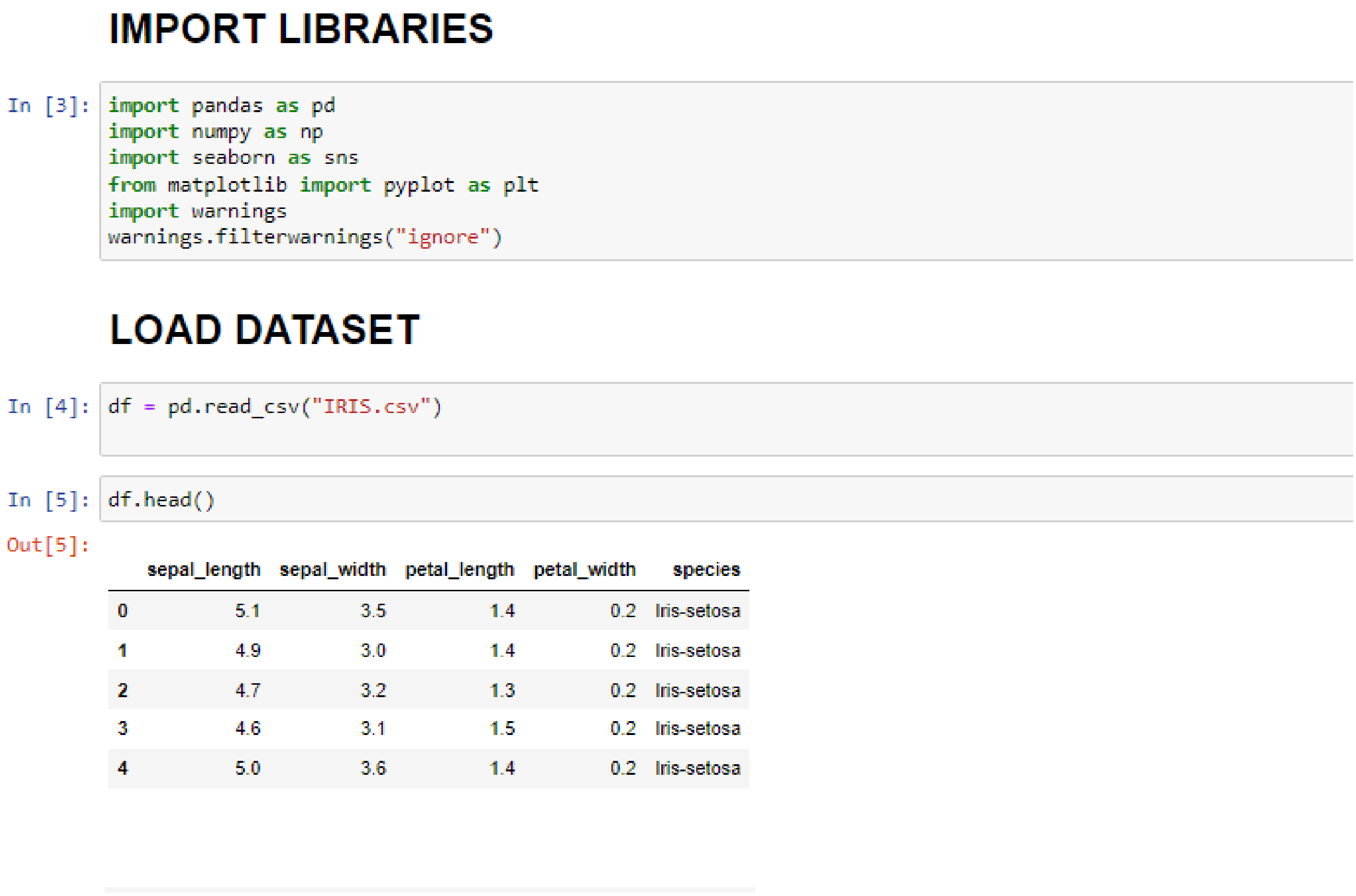
MACHINE LEARNING

I.VINOTH D SILVA

FEB-4-2023



IMPORT LIBRARIES

```
In [3]: import pandas as pd
        import numpy as np
        import seaborn as sns
        from matplotlib import pyplot as plt
        import warnings
        warnings.filterwarnings("ignore")
```

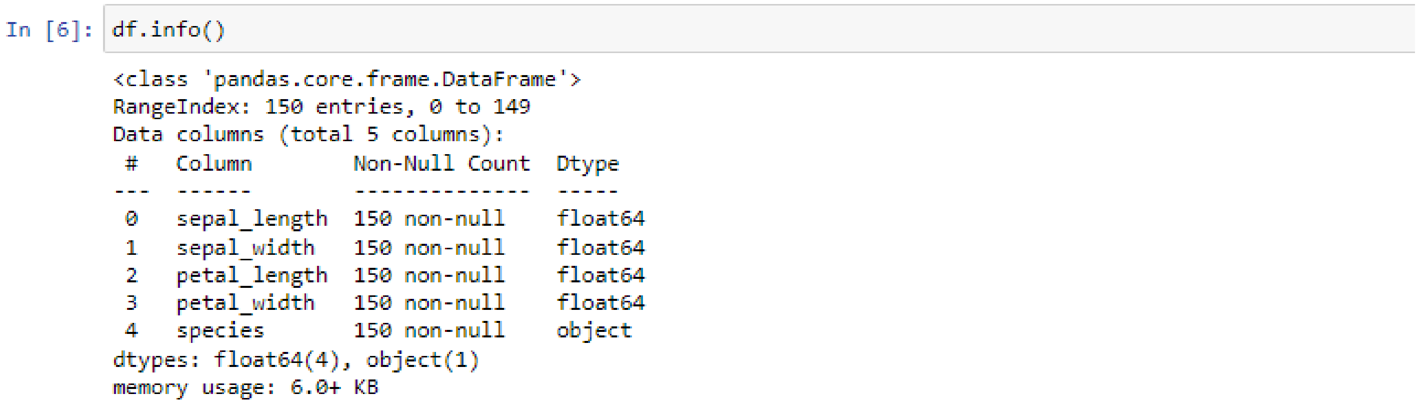
LOAD DATASET

```
In [4]: df = pd.read_csv("IRIS.csv")
```

```
In [5]: df.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
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



