In [2]:
```python
import pandas as pd
import numpy as np
```

In [3]:
```python
df_iris = pd.read_csv("./iris_csv (1).csv")
```

In [4]:
```python
df_iris
```

Out[4]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [5]:
```python
# Drop unnecessart rows
df_iris = df_iris.drop(index = [2, 18])

# Drop rows with na values
df_iris = df_iris.dropna()
```

In [6]:
```python
df_iris.shape
```

Out[6]:
```
(128, 5)
```

In [7]:
```python
df_iris["class"].value_counts()
```

Out[7]:
```
class
Iris-versicolor    50
Iris-virginica     50
Iris-setosa        28
Name: count, dtype: int64
```

Linear regression

In [1]:
```python
import statsmodels.api as sm
```

In [8]:
```python
df_iris.dropna()["sepalwidth"]
```

Out[8]:
```
0      3.5
1      3.0
3      3.1
4      3.6
5      3.9
      ...
145    3.0
146    2.5
147    3.0
148    3.4
149    3.0
Name: sepalwidth, Length: 128, dtype: float64
```

In [9]:
```python
# spector_data.exog = sm.add_constant(spector_data.exog, prepend=False)

mod = sm.OLS(endog = df_iris.dropna()["petalwidth"], exog = sm.add_constant(df_iris

res = mod.fit()

print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             petalwidth   R-squared:                       0.041
Model:                            OLS   Adj. R-squared:                  0.033
Method:                 Least Squares   F-statistic:                     5.372
Date:                Fri, 25 Aug 2023   Prob (F-statistic):             0.0221
Time:                        20:30:54   Log-Likelihood:                -132.08
No. Observations:                 128   AIC:                             268.2
Df Residuals:                     126   BIC:                             273.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
sepalwidth    -0.3588      0.155     -2.318      0.022      -0.665      -0.052
const          2.4370      0.465      5.239      0.000       1.516       3.358
==============================================================================
Omnibus:                       14.949   Durbin-Watson:                   0.422
Prob(Omnibus):                  0.001   Jarque-Bera (JB):                4.846
Skew:                          -0.083   Prob(JB):                       0.0887
Kurtosis:                       2.061   Cond. No.                         25.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

In [10]:
```python
sm.add_constant(df_iris.dropna()["sepalwidth"])
```

Out[10]:

	const	sepalwidth
0	1.0	3.5
1	1.0	3.0
3	1.0	3.1
4	1.0	3.6
5	1.0	3.9
...	...	...
145	1.0	3.0
146	1.0	2.5
147	1.0	3.0
148	1.0	3.4
149	1.0	3.0

128 rows × 2 columns

In [11]:
```
2.55-(0.41*4.2)
```

Out[11]:
```
0.8279999999999998
```

In [12]:
```
sm.add_constant(pd.DataFrame({"c": [1], "a": [4.2]}))
```

Out[12]:

	c	a
0	1	4.2

In [13]:
```
pd.DataFrame({"c": [1], "a": [4.2]})
```

Out[13]:

	c	a
0	1	4.2

In [14]:
```
res.predict(pd.DataFrame({"c": [1], "a": [4.2]}))
```

Out[14]:
```
0    9.876792
dtype: float64
```

kNN

In [ ]:
```
!pip install scikit-learn
```

In [15]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [16]:
```
model_knn = KNeighborsClassifier(n_neighbors = 5)
```

In [17]:
```
df_iris.iloc[:, :-1]
```

Out[17]:

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

128 rows × 4 columns

In [18]:
```
# Fitting / training the kNN machine learning classifier

model_knn.fit(X = df_iris.iloc[:, :-1], y = df_iris["class"])
```

Out[18]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [19]:
```
# Predicting the class label for a new datapoint

model_knn.predict([[1.2, 1, 3, 4.9]])
```

```
C:\python 3114\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have v
alid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

Out[19]:
```
array(['Iris-versicolor'], dtype=object)
```

In [ ]:

In [ ]:

Train -test split

In [20]:
```
df_iris
```

Out[20]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

128 rows × 5 columns

Regression

In [21]:
```
# Target (dependent variable) : petalwidth
# Independednt variables : sepallength, sepalwidth, petallength
```

