Using the given data to predict the species of flower

based on :https://youtu.be/rdaG53khzv0?si=yJE4w1ByxfdmnFv-

```
from sklearn.datasets import load_iris
```

```
iris_data=load_iris()
```

```
print(iris_data.DESCR)
```

```
**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

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

```
df=load_iris()
df
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
```

```
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
```

```
df.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
print(df["DESCR"])
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

```
    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
```

```
df["feature_names"]
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
df["target_names"]
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
sns.set()
```

```
df['data']
```

```
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
```

```
df1=pd.DataFrame(df['data'],columns=df["feature_names"])
df1
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
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

150 rows × 4 columns

```
df1["target"]=df["target"]
```

```
df1.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
df1.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

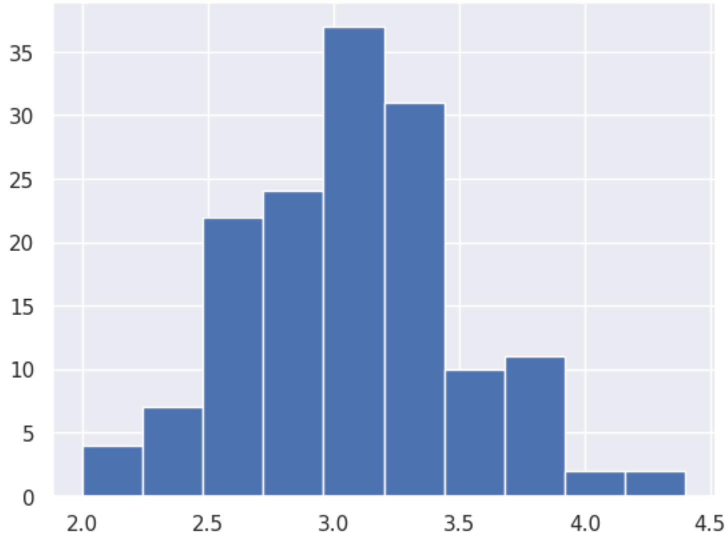
```
df1["target"]
```

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

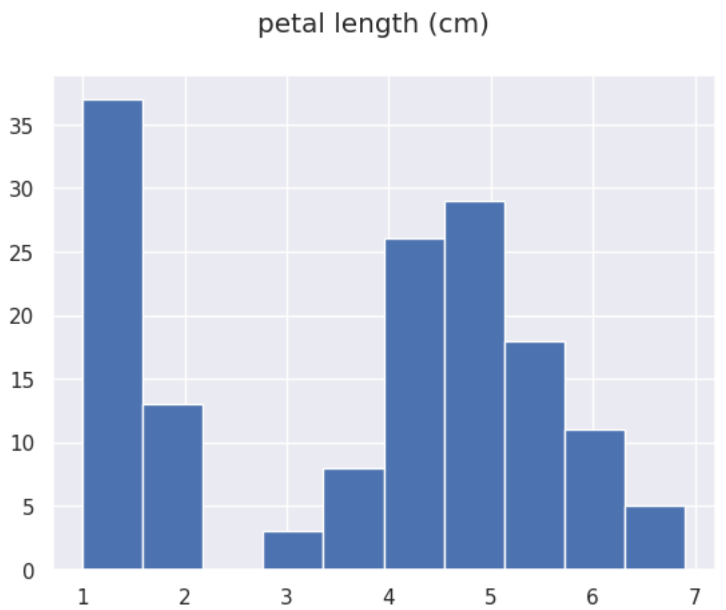
```
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int64
```

```
df1["sepal width (cm)"].hist()
```

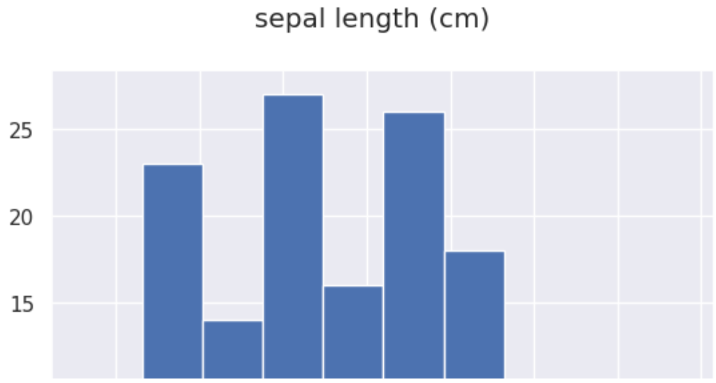
```
<Axes: >
```




```
col="petal length (cm)"
df1[col].hist()
plt.suptitle(col)
plt.show()
```




```
col="sepal length (cm)"
df1[col].hist()
plt.suptitle(col)
plt.show()
```