```
In [6]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

MISSING VALUES

In [7]:

```
print("sepal_length:")
print("Null_values:",df['sepal_length'].isna().sum())
print("value counts:\n",df['sepal_length'].value_counts())
```

```
sepal_length:
Null_values: 0
value counts:

  5.0    10
 5.1     9
 6.3     9
 5.7     8
 6.7     8
 5.8     7
 5.5     7
 6.4     7
 4.9     6
 5.4     6
 6.1     6
 6.0     6
 5.6     6
 4.8     5
 6.5     5
 6.2     4
 7.7     4
 6.9     4
 4.6     4
 5.2     4
 5.9     3
 4.4     3
 7.2     3
 6.8     3
 6.6     2
 4.7     2
 7.6     1
 7.4     1
 7.3     1
 7.0     1
 7.1     1
 5.3     1
 4.3     1
 4.5     1
 7.9     1
 Name: sepal_length, dtype: int64
```

```
In [8]: print("sepal_width:")
        print("Null_values:",df['sepal_width'].isna().sum())
        print("value counts:\n",df['sepal_width'].value_counts())
```

```
sepal_width:
Null_values: 0
value counts:
 3.0    26
2.8    14
3.2    13
3.1    12
3.4    12
2.9    10
2.7     9
2.5     8
3.5     6
3.3     6
3.8     6
2.6     5
2.3     4
3.7     3
2.4     3
2.2     3
3.6     3
3.9     2
4.4     1
4.0     1
4.1     1
4.2     1
2.0     1
Name: sepal_width, dtype: int64
```

```
In [9]: print("petal_length:")
        print("Null_values:",df['petal_length'].isna().sum())
        print("value counts:\n",df['petal_length'].value_counts())
```

```
petal_length:
Null_values: 0
value counts:
 1.5    14
1.4    12
5.1     8
4.5     8
1.6     7
1.3     7
5.6     6
4.7     5
4.9     5
4.0     5
4.2     4
5.0     4
4.4     4
4.8     4
1.7     4
3.9     3
4.6     3
5.7     3
4.1     3
5.5     3
6.1     3
5.8     3
3.3     2
5.4     2
6.7     2
5.3     2
5.9     2
6.0     2
1.2     2
4.3     2
1.9     2
3.5     2
5.2     2
3.0     1
1.1     1
3.7     1
3.8     1
6.6     1
6.3     1
1.0     1
6.9     1
3.6     1
6.4     1
Name: petal_length, dtype: int64
```

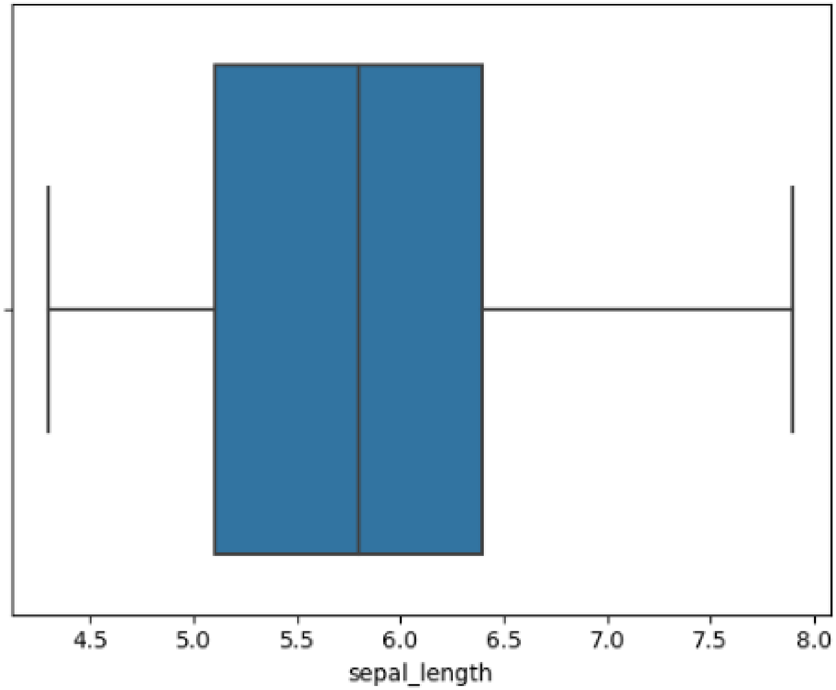
In [10]:
```
print("petal_width:")
print("Null_values:",df['petal_width'].isna().sum())
print("value counts:\n",df['petal_width'].value_counts())
```

```
petal_width:
Null_values: 0
value counts:
 0.2    28
1.3    13
1.8    12
1.5    12
1.4     8
2.3     8
1.0     7
0.4     7
0.3     7
0.1     6
2.1     6
2.0     6
1.2     5
1.9     5
1.6     4
2.5     3
2.2     3
2.4     3
1.1     3
1.7     2
0.6     1
0.5     1
Name: petal_width, dtype: int64
```