Linear train test split

In [22]:
```
df_train = df_iris.iloc[:int(128*0.8), :]
df_test = df_iris.iloc[int(128*0.8):, :]
```

In [23]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(102, 5)
(26, 5)
```

Random train test split

In [24]:
```
np.random.choice(range(128), int(128*0.8), replace = False)
```

```
Out[24]: array([ 34, 110,  55,   7,  57, 112,  78,  65,  40,  87, 106,  95,  71,
          58, 122,  52,  88, 121,  93,  25,  36, 105,   2,   6,  39,  90,
          99, 114,  43,  83,  84,  92,  10,  32,  15,  33,  22,  79,   8,
          81,  11,  61,  16,  80, 103,  62,  54,  66,  45,  41,  98, 127,
          82,  38,  59,  75,  48,  49,  89,  96,  85, 100, 125,   9,  86,
           3,  63,  47,  50,  13,  51,  53,  14,  28,  94,  19, 108,  69,
          64, 104,  26,  12, 109, 118,   0,  29,  44,   4,  31,  68,  72,
          42,  73, 117, 107, 119,  23, 101,   5,  60,  97,  67])
```

In [25]:
```
# Generate random indices for the training data
df_train = df_iris.iloc[np.random.choice(range(128), int(128*0.8), replace = False)
df_train
```

Out[25]:

	sepallength	sepalwidth	petallength	petalwidth	class
93	5.0	2.3	3.3	1.0	Iris-versicolor
68	6.2	2.2	4.5	1.5	Iris-versicolor
115	6.4	3.2	5.3	2.3	Iris-virginica
21	5.1	3.7	1.5	0.4	Iris-setosa
57	4.9	2.4	3.3	1.0	Iris-versicolor
...	...	...	...	...	...
118	7.7	2.6	6.9	2.3	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
49	5.0	3.3	1.4	0.2	Iris-setosa
83	6.0	2.7	5.1	1.6	Iris-versicolor
91	6.1	3.0	4.6	1.4	Iris-versicolor

102 rows × 5 columns

In [26]:
```
# Get those indices which has not been used in training
set(df_iris.index) - set(df_train.index)
```

Out[26]:
```
{19,
 25,
 39,
 42,
 46,
 50,
 64,
 82,
 85,
 86,
 87,
 88,
 89,
 92,
 101,
 103,
 111,
 112,
 121,
 127,
 129,
 138,
 140,
 147,
 148,
 149}
```

In [27]:
```
# Use the left out indices for test data
df_test = df_iris.loc[list(set(df_iris.index) - set(df_train.index)), :]
df_test
```

Out[27]:

	sepallength	sepalwidth	petallength	petalwidth	class
129	7.2	3.0	5.8	1.6	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
19	5.1	3.8	1.5	0.3	Iris-setosa
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica
25	5.0	3.0	1.6	0.2	Iris-setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor
64	5.6	2.9	3.6	1.3	Iris-versicolor
82	5.8	2.7	3.9	1.2	Iris-versicolor
85	6.0	3.4	4.5	1.6	Iris-versicolor
86	6.7	3.1	4.7	1.5	Iris-versicolor
87	6.3	2.3	4.4	1.3	Iris-versicolor
88	5.6	3.0	4.1	1.3	Iris-versicolor
89	5.5	2.5	4.0	1.3	Iris-versicolor
92	5.8	2.6	4.0	1.2	Iris-versicolor
101	5.8	2.7	5.1	1.9	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
111	6.4	2.7	5.3	1.9	Iris-virginica
112	6.8	3.0	5.5	2.1	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica

Using train_test_split function from sklearn

In [28]:

```
X = df_iris.loc[:, ["sepallength", "sepalwidth", "petallength"]]
y = df_iris.loc[:, "petalwidth"]
```

In [29]:

```
X
```

Out[29]:

	sepallength	sepalwidth	petallength
0	5.1	3.5	1.4
1	4.9	3.0	1.4
3	4.6	3.1	1.5
4	5.0	3.6	1.4
5	5.4	3.9	1.7
...	...	...	...
145	6.7	3.0	5.2
146	6.3	2.5	5.0
147	6.5	3.0	5.2
148	6.2	3.4	5.4
149	5.9	3.0	5.1

128 rows × 3 columns

In [30]:
```
y
```

Out[30]:
```
0      0.2
1      0.2
3      0.2
4      0.2
5      0.4
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petalwidth, Length: 128, dtype: float64
```

In [31]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_
```