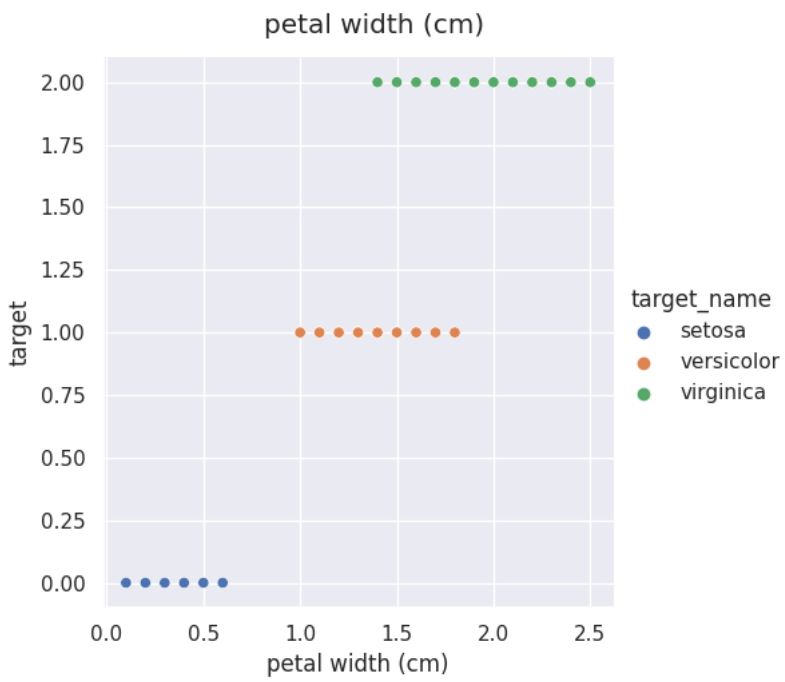



```
df["target_names"]
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
df1["target_name"]=df1["target"].map({0:"setosa",1:"versicolor",2:"virginica"})
```

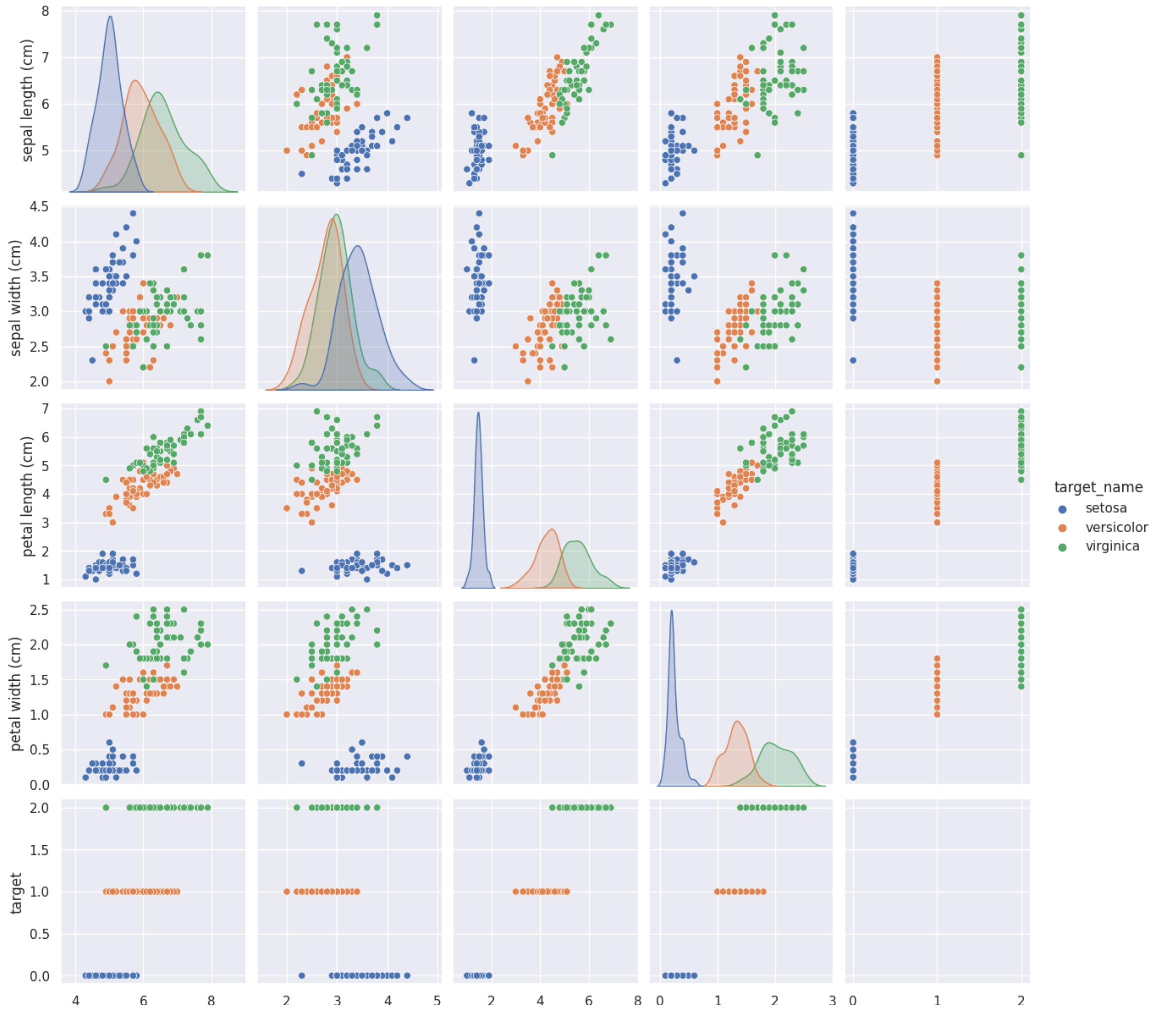
```
col="petal width (cm)"
sns.relplot(x=col,y="target",hue="target_name",data=df1)
plt.suptitle(col,y=1.03)
plt.show()
```



