In [1]:
```
import sklearn
import numpy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:
```
df=pd.read_csv('C:\\Users\\sanki\\OneDrive\\Desktop\\iris.data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [4]:
```
# some basic statistical analaysis adout the data
df.describe()
```

Out[4]:

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [5]:
```
#basic info of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [6]:
```
#sample of each class
#class=iris-setosa
df['Iris-setosa'].value_counts()
```

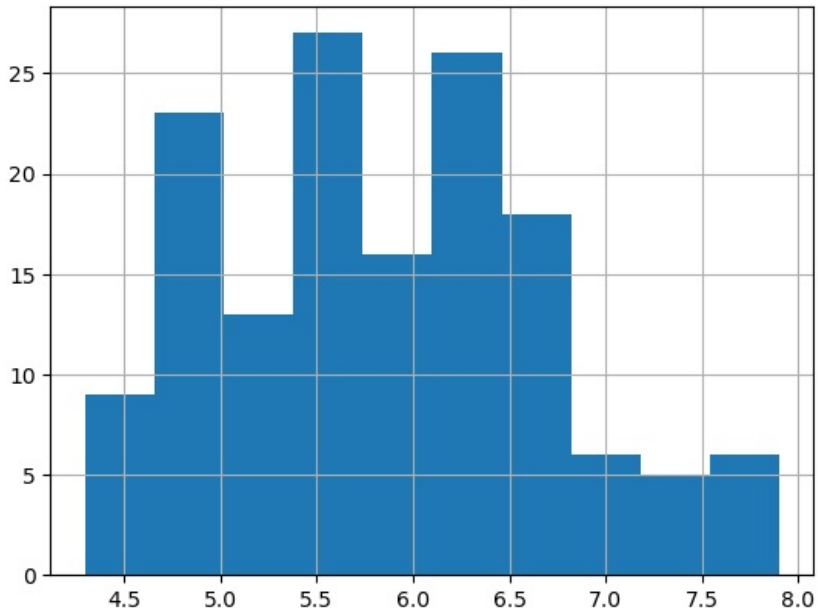
Out[6]:
```
Iris-setosa
Iris-versicolor    50
Iris-virginica     50
Iris-setosa        49
Name: count, dtype: int64
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
5.1            0
3.5            0
1.4            0
0.2            0
Iris-setosa    0
dtype: int64
```

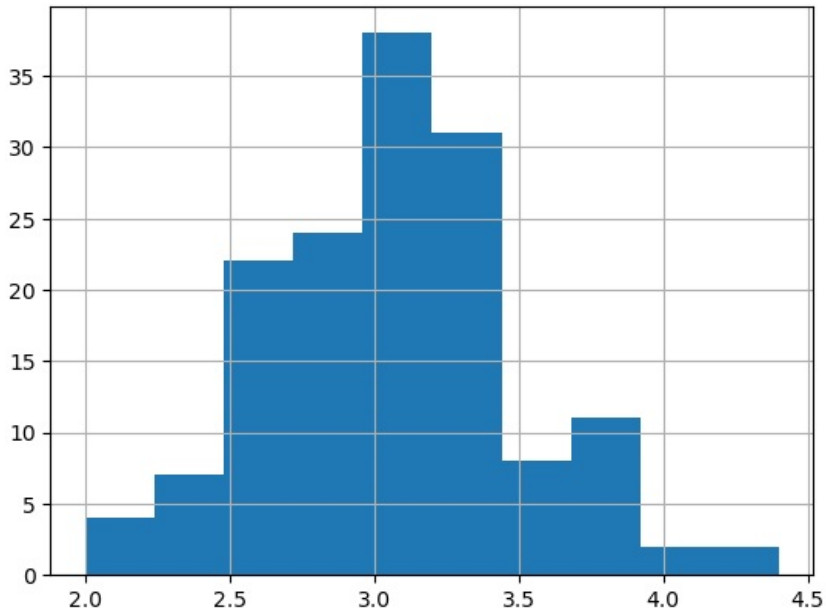
In [8]:
```
#5.1=sepallength