In [32]:
```
X_train
```

Out[32]:

	sepallength	sepalwidth	petallength
92	5.8	2.6	4.0
100	6.3	3.3	6.0
69	5.6	2.5	3.9
0	5.1	3.5	1.4
22	4.6	3.6	1.0
...	...	...	...
128	6.4	2.8	5.6
24	4.8	3.4	1.9
114	5.8	2.8	5.1
73	6.1	2.8	4.7
124	6.7	3.3	5.7

102 rows × 3 columns

In [33]:
```
X_test.shape
```

Out[33]:
```
(26, 3)
```

In [34]:
```
y_train
```

Out[34]:
```
92     1.2
100    2.5
69     1.1
0      0.2
22     0.2
      ...
128    2.1
24     0.2
114    2.4
73     1.2
124    2.1
Name: petalwidth, Length: 102, dtype: float64
```

In [35]:
```
len(y_test)
```

Out[35]:
```
26
```

Model training

In [36]:
```
from sklearn.linear_model import LinearRegression
```

In [37]:
```
# Creating the model object
model_lr = LinearRegression()
```

In [38]:
```
model_lr.fit(X_train, y_train)
```

Out[38]:
```
▾ LinearRegression
LinearRegression()
```

In [39]:
```
model_lr.score(X_train, y_train)
```

Out[39]:
```
0.9104513929181576
```

In [40]:
```
dir(model_lr)
```

```
Out[40]: ['__abstractmethods__',
 '__annotations__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setstate__',
 '__sizeof__',
 '__sklearn_clone__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_abc_impl',
 '_build_request_for_signature',
 '_check_feature_names',
 '_check_n_features',
 '_decision_function',
 '_estimator_type',
 '_get_default_requests',
 '_get_metadata_request',
 '_get_param_names',
 '_get_tags',
 '_more_tags',
 '_parameter_constraints',
 '_repr_html_',
 '_repr_html_inner',
 '_repr_mimebundle_',
 '_set_intercept',
 '_validate_data',
 '_validate_params',
 'coef_',
 'copy_X',
 'feature_names_in_',
 'fit',
 'fit_intercept',
 'get_metadata_routing',
 'get_params',
```

```
 'intercept_',
 'n_features_in_',
 'n_jobs',
 'positive',
 'predict',
 'rank_',
 'score',
 'set_fit_request',
 'set_params',
 'set_score_request',
 'singular_']
```

In [41]:
```
model_lr.feature_names_in_
```

Out[41]:
```
array(['sepallength', 'sepalwidth', 'petallength'], dtype=object)
```

In [42]:
```
model_lr.coef_
```

Out[42]:
```
array([-0.26107413,  0.24827545,  0.54700606])
```

In [43]:
```
model_lr.intercept_
```

Out[43]:
```
-0.09575419984582423
```

In [44]:
```
dict(zip(model_lr.feature_names_in_, model_lr.coef_))
```

Out[44]:
```
{'sepallength': -0.2610741299844725,
 'sepalwidth': 0.24827545186904953,
 'petallength': 0.5470060612404106}
```

In [45]:
```
# Prediction on test data
y_pred_lr1 = model_lr.predict(X_test)
y_pred_lr1
```

Out[45]:
```
array([1.63490579, 1.07203126, 0.01155363, 1.22739587, 2.0152503 ,
       1.51932711, 1.57274784, 1.60638655, 2.04271153, 0.25968118,
       1.96062365, 0.18399263, 1.49563152, 0.39263006, 2.313833  ,
       1.97553833, 1.41738323, 2.02278159, 1.79652183, 1.70719847,
       0.17894711, 0.26849234, 1.98793697, 0.311896  , 1.73684966,
       1.30308442])
```

In [46]:
```
y_test.values
```

Out[46]:
```
array([1.7, 1. , 0.3, 1.3, 2.3, 1.5, 1.4, 1.8, 1.8, 0.2, 1.8, 0.2, 1.5,
       0.4, 2.3, 2.3, 1.3, 2.1, 2. , 1.7, 0.3, 0.2, 2.4, 0.4, 1.8, 1. ])
```

