].values
print(X)
print(y)
```

```
[[0.00e+00 2.00e+00 3.91e+01 1.87e+01 1.81e+02 3.75e+03]
 [0.00e+00 2.00e+00 3.95e+01 1.74e+01 1.86e+02 3.80e+03]
 [0.00e+00 2.00e+00 4.03e+01 1.80e+01 1.95e+02 3.25e+03]
 ...
 [2.00e+00 0.00e+00 5.04e+01 1.57e+01 2.22e+02 5.75e+03]
 [2.00e+00 0.00e+00 4.52e+01 1.48e+01 2.12e+02 5.20e+03]
 [2.00e+00 0.00e+00 4.99e+01 1.61e+01 2.13e+02 5.40e+03]]
[2 1 1 2 1 2 1 2 2 2 2 2 1 2 2 1 1 2 1 2 1 2 1 2 2 1 2 1 1 2 1 2 1 2 1
2 2
 1 1 2 1 2 1 2 1 2 2 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
```

```
1 2
 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
2 1
 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
2 1
 1 2 1 2 1 2 2 1 2 1 1 2 1 2 1 2 1 2 1 2 2 1 1 2 1 2 1 2 1 2 1 2 1
2 1
 2 2 1 1 2 1 2 2 1 2 1 1 2 1 2 2 1 1 2 1 2 1 2 1 2 2 1 2 1 1 2 1 2 2 1
1 2
 1 2 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 1 2 1 2 2 2 1 2 2 1 1 2 1 2 1
2 1
 2 1 2 1 2 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 2 1 2 1 2 2 1
1 2
 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 1 2 1 2 1 2 2 2 1 2 1 2 1
2 1
 2 1 2 0 2 1 2 1 2 1 2]
```

```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
df_scaled=ss.fit_transform(df_numeric)
df_scaled_ds=pd.DataFrame(df_scaled,columns=df_numeric.columns)
df_scaled_ds.head()
df_scaled_ds.describe()
```


```
{"summary":"{\n  \"name\": \"df_scaled_ds\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"species\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
121.5521310324259,\n        \"min\": -1.0298023005653985,\n
\"max\": 344.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          344.0,\n          1.652424966883954e-16,\n
1.2122989107921782\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"island\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 121.52750638660882,\n        \"min\": -0.9140203899996145,\n
\"max\": 344.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          344.0,\n          1.2393187251629654e-16,\n
1.8440762254378191\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"culmen_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 121.54164678901914,\n        \"min\": -
2.170149887528004,\n        \"max\": 344.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          -
1.2393187251629655e-15,\n          0.06332342509961904,\n
344.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"culmen_depth_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 121.56682803053879,\n        \"min\": -
2.0599619248984244,\n        \"max\": 344.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          -
```

```
1.3219399735071631e-15,\n          0.0761415625725022,\n
344.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flipper_length_mm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 121.58287564758902,\n        \"min\": -
2.057189318918569,\n        \"max\": 344.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          -
4.1310624172098847e-16,\n          -0.27463736499416697,\n
344.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"body_mass_g\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 121.5583054740753,\n        \"min\": -
1.8765391806496718,\n        \"max\": 344.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          -
4.544168658930873e-16,\n          -0.2182868376095008,\n
344.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
df_mms=mms.fit_transform(df_numeric)
df_mms_ds=pd.DataFrame(df_mms,columns=df_numeric.columns)
df_mms_ds.head()
```