df['5.1'].hist()
```

Out[8]:
```
<Axes: >
```



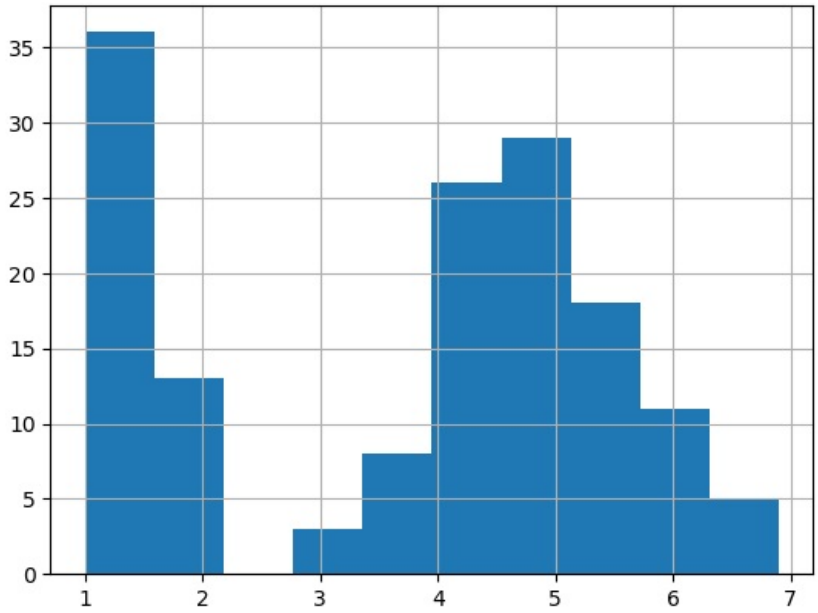
In [9]:
```
df['3.5'].hist()
```

Out[9]:
```
<Axes: >
```



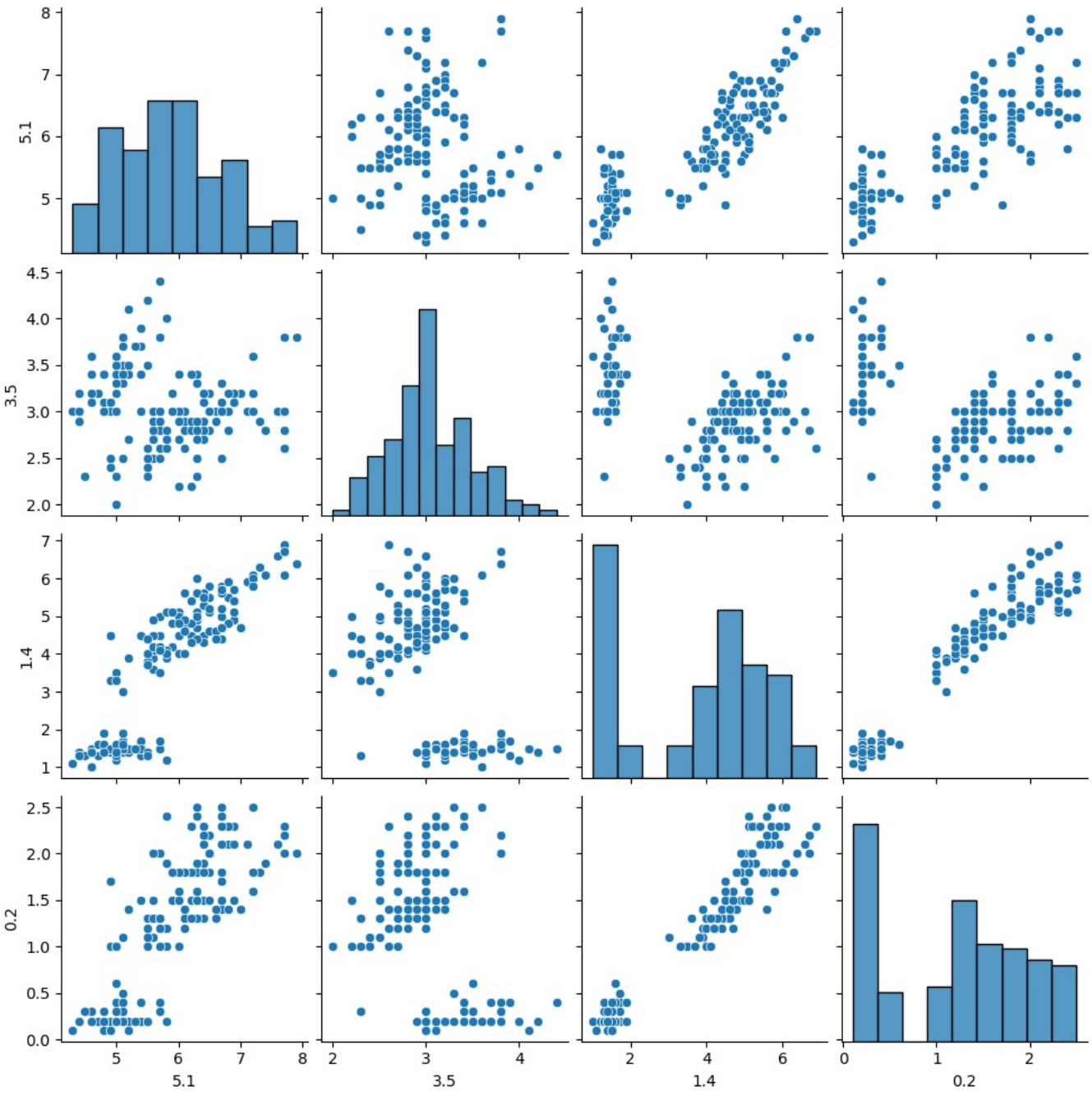
In [10]:
```
df['1.4'].hist()
```

Out[10]:
```
<Axes: >
```



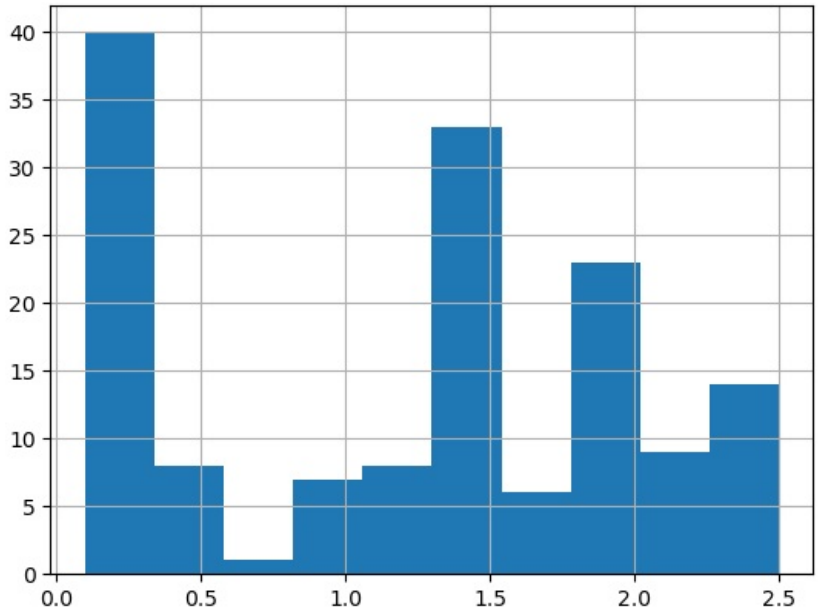
In [11]:
```
sns.pairplot(df)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x2073b033b90>
```


5.1
3.5
1.4
0.2

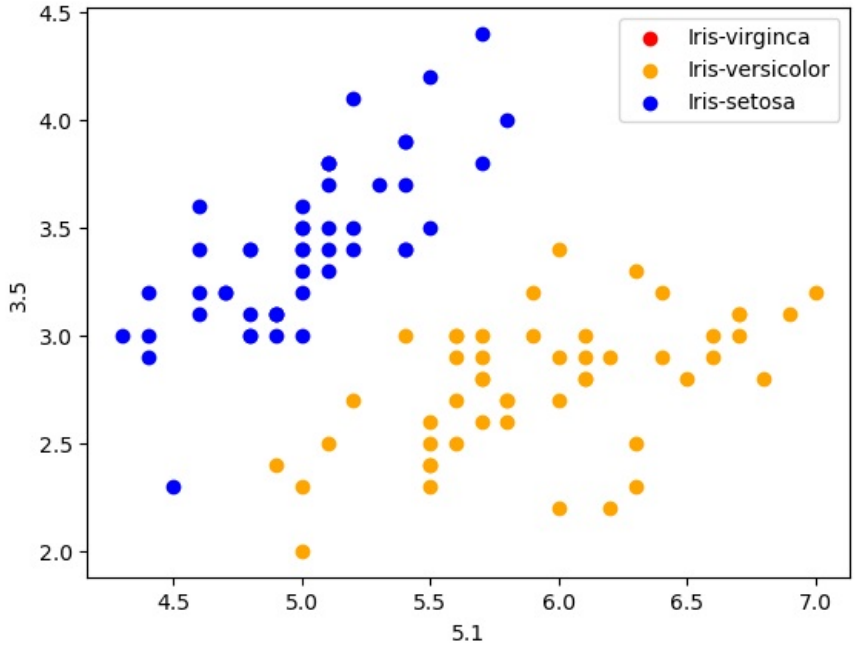