In [47]:
```
y_test.shape
```

Out[47]:
```
(26,)
```

In [49]:
```
np.sqrt(np.sum((y_test.values - y_pred_lr1)**2)/26) # RMSE
```

Out[49]:
```
0.17482930839177857
```

In [50]:
```
np.sum((y_test.values - y_pred_lr1)**2)/26 #MSE
```

Out[50]: 0.030565287072747614

In [51]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test.values, y_pred_lr1)
```

Out[51]: 0.030565287072747614

In [52]:
```
np.round(y_pred_lr1, 1)
```

Out[52]:
```
array([1.6, 1.1, 0. , 1.2, 2. , 1.5, 1.6, 1.6, 2. , 0.3, 2. , 0.2, 1.5,
       0.4, 2.3, 2. , 1.4, 2. , 1.8, 1.7, 0.2, 0.3, 2. , 0.3, 1.7, 1.3])
```

In [53]:
```
TSS = np.sum((y_test.values - np.round(y_test.values.mean(), 1))**2)
ESS = np.sum((np.round(y_pred_lr1, 1) - np.round(y_test.values.mean(), 1))**2)
RSS = np.sum((y_test.values - np.round(y_pred_lr1, 1))**2)
```

In [54]:
```
TSS
```

Out[54]: 14.18

In [55]:
```
ESS + RSS
```

Out[55]: 13.66

In [ ]:
```

```

Classification using kNN

- Feature set (independent variables): 'sepallength', 'sepalwidth', 'petallength', 'petalwidth'
- Target class (dependent variable): 'class'

In [56]:
```
X = df_iris.loc[:, ['sepallength', 'sepalwidth', 'petallength', 'petalwidth']]
y = df_iris.loc[:, 'class']
```

In [57]:
```
X
```

Out[57]:

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

128 rows × 4 columns

In [58]:

```
y
```

Out[58]:

```
0          Iris-setosa
1          Iris-setosa
3          Iris-setosa
4          Iris-setosa
5          Iris-setosa
              ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: class, Length: 128, dtype: object
```

Performing train test split

In [98]:

```
from sklearn.model_selection import train_test_split

X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, train_size = 0.8, ran
```

In [99]:

```
X_train2
```

Out[99]:

	sepallength	sepalwidth	petallength	petalwidth
143	6.8	3.2	5.9	2.3
53	5.5	2.3	4.0	1.3
89	5.5	2.5	4.0	1.3
88	5.6	3.0	4.1	1.3
124	6.7	3.3	5.7	2.1
...	...	...	...	...
78	6.0	2.9	4.5	1.5
76	6.8	2.8	4.8	1.4
139	6.9	3.1	5.4	2.1
114	5.8	2.8	5.1	2.4
148	6.2	3.4	5.4	2.3

102 rows × 4 columns

In [100…

```
X_test2
```

Out[100]:

	sepallength	sepalwidth	petallength	petalwidth
17	5.1	3.5	1.4	0.3
4	5.0	3.6	1.4	0.2
61	5.9	3.0	4.2	1.5
107	7.3	2.9	6.3	1.8
20	5.4	3.4	1.7	0.2
123	6.3	2.7	4.9	1.8
43	5.0	3.5	1.6	0.6
65	6.7	3.1	4.4	1.4
93	5.0	2.3	3.3	1.0
106	4.9	2.5	4.5	1.7
137	6.4	3.1	5.5	1.8
116	6.5	3.0	5.5	1.8
47	4.6	3.2	1.4	0.2
57	4.9	2.4	3.3	1.0
42	4.4	3.2	1.3	0.2
118	7.7	2.6	6.9	2.3
132	6.4	2.8	5.6	2.2
97	6.2	2.9	4.3	1.3
50	7.0	3.2	4.7	1.4
90	5.5	2.6	4.4	1.2
92	5.8	2.6	4.0	1.2
48	5.3	3.7	1.5	0.2
104	6.5	3.0	5.8	2.2
45	4.8	3.0	1.4	0.3
125	7.2	3.2	6.0	1.8
7	5.0	3.4	1.5	0.2

In [115…