```
{"summary":"{\n  \"name\": \"df_mms_ds\",\n  \"rows\": 344,\n
\"fields\": [\n    {\n      \"column\": \"species\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4466598831212752,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0.0,\n
0.5,\n          1.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"island\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.36309702111440134,\n        \"min\": 0.0,\n        \"max\":
1.0,\n        \"num_unique_values\": 3,\n        \"samples\": [\n
1.0,\n          0.0,\n          0.5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"culmen_length_mm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.1981047905982521,\n        \"min\": 0.0,\n        \"max\":
0.9999999999999998,\n        \"num_unique_values\": 164,\n
\"samples\": [\n          0.5854545454545454,\n
0.6436363636363636,\n          0.47272727272727266\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"culmen_depth_mm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2344120195685686,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 80,\n        \"samples\": [\n
0.4523809523809521,\n          0.6666666666666665,\n
0.6904761904761902\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"flipper_length_mm\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.238055358533579,\n   \"min\":
0.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 55,\n
\"samples\": [\n   0.4915254237288136,\n
0.13559322033898313,\n   0.6779661016949157\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"body_mass_g\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.2222772009051971,\n   \"min\": 0.0,\n   \"max\": 1.0,\n
\"num_unique_values\": 94,\n   \"samples\": [\n
0.45833333333333326,\n   0.4027777777777777,\n
0.22916666666666663\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_mms_ds"}

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, train_size = 0.80,
random_state = 42)

X_train ,X_test , y_train, y_test=
train_test_split(X,y,random_state=42,test_size=0.25,shuffle=True)
print('X train:')
print(X_train.shape)
print('X test:')
print(X_test.shape)
print('Y train:')
print(y_train.shape)
print('Y test:')
print(y_test.shape)
```

```
X train:
(258, 6)
X test:
(86, 6)
Y train:
(258,)
Y test:
(86,)
```

```
from sklearn.model_selection import GridSearchCV
def grid_search_best_model(model, params, k_fold, X_train, y_train):

grid_search=GridSearchCV(model,params,cv=k_fold).fit(X_train,y_train)
  print("Best params", grid_search.best_params_)
  print("Best estimator", grid_search.best_estimator_)
  print("Best score:", grid_search.best_score_)
  return grid_search.best_estimator_

model_results = {}
def score_model(model, X_train, X_test, y_train, y_test,
```

```
show_plot=True):
  y_pred= model.predict(X_test)
  print(f"Training score: {model,score(X_train,y_train)}")
  print(f"Test score: (r2_score(y_test, y_pred))")
  print("MSE: ", mean_squared_error(y_test, y_pred))
  predictions_comparision= pd.DataFrame({'Actual': y_test.tolist(),
'Predicted': y_pred.tolist()}).sample(25)
  if show_plot == True:
                predictions_comparision.plot(kind="bar",
figsize=(12,8), title="Actual vs predicted values")
  print(predictions_comparision.sample(10))
  return {
      "training_score": model.score(X_train,y_train),
      "test_score_r2": r2_score(y_test, y_pred),
      "test_score_mse": mean_squared_error(y_test, y_pred)
      }
def compare_results():
  for key in model_results:
                print("Regression: ", key)
                print("Trainign score", model_results[key]
["training_score"])
                print("R2 Test score  ", model_results[key]
["test_score_r2"])
                print("MSE Test score", model_results[key]
["test_score_mse"])
                print()

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X,y)

LinearRegression()

y_pred_=lr.predict(X_test)
y_pred=y_pred_.astype(int)
y_pred

array([1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 1,
1,
       1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 1, 2, 1, 2, 2,
1,
       1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1])

print(y_test.dtype)
print(y_pred.dtype)

int64
int64
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix=confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