Exploratory Data Analysis

```
sns.pairplot(df1,hue="target_name")
```

```
<seaborn.axisgrid.PairGrid at 0x7fb2cdd27610>
```



From the graphs, concluded that Setosa can be easily identified since they are clustered apart from other two species. Versicolor and virginica overlaps in data sometimes making them hard to differentiate

```
from sklearn.model_selection import train_test_split
```

```
df1_train,df1_test=train_test_split(df1,test_size=0.2)
```

```
df1_train.shape
```

```
(120, 6)
```

```
df1_test.shape
```

```
(30, 6)
```

```
df1_train.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	target_name
120	6.9	3.2	5.7	2.3	2	virginica
18	5.7	3.8	1.7	0.3	0	setosa
55	5.7	2.8	4.5	1.3	1	versicolor
92	5.8	2.6	4.0	1.2	1	versicolor
72	6.3	2.5	4.9	1.5	1	versicolor

```
x_train=df1_train.drop(columns=["target","target_name"]).values
y_train=df1_train["target"].values
```

Manual modelling

```
def single_feature_prediction(petal_length):
  """predicts the iris species given the petal length"""
```

```
  if petal_length<2.5:
    return 0
  elif petal_length<4.8:
    return 1
  else:
    return 2
```

```
df1_train.columns
```

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'target', 'target_name'],
      dtype='object')
```

```
manual_y_predictions=np.array([single_feature_prediction(val) for val in x_train[:, 2]])
```

```
np.mean(manual_y_predictions==y_train)
```

```
0.9583333333333334
```

Manual model accuracy : 94.166%

Modeling using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(max_iter=400,C=10)
```

```
xt,xv,yt,yv=train_test_split(x_train,y_train,test_size=0.25)
```

```
model.fit(xt,yt)
```

```
▾          LogisticRegression
LogisticRegression(C=10, max_iter=400)
```

```
y_pred=model.predict(xv)
```

```
np.mean(y_pred==yv)
```

```
0.9666666666666667
```

```
model.score(xv,yv)
```

```
0.9666666666666667
```

```
from sklearn.model_selection import cross_val_score, cross_val_predict
```

```
np.mean(cross_val_score(model,x_train,y_train,cv=5,scoring="accuracy"))
```

```
0.975
```

```
y_pred=cross_val_predict(model,x_train,y_train,cv=5)
y_pred==y_train
```

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

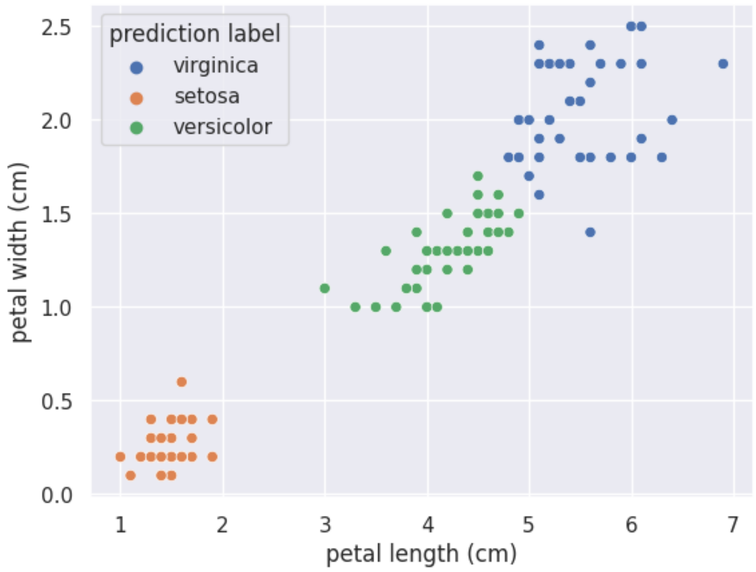
```
pred_correct=y_pred==y_train
```