OUTLIERS

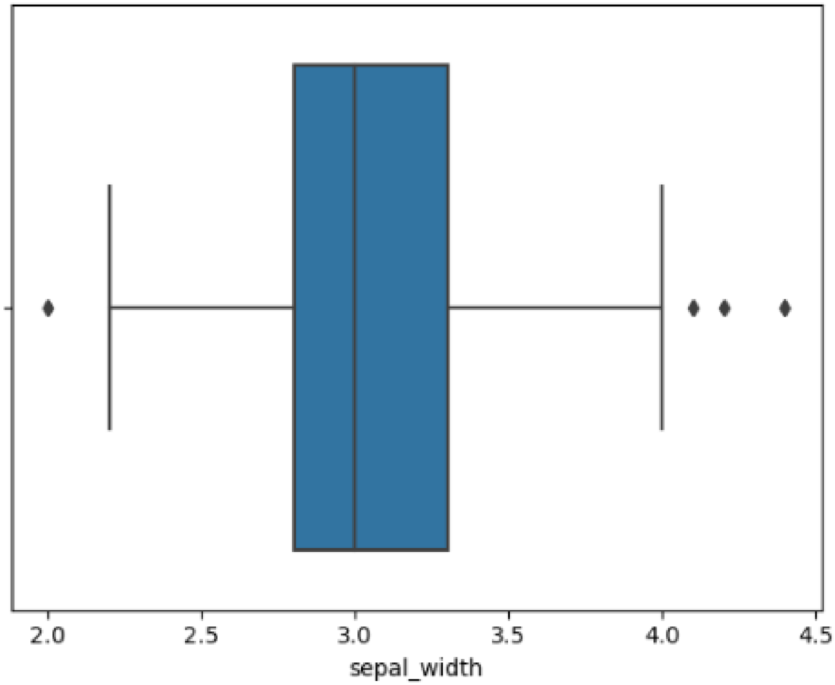
In [13]:
```
sns.boxplot(df["sepal_length"])
plt.show
```

Out[13]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [14]:
```
sns.boxplot(df["sepal_width"])
plt.show
```

Out[14]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [17]:
```
q3 = df["sepal_width"].quantile(0.75)
q1 = df["sepal_width"].quantile(0.25)
iqr = q3-q1
iqr
```

Out[17]:
```
0.5
```

In [18]:
```
upper_whisker = q3+1.5*iqr
lower_whisker = q1-1.5*iqr
```

In [19]:
```
print(iqr)
print(upper_whisker)
print(lower_whisker)
```

```
0.5
4.05
2.05
```

In [20]:
```
upper_whisker_value = df[(df["sepal_width"] > upper_whisker)].index
upper_whisker_value
```

Out[20]:
```
Int64Index([15, 32, 33], dtype='int64')
```

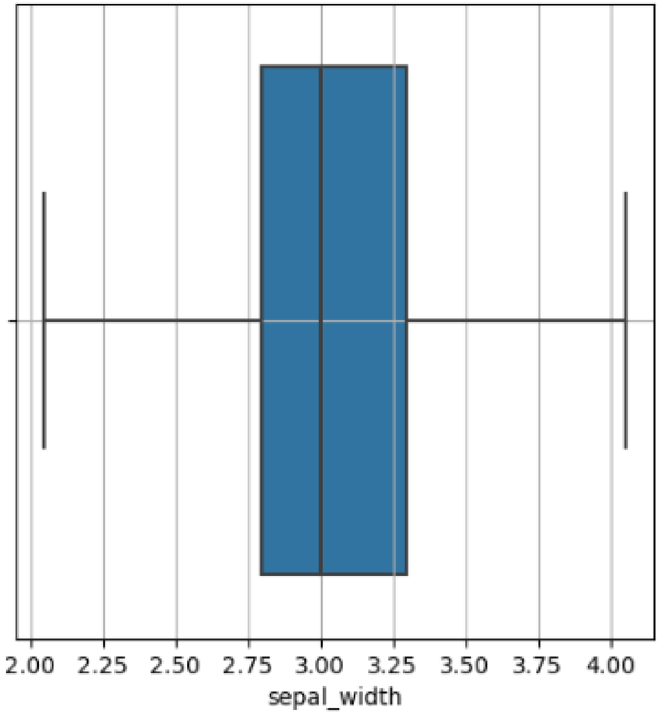
In [21]:
```
lower_whisker_value = df[(df["sepal_width"] < lower_whisker)].index
lower_whisker_value
```

Out[21]:
```
Int64Index([60], dtype='int64')
```

In [22]:
```
df.loc[upper_whisker_value,"sepal_width"] = np.nan
df.fillna(upper_whisker,inplace=True)
```

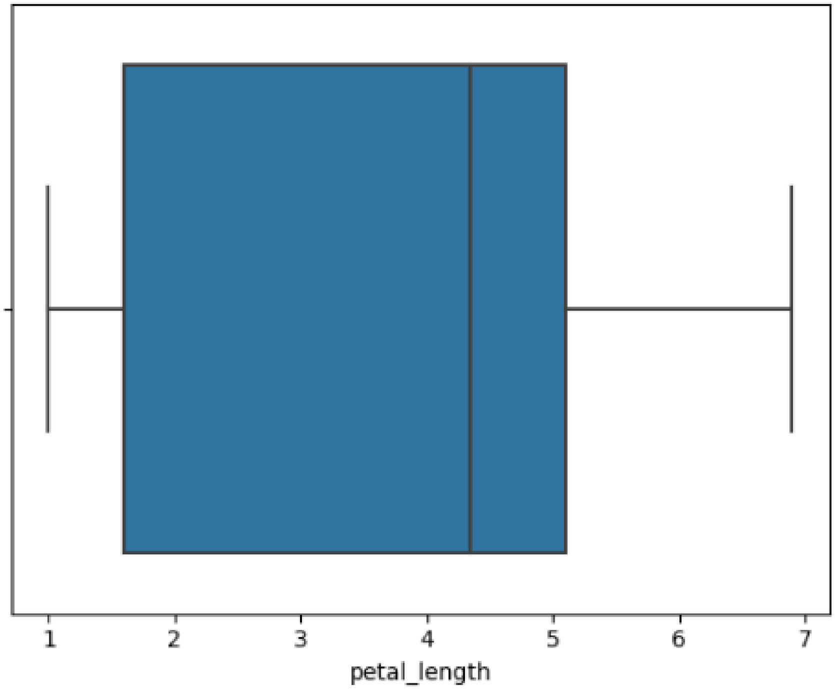
In [23]:
```
df.loc[lower_whisker_value,"sepal_width"] = np.nan
df.fillna(lower_whisker,inplace=True)
```