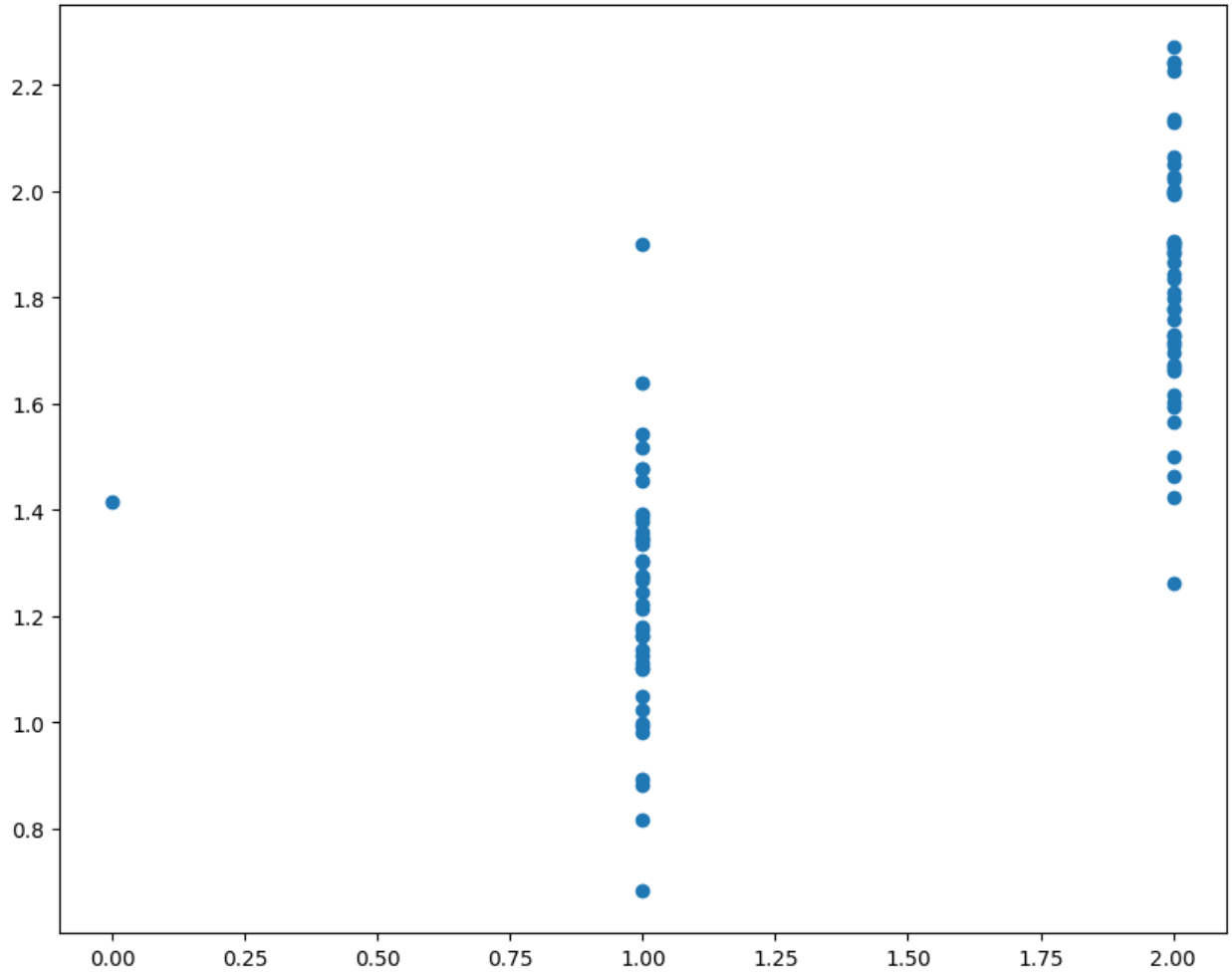
```
[[ 0  1  0]
 [ 7 34  0]
 [ 0 33 11]]
```

```
print("Training Accuracy=",lr.score(X_train,y_train))
print("Test Accuracy=",lr.score(X_test,y_test))
```

```
Training Accuracy= 0.5490165818504891
Test Accuracy= 0.5803489808642865
```

```
plt.scatter(y_test,y_pred_)
```

```
<matplotlib.collections.PathCollection at 0x7bfe4ffbf970>
```



```
from sklearn.tree import DecisionTreeRegressor
DTR = DecisionTreeRegressor()
DTR.fit(X_train, y_train)
```

```
DecisionTreeRegressor()
```

```
y_pred1 = DTR.predict(X_test)
y_pred1
```

```
array([2., 2., 1., 1., 1., 2., 2., 2., 2., 1., 1., 2., 1., 1., 1., 2.,
2.,
       2., 2., 2., 2., 1., 2., 2., 2., 1., 2., 2., 1., 1., 2., 1., 2.,
1.,
       2., 1., 2., 1., 2., 2., 1., 2., 2., 2., 2., 2., 2., 1., 1., 1.,
2.,
       2., 2., 2., 1., 1., 1., 1., 2., 2., 2., 2., 2., 2., 2., 1., 2.,
2.,
       1., 1., 2., 2., 1., 2., 2., 2., 1., 1., 2., 1., 1., 1., 2., 2.,
1.,
       1.])
```

```
from sklearn.metrics import r2_score
DTR_r2score=r2_score(y_test,y_pred1)
print("R-squared:", DTR_r2score)
```

```
R-squared: 0.23404255319148937
```

```
print("Training Accuracy=",DTR.score(X_train,y_train))
print("Test Accuracy=",DTR.score(X_test,y_test))
```

```
Training Accuracy= 1.0
Test Accuracy= 0.23404255319148937
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred1)
print(cm)
```

```
[[ 0  0  1]
 [ 0 31 10]
 [ 0  4 40]]
```

```
from sklearn.ensemble import RandomForestRegressor
RFR = RandomForestRegressor(n_estimators = 20 , random_state = 0 )
RFR.fit(X_train,y_train)
y_pred2_ = RFR.predict(X_test)
y_pred2=y_pred2_.astype(int)
y_pred2
RFR_r2score = r2_score(y_test,y_pred2)
```

```
from sklearn import metrics
print('R_squared: ',RFR_r2score)