```
y_train2
```

Out[115]:
```
143     Iris-virginica
53     Iris-versicolor
89     Iris-versicolor
88     Iris-versicolor
124     Iris-virginica
            ...
78     Iris-versicolor
76     Iris-versicolor
139     Iris-virginica
114     Iris-virginica
148     Iris-virginica
Name: class, Length: 102, dtype: object
```

In [114…
```
y_test2
```

Out[114]:
```
17         Iris-setosa
4          Iris-setosa
61     Iris-versicolor
107     Iris-virginica
20         Iris-setosa
123     Iris-virginica
43         Iris-setosa
65     Iris-versicolor
93     Iris-versicolor
106     Iris-virginica
137     Iris-virginica
116     Iris-virginica
47         Iris-setosa
57     Iris-versicolor
42         Iris-setosa
118     Iris-virginica
132     Iris-virginica
97     Iris-versicolor
50     Iris-versicolor
90     Iris-versicolor
92     Iris-versicolor
48         Iris-setosa
104     Iris-virginica
45         Iris-setosa
125     Iris-virginica
7          Iris-setosa
Name: class, dtype: object
```

In [101…
```
y_test2.shape
```

Out[101]:
```
(26,)
```

Train a kNN classifier using the training data

In [102…
```
from sklearn.neighbors import KNeighborsClassifier
```

In [103…
```
# Initializing the model object
model_knn = KNeighborsClassifier(n_neighbors = 4)
```

In [104…
```python
# Fit the model on training data
model_knn.fit(X_train2, y_train2)
```

Out[104]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

In [123…
```python
# Run predictions on test data
y_pred_knn = model_knn.predict(X_test2)
y_pred_knn
```

Out[123]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

In [124…
```python
# Original targets
y_test2.values
```

Out[124]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

In [127…
```python
incorrects = np.nonzero(model_knn.predict(X_test2) != y_test2)[0]
incorrects
```

Out[127]:
```
array([9], dtype=int64)
```

In [135…
```python
incorrect_indices = np.nonzero(model_knn.predict(X_test2) != y_test2)[0]

print("Length of y_test2:", len(y_test2))
print("Length of incorrect_indices:", len(incorrect_indices))
print("Indices in incorrect_indices:", incorrect_indices)

for index in incorrect_indices:
    if index < len(y_test2):
        original_value = y_test2.iloc[index]  # Using iloc to access by integer ind
        predicted_value = model_knn.predict(X_test2.iloc[index].values.reshape(1, -
        print(f"Original Value: {original_value}, Predicted Value: {predicted_value
    else:
        print(f"Index {index} is out of bounds for y_test2")
```

```
Length of y_test2: 26
Length of incorrect_indices: 1
Indices in incorrect_indices: [9]
Original Value: Iris-virginica, Predicted Value: Iris-versicolor
```

```
C:\python 3114\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have v
alid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

Evaluating the kNN classifier

In [136…
```python
model_knn.score(X_test2, y_test2.values)
```

Out[136]:
```
0.9615384615384616
```

In [147…
```python
# Confusion matrix for our test results
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test2.values, y_pred_knn)
```

Out[147]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 1, 8]], dtype=int64)
```

In [138…
```python
25/26
```

Out[138]:
```
0.9615384615384616
```

In [139…
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test2, y_pred_knn)
```

Out[139]:
```
0.9615384615384616
```

In [144…
```python
from sklearn.metrics import precision_score
precision_score(y_test2, y_pred_knn, average="weighted")
```

Out[144]:
```
0.9658119658119658
```

Classification using Logistic regression

- Feature set (independent variables): 'sepallength', 'sepalwidth', 'petallength', 'petalwidth'
- Target class (dependent variable): 'class'

In [113…
```python
from sklearn.linear_model import LogisticRegression
```

In [176…
```python
from sklearn.model_selection import train_test_split

X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, train_size = 0.8, ran
```

In [303…
```python
# Initializing the model object
model_logistic1 = LogisticRegression(random_state = 12).fit(X_train3,y_train3)
```

In [304…
```python
# Fit the model on training data
model_logistic1.fit(X_train3, y_train3)
```

Out[304]:

```
▾ LogisticRegression
LogisticRegression(random_state=12)
```

In [305…

```
# Run predictions on test data
y_pred_lr2 = model_logistic1.predict(X_test3)
y_pred_lr2
```

Out[305]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

In [306…

```
# Get probability values on test data
model_logistic1.predict_proba(X_test3)
```