In [24]:
```
plt.figure(figsize=(5,5))
sns.boxplot(df['sepal_width'])
plt.grid()
```



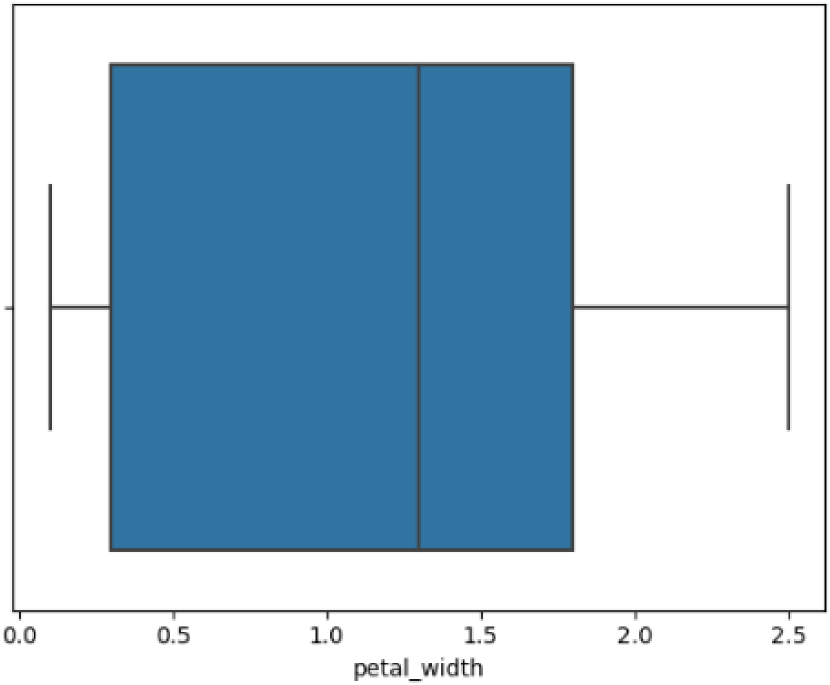
```
In [15]: sns.boxplot(df["petal_length"])
         plt.show
```

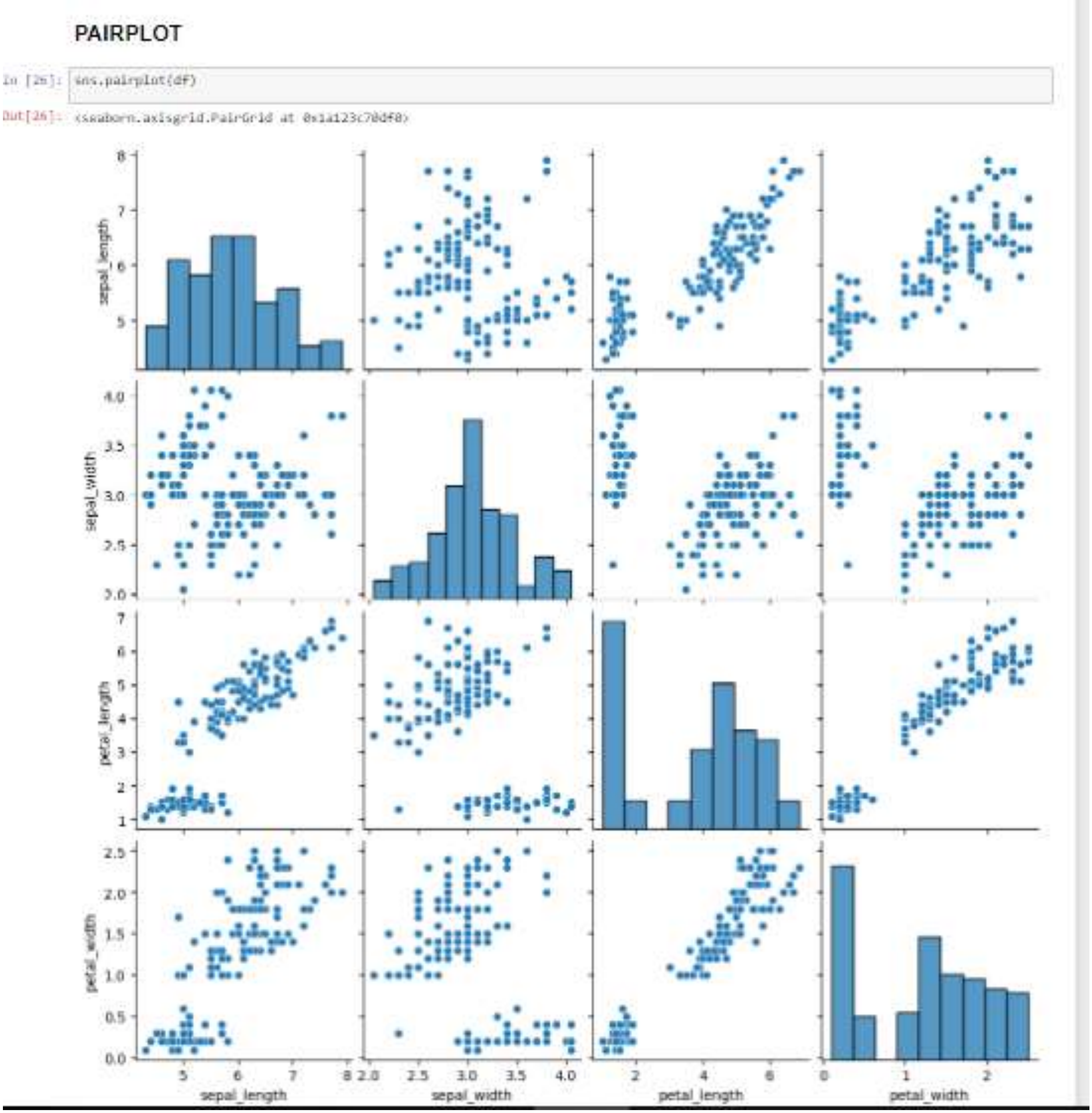
```
Out[15]: <function matplotlib.pyplot.show(close=None, block=None)>
```



```
In [16]: sns.boxplot(df["petal_width"])
         plt.show
```

```
Out[16]: <function matplotlib.pyplot.show(close=None, block=None)>
```