```
not_pred_correct=~pred_correct

df1_pred=df1_train.copy()

df1_pred["prediction check"]=pred_correct

df1_pred["prediction"]=y_pred

df1_pred["prediction label"]=df1_pred["prediction"].map({0:"setosa",1:"versicolor",2:"virginica"})

df1_pred.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	target_name	prediction check	prediction	prediction label
120	6.9	3.2	5.7	2.3	2	virginica	True	2	virginica
18	5.7	3.8	1.7	0.3	0	setosa	True	0	setosa
55	5.7	2.8	4.5	1.3	1	versicolor	True	1	versicolor
92	5.8	2.6	4.0	1.2	1	versicolor	True	1	versicolor

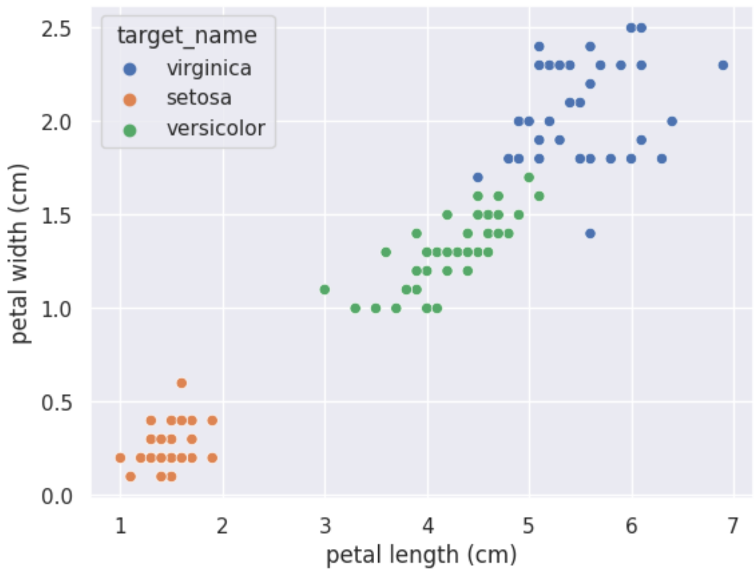
```
sns.scatterplot(x="petal length (cm)",y="petal width (cm)",hue="prediction label",data=df1_pred)
```

<Axes: xlabel='petal length (cm)', ylabel='petal width (cm)'>



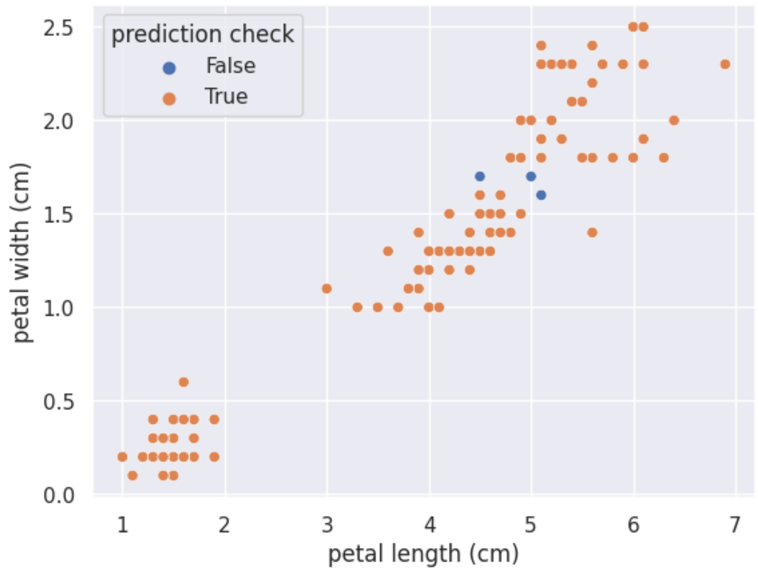

```
sns.scatterplot(x="petal length (cm)",y="petal width (cm)",hue="target_name",data=df1_pred)
```

<Axes: xlabel='petal length (cm)', ylabel='petal width (cm)'>



```
sns.scatterplot(x="petal length (cm)",y="petal width (cm)",hue="prediction check",data=df1_pred)
```

```
<Axes: xlabel='petal length (cm)', ylabel='petal width (cm)'>
```



Conclusion:

- The model achived an accuracy of 96.67 percent
- From the "prediction check" plot, it is visually identified that only 3 of the predictions were false

0s completed at 19:24