Out[306]:

```
array([[9.55403239e-01, 4.45966892e-02, 7.14782839e-08],
       [9.63467651e-01, 3.65323046e-02, 4.47828657e-08],
       [1.89697290e-02, 8.98737115e-01, 8.22931564e-02],
       [1.48171502e-06, 2.18108016e-02, 9.78187717e-01],
       [8.86857701e-01, 1.13142008e-01, 2.90880537e-07],
       [1.06999642e-03, 4.20181301e-01, 5.78748703e-01],
       [9.40671448e-01, 5.93281564e-02, 3.95804966e-07],
       [6.20026320e-03, 9.12660054e-01, 8.11396824e-02],
       [1.40224320e-01, 8.57162866e-01, 2.61281388e-03],
       [1.24958187e-02, 6.20995522e-01, 3.66508660e-01],
       [1.41159992e-04, 1.35692875e-01, 8.64165965e-01],
       [1.16110947e-04, 1.36947749e-01, 8.62936140e-01],
       [9.64239902e-01, 3.57600427e-02, 5.56436090e-08],
       [1.66740364e-01, 8.30700426e-01, 2.55920995e-03],
       [9.75934964e-01, 2.40650057e-02, 3.04402572e-08],
       [1.11054814e-08, 1.11892694e-03, 9.98881062e-01],
       [2.18591285e-05, 3.49853332e-02, 9.64992808e-01],
       [1.07072974e-02, 9.29942834e-01, 5.93498689e-02],
       [2.55652293e-03, 8.45693000e-01, 1.51750477e-01],
       [1.33055040e-02, 9.07123542e-01, 7.95709542e-02],
       [2.25148990e-02, 9.52687865e-01, 2.47972359e-02],
       [9.45765150e-01, 5.42347731e-02, 7.70498752e-08],
       [1.02277379e-05, 2.21428882e-02, 9.77846884e-01],
       [9.46953730e-01, 5.30461636e-02, 1.06780408e-07],
       [9.31431761e-06, 5.05565925e-02, 9.49434093e-01],
       [9.46611609e-01, 5.33883003e-02, 9.07041265e-08]])
```

Evaluating the Logistic regression classifier

In [307…

```
model_logistic1.score(X_test3, y_test3.values)
```

Out[307]:

```
0.9615384615384616
```

In [308…

```
confusion_matrix(y_test3.values, y_pred_lr2)
```

Out[308]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 1, 8]], dtype=int64)
```

In [312…
```
# Initializing the model object
model_logistic2 = LogisticRegression(random_state = 12,solver='liblinear').fit(X_tr
```

In [313…
```
# Fit the model on training data
model_logistic2.fit(X_train3, y_train3)
```

Out[313]:
```
▾ LogisticRegression
LogisticRegression(random_state=12, solver='liblinear')
```

In [314…
```
# Run predictions on test data
y_pred_lr3 = model_logistic2.predict(X_test3)
y_pred_lr3
```

Out[314]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

In [315…
```
# Get probability values on test data
model_logistic2.predict_proba(X_test3)
```

Out[315]:
```
array([[7.91157011e-01, 2.08812269e-01, 3.07195466e-05],
       [7.93944127e-01, 2.06031486e-01, 2.43867568e-05],
       [3.97568242e-02, 6.37740348e-01, 3.22502828e-01],
       [2.28191327e-04, 4.29320145e-01, 5.70451664e-01],
       [7.16118825e-01, 2.83850034e-01, 3.11407430e-05],
       [2.96986768e-03, 4.10284516e-01, 5.86745616e-01],
       [8.34580169e-01, 1.65290863e-01, 1.28968121e-04],
       [4.25757068e-02, 8.47805955e-01, 1.09618338e-01],
       [8.77254436e-02, 7.91997733e-01, 1.20276824e-01],
       [3.15090117e-03, 2.82535775e-01, 7.14313324e-01],
       [1.41443573e-03, 3.42318459e-01, 6.56267105e-01],
       [1.24767063e-03, 3.65982761e-01, 6.32769568e-01],
       [7.48467160e-01, 2.51455239e-01, 7.76008103e-05],
       [1.00964864e-01, 7.70051003e-01, 1.28984133e-01],
       [7.67072192e-01, 2.32841144e-01, 8.66637757e-05],
       [3.11758411e-05, 3.99787882e-01, 6.00180942e-01],
       [4.95300431e-04, 2.70045896e-01, 7.29458803e-01],
       [3.31458945e-02, 7.99774831e-01, 1.67079274e-01],
       [2.76950607e-02, 8.63551784e-01, 1.08753155e-01],
       [1.11143414e-02, 6.14840215e-01, 3.74045444e-01],
       [3.55934778e-02, 7.95358492e-01, 1.69048030e-01],
       [7.87380926e-01, 2.12602719e-01, 1.63540285e-05],
       [4.49519684e-04, 2.55510690e-01, 7.44039790e-01],
       [7.21238090e-01, 2.78668007e-01, 9.39032434e-05],
       [7.28342350e-04, 4.20570451e-01, 5.78701207e-01],
       [7.53648272e-01, 2.46312826e-01, 3.89023051e-05]])
```

In [316…
```
model_logistic2.score(X_test3, y_test3.values)
```

Out[316]:
```
1.0
```

In [317…
```
confusion_matrix(y_test3.values, y_pred_lr3)
```

Out[317]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 0, 9]], dtype=int64)
```