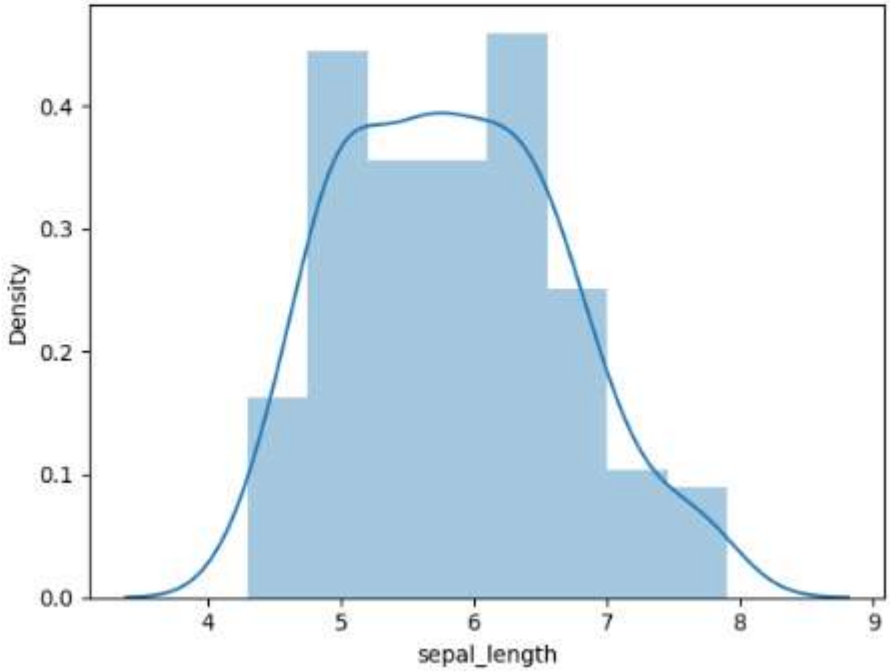
PAIRPLOT

```
In [26]: sns.pairplot(dF)

Out[26]: <seaborn.axisgrid.PairGrid at 0x1a123c70df8>
```

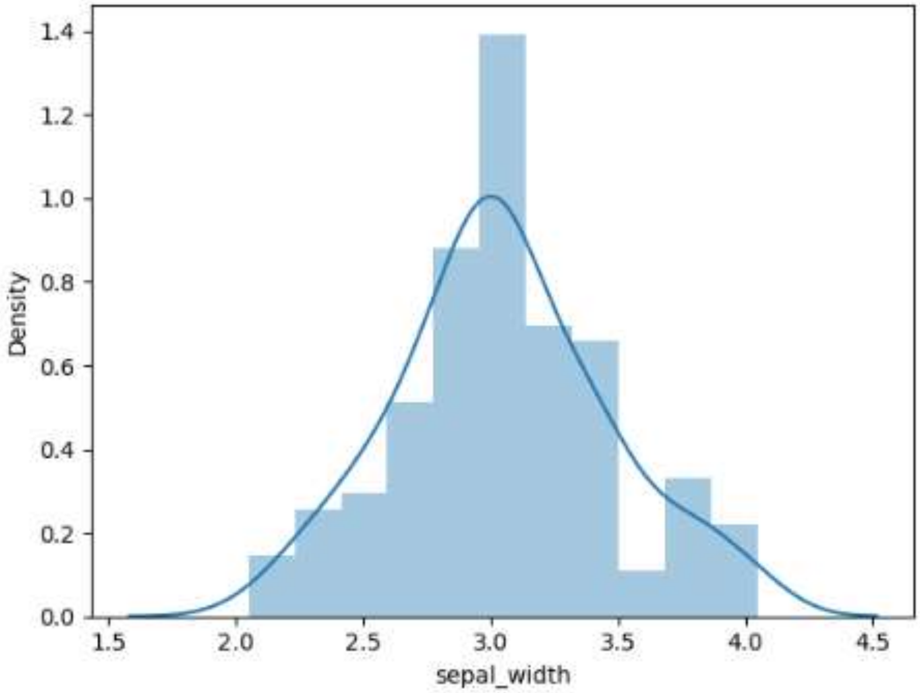
SKEW

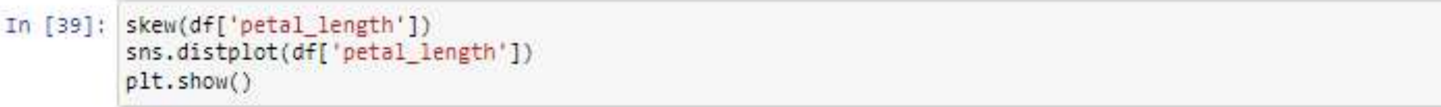
```
In [35]: from scipy.stats import skew
```

```
In [37]: skew(df['sepal_length'])
         sns.distplot(df['sepal_length'])
         plt.show()
```