```

```
R_squared:  0.1063829787234043
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
print("Training Accuracy= ", RFR.score(X_train, y_train))
print("Test Accuracy= ", RFR.score(X_test,y_test))
cm=confusion_matrix(y_test, y_pred2)
print(cm)
```

```
Training Accuracy=  0.9292347737120847
Test Accuracy=  0.5958510638297874
[[ 0  1  0]
 [ 0 39  2]
 [ 0 18 26]]
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
knn_regressor = KNeighborsRegressor(n_neighbors=15)
knn_regressor.fit(X_train, y_train)
y_pred3_ = knn_regressor.predict(X_test)
y_pred3=y_pred3_.astype(int)
KNN_r2score = r2_score(y_test, y_pred3)
print("R-squared:", KNN_r2score)
print("Training Accuracy=", knn_regressor.score(X_train,y_train))
print("Test Accuracy=", knn_regressor.score(X_test,y_test))
```

```
R-squared: -0.14893617021276606
Training Accuracy= 0.4383983309260151
Test Accuracy= 0.30780141843971653
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred2)
print(cm)
```

```
[[ 0  1  0]
 [ 0 39  2]
 [ 0 18 26]]
```

```
import xgboost as xgb
regressor = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1,
max_depth=3, objective='reg:squarederror', random_state=0)
regressor.fit(X_train, y_train)
y_pred4_ = regressor.predict(X_test)
y_pred4=y_pred4_.astype(int)
XGB_r2score = r2_score(y_test, y_pred4)
print("R-squared:", XGB_r2score)
print("Training Accuracy= ", regressor.score(X_train,y_train))
print("Test Accuracy= ", regressor.score(X_test,y_test))
```

```
R-squared: -0.7021276595744681
Training Accuracy=  0.8988719605332793
Test Accuracy=  0.6432937211139576
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred4)
print(cm)
```

```
[[ 0  1  0]
 [11 29  1]
 [ 1 23 20]]
```

```
compare_results()
```