Classification using Support Vector Machine

- Feature set (independent variables): 'sepallength', 'sepalwidth', 'petallength', 'petalwidth'
- Target class (dependent variable): 'class'

In [182…
```
from sklearn.svm import SVC
```

In [183…
```
# Initializing the model object
model_svm = SVC()
```

In [184…
```
# Fit the model on training data
model_svm.fit(X_train2, y_train2)
```

Out[184]:

```
▾ SVC
SVC()
```

In [185…

```python
# Run predictions on test data
y_pred_svc = model_svm.predict(X_test2)
y_pred_svc
```

Out[185]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

Evaluating the SVM classifier

In [186…

```python
model_svm.score(X_test2, y_test2.values)
```

Out[186]:

```
0.9615384615384616
```

In [187…

```python
# Confusion matrix for our test results
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test2.values, y_pred_svc)
```

Out[187]:

```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 1, 8]], dtype=int64)
```

In [193…

```python
# Using linear kernel instead of rbf

# Initializing the model object
model_svm2 = SVC(kernel = "linear")

# Fit the model on training data
model_svm2.fit(X_train2, y_train2)

# Run predictions on test data
y_pred_svc2 = model_svm2.predict(X_test2)
y_pred_svc2
```

Out[193]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

In [194…

```python
model_svm2.score(X_test2, y_test2.values)
```

Out[194]: 1.0

In [195…
```
confusion_matrix(y_test2.values, y_pred_svc2)
```

Out[195]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 0, 9]], dtype=int64)
```

Classification using Decision Tree classifier

- Feature set (independent variables): 'sepallength', 'sepalwidth', 'petallength', 'petalwidth'
- Target class (dependent variable): 'class'

In [196…
```
from sklearn.tree import DecisionTreeClassifier
```

In [246…
```
# Initializing the model object
model_dt = DecisionTreeClassifier(random_state = 10)
```

In [247…
```
# Fit the model on training data
model_dt.fit(X_train2, y_train2)
```

Out[247]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=10)
```

In [248…
```
# Run predictions on test data
y_pred_dt = model_dt.predict(X_test2)
y_pred_dt
```

Out[248]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

Evaluating the Decision Tree classifier

In [249…
```
model_dt.score(X_test2, y_test2.values)
```

Out[249]: 0.9615384615384616

In [250…
```
# Confusion matrix for our test results
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test2.values, y_pred_dt)
```

Out[250]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 1, 8]], dtype=int64)
```

Classification using Random Forest classifier

- Feature set (independent variables): 'sepallength', 'sepalwidth', 'petallength', 'petalwidth'
- Target class (dependent variable): 'class'

In [202...
```
from sklearn.ensemble import RandomForestClassifier
```

In [203...
```
# Initializing the model object
model_rf = RandomForestClassifier(random_state = 10)
```

In [205...
```
# Fit the model on training data
model_rf.fit(X_train2, y_train2)
```

Out[205]:
```
RandomForestClassifier
RandomForestClassifier(random_state=10)
```

In [206...
```
# Run predictions on test data
y_pred_rf = model_rf.predict(X_test2)
y_pred_rf
```

Out[206]:
```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-setosa'], dtype=object)
```

Evaluating the Random Forest classifier

In [207...
```
model_rf.score(X_test2, y_test2.values)
```

Out[207]:
```
0.9615384615384616
```

In [208...
```
model_rf.get_params()
```

Out[208]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 10,
 'verbose': 0,
 'warm_start': False}
```