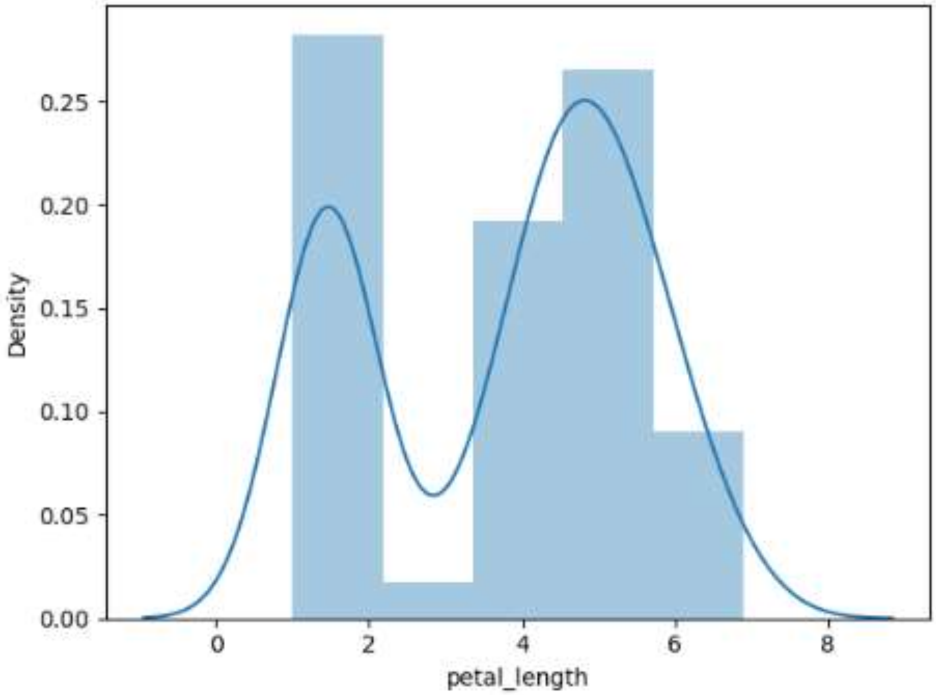


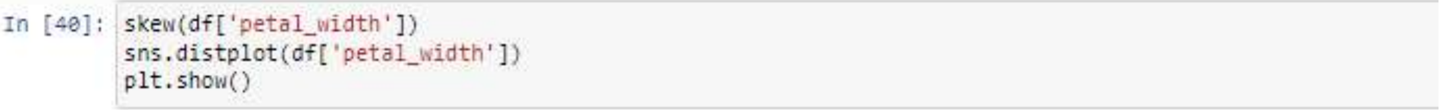
```
In [38]: skew(df['sepal_width'])
         sns.distplot(df['sepal_width'])
         plt.show()
```



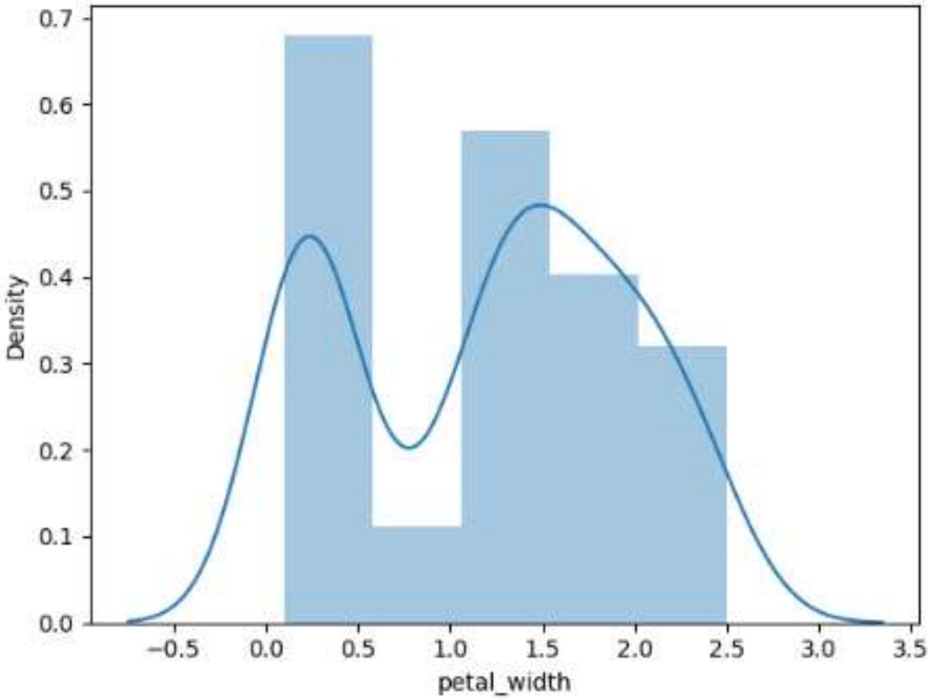


```
In [39]: skew(df['petal_length'])
         sns.distplot(df['petal_length'])
         plt.show()
```





```
In [40]: skew(df['petal_width'])
         sns.distplot(df['petal_width'])
         plt.show()
```