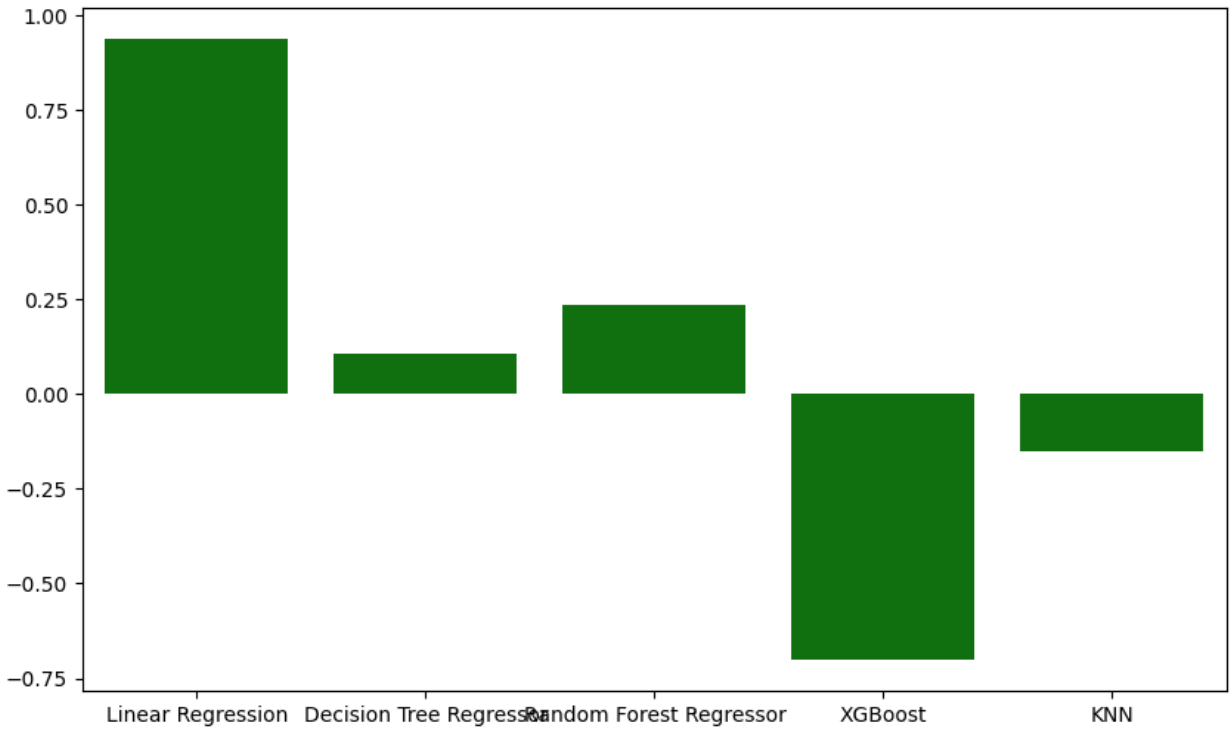
```
import pandas as pd
lr_r2score = 0.9369526600996167  # Replace with the actual value

# Create the accuracy_df DataFrame
accuracy_df = pd.DataFrame({'model': ['Linear Regression', 'Decision
Tree Regressor', 'Random Forest Regressor', 'XGBoost', 'KNN'],
                            'R2_score': [lr_r2score, RFR_r2score,
DTR_r2score, XGB_r2score, KNN_r2score]})

# Display the accuracy_df DataFrame
accuracy_df
```


```
{"summary":"{\n  \"name\": \"accuracy_df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"model\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Decision Tree Regressor\",\n          \"KNN\",\n          \"Random
Forest Regressor\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"R2_score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.5964038036815157,\n        \"min\": -
0.7021276595744681,\n        \"max\": 0.9369526600996168,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.1063829787234043,\n          -0.14893617021276606,\n
0.23404255319148937\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"accuracy_df"}
```


```
model = ['Linear Regression', 'Decision Tree Regressor', 'Random
Forest Regressor', 'XGBoost', 'KNN']
R2_score = [lr_r2score, RFR_r2score, DTR_r2score, XGB_r2score,
KNN_r2score]
plt.figure(figsize=(10,6))
sns.barplot(x=model,y=R2_score, color = 'Green')
plt.show()
```




```
import pickle
pickle.dump(knn_regressor, open("kmodel.pkl","wb"))
print("pickel model downloaded successfully")
```

```
pickel model downloaded successfully
```

```
#cheking with knn model
sample=[[0,2,43.7,15.0,180.8,1892]]
new=ss.transform(sample)
predict_sex=knn_regressor.predict(new)
print(predict_sex)
```

```
[1.06666667]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```