In [209…
```
confusion_matrix(y_test2.values, y_pred_rf)
```

Out[209]:
```
array([[9, 0, 0],
       [0, 8, 0],
       [0, 1, 8]], dtype=int64)
```

Unsupervised ML

k-means

In [210…
```
from sklearn.cluster import KMeans
```

In [287…
```
# Initializing the model object
model_kmeans = KMeans(n_clusters = 3, random_state=0, n_init='auto')
```

In [288…
```
X_train2
```

Out[288]:

	sepallength	sepalwidth	petallength	petalwidth
143	6.8	3.2	5.9	2.3
53	5.5	2.3	4.0	1.3
89	5.5	2.5	4.0	1.3
88	5.6	3.0	4.1	1.3
124	6.7	3.3	5.7	2.1
...	...	...	...	...
78	6.0	2.9	4.5	1.5
76	6.8	2.8	4.8	1.4
139	6.9	3.1	5.4	2.1
114	5.8	2.8	5.1	2.4
148	6.2	3.4	5.4	2.3

102 rows × 4 columns

In [296…

```
# Fit the model on training data
model_kmeans.fit(X_train2)
```

Out[296]:

```
▾ KMeans
KMeans(n_clusters=3, n_init='auto', random_state=0)
```

In [297…

```
model_kmeans.labels_
```

Out[297]:

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

In [298…

```
model_kmeans.predict(X_test2)
```

Out[298]:

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

Reverse engineering to validate the clustering

In [299…

```
# Creating a 0, 1, 2 mapping dictionary of the actual class labels
y_test_dict = {0: "Iris-versicolor", 1: "Iris-setosa", 2: "Iris-virginica"}
y_test_dict
```

Out[299]:

```
{0: 'Iris-versicolor', 1: 'Iris-setosa', 2: 'Iris-virginica'}
```

In [300…

```
# List of predicted cluster lables mapped to the dictionary
[y_test_dict[i] for i in model_kmeans.predict(X_test2)]
```

Out[300]:
```
['Iris-setosa',
 'Iris-setosa',
 'Iris-versicolor',
 'Iris-virginica',
 'Iris-setosa',
 'Iris-versicolor',
 'Iris-setosa',
 'Iris-versicolor',
 'Iris-versicolor',
 'Iris-versicolor',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-setosa',
 'Iris-versicolor',
 'Iris-setosa',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-versicolor',
 'Iris-versicolor',
 'Iris-versicolor',
 'Iris-versicolor',
 'Iris-setosa',
 'Iris-virginica',
 'Iris-setosa',
 'Iris-virginica',
 'Iris-setosa']
```

In [301…
```
# Actual test labels
y_test2
```

Out[301]:
```
17          Iris-setosa
4           Iris-setosa
61      Iris-versicolor
107      Iris-virginica
20          Iris-setosa
123      Iris-virginica
43          Iris-setosa
65      Iris-versicolor
93      Iris-versicolor
106      Iris-virginica
137      Iris-virginica
116      Iris-virginica
47          Iris-setosa
57      Iris-versicolor
42          Iris-setosa
118      Iris-virginica
132      Iris-virginica
97      Iris-versicolor
50      Iris-versicolor
90      Iris-versicolor
92      Iris-versicolor
48          Iris-setosa
104      Iris-virginica
45          Iris-setosa
125      Iris-virginica
7           Iris-setosa
Name: class, dtype: object
```

In [302…
```python
# Comparing the mapped label names to the actual labels
[y_test_dict[i] for i in model_kmeans.predict(X_test2)] == y_test2
```

```
Out[302]: 17      True
          4       True
          61      True
          107     True
          20      True
          123    False
          43      True
          65      True
          93      True
          106    False
          137     True
          116     True
          47      True
          57      True
          42      True
          118     True
          132     True
          97      True
          50      True
          90      True
          92      True
          48      True
          104     True
          45      True
          125     True
          7       True
          Name: class, dtype: bool
```