CORRELATION

In [50]:

```
df.corr().style.background_gradient()
```

Out[50]:

	sepal_length	sepal_width	petal_length	petal_width	species
sepal_length	1.000000	-0.110343	0.871754	0.817954	0.782561
sepal_width	-0.110343	1.000000	-0.419823	-0.355582	-0.419264
petal_length	0.871754	-0.419823	1.000000	0.962757	0.949043
petal_width	0.817954	-0.355582	0.962757	1.000000	0.956464
species	0.782561	-0.419264	0.949043	0.956464	1.000000

ENCODING

In [42]:

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
df['species'] = oe.fit_transform(df[['species']])
```

SCALING

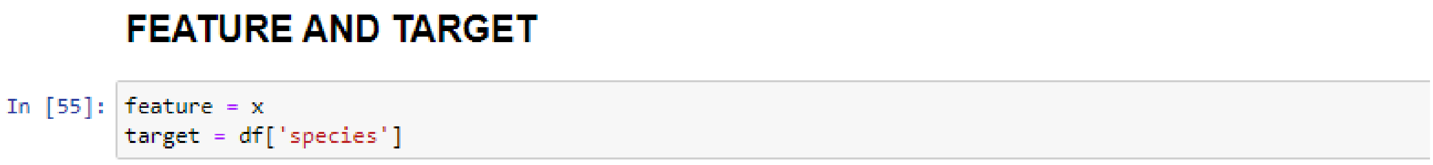
In [54]:

```
x = df.select_dtypes("float64").columns
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
x = sc.fit_transform(feature)
x =pd.DataFrame(x)
x.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
x
```

Out[54]:

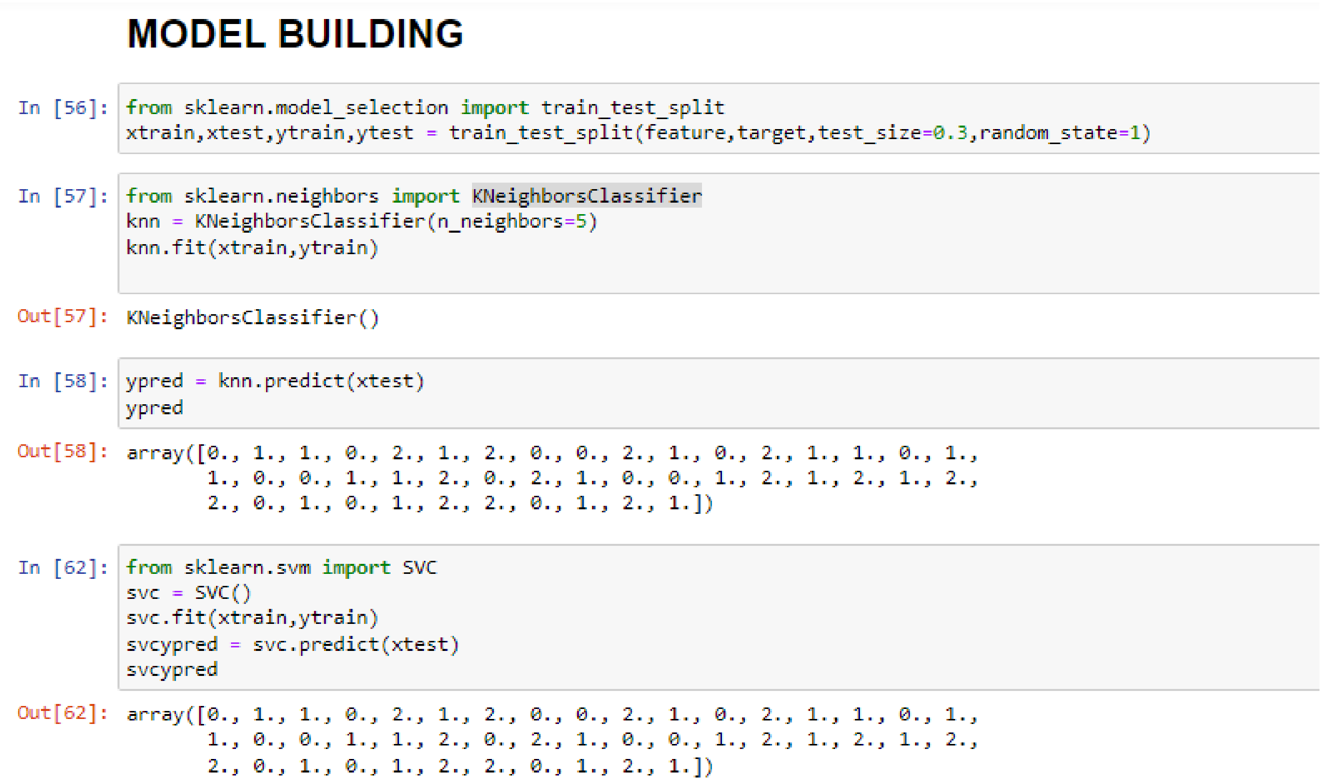
	sepal_length	sepal_width	petal_length	petal_width
0	0.222222	0.725	0.067797	0.041667
1	0.166667	0.475	0.067797	0.041667
2	0.111111	0.575	0.050847	0.041667
3	0.083333	0.525	0.084746	0.041667
4	0.194444	0.775	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.475	0.711864	0.916667
146	0.555556	0.225	0.677966	0.750000
147	0.611111	0.475	0.711864	0.791667
148	0.527778	0.675	0.745763	0.916667
149	0.444444	0.475	0.694915	0.708333

150 rows × 4 columns



FEATURE AND TARGET

```
In [55]: feature = x
         target = df['species']
```



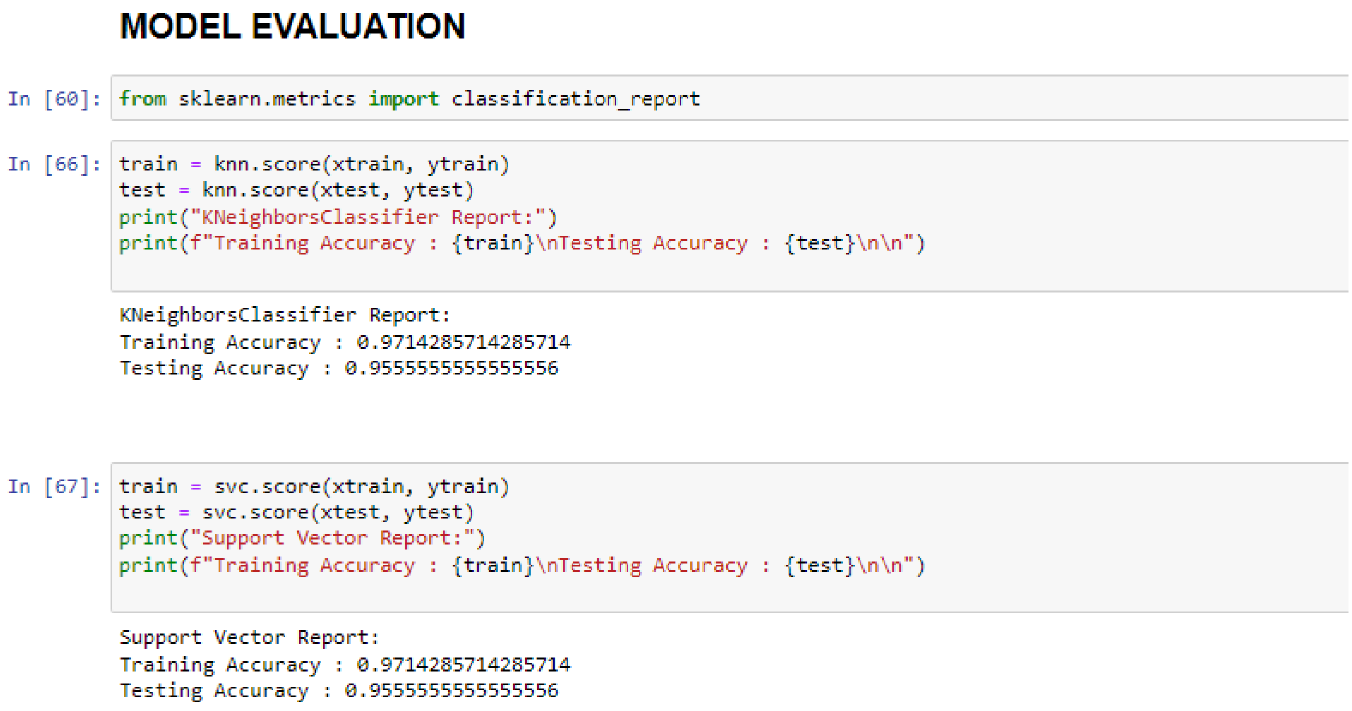
MODEL BUILDING

```
In [56]: from sklearn.model_selection import train_test_split
         xtrain,xtest,ytrain,ytest = train_test_split(feature,target,test_size=0.3,random_state=1)
```

```
In [57]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(xtrain,ytrain)

Out[57]: KNeighborsClassifier()
```

```
In [58]: ypred = knn.predict(xtest)
         ypred

Out[58]: array([0., 1., 1., 0., 2., 1., 2., 0., 0., 2., 1., 0., 2., 1., 1., 0., 1.,
                1., 0., 0., 1., 1., 2., 0., 2., 1., 0., 0., 1., 2., 1., 2., 1., 2.,
                2., 0., 1., 0., 1., 2., 2., 0., 1., 2., 1.])
```

```
In [62]: from sklearn.svm import SVC
         svc = SVC()
         svc.fit(xtrain,ytrain)
         svcypred = svc.predict(xtest)
         svcypred

Out[62]: array([0., 1., 1., 0., 2., 1., 2., 0., 0., 2., 1., 0., 2., 1., 1., 0., 1.,
                1., 0., 0., 1., 1., 2., 0., 2., 1., 0., 0., 1., 2., 1., 2., 1., 2.,
                2., 0., 1., 0., 1., 2., 2., 0., 1., 2., 1.])
```



MODEL EVALUATION

```
In [60]: from sklearn.metrics import classification_report
```

```
In [66]: train = knn.score(xtrain, ytrain)
         test = knn.score(xtest, ytest)
         print("KNeighborsClassifier Report:")
         print(f"Training Accuracy : {train}\nTesting Accuracy : {test}\n\n")

         KNeighborsClassifier Report:
         Training Accuracy : 0.9714285714285714
         Testing Accuracy : 0.9555555555555556
```

```
In [67]: train = svc.score(xtrain, ytrain)
         test = svc.score(xtest, ytest)
         print("Support Vector Report:")
         print(f"Training Accuracy : {train}\nTesting Accuracy : {test}\n\n")

         Support Vector Report:
         Training Accuracy : 0.9714285714285714
         Testing Accuracy : 0.9555555555555556
```