In [12]:
```
#petalwidth=0.2
df['0.2'].hist()
```

Out[12]:
```
<Axes: >
```

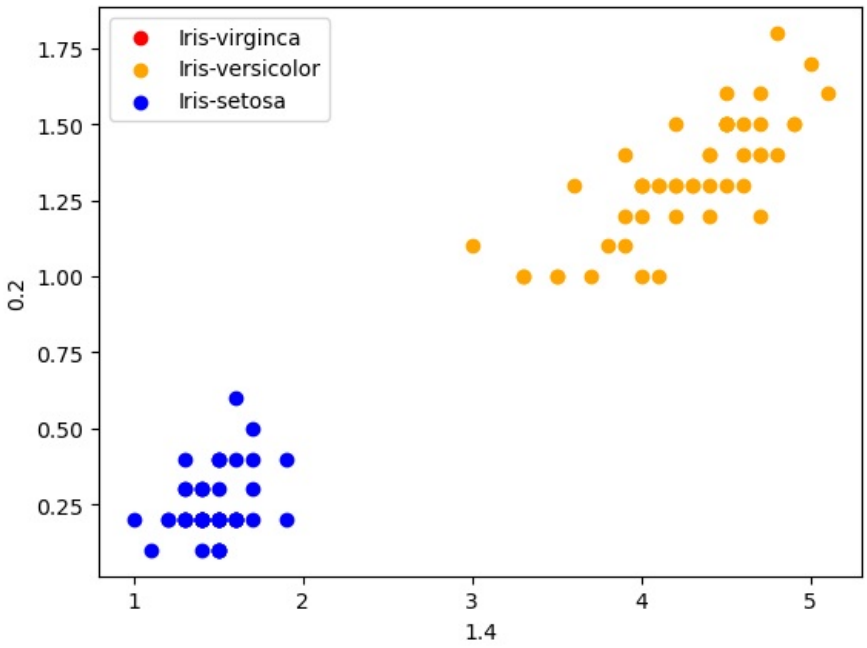


In [13]:
```
colors=['red','orange','blue']
species=['Iris-virginca','Iris-versicolor','Iris-setosa']
```

In [14]:
```
for i in range(3):
    x=df[df['Iris-setosa']==species[i]]
    plt.scatter(x['5.1'],x['3.5'],c=colors[i],label=species[i])
    plt.xlabel('5.1')
    plt.ylabel('3.5')
    plt.legend()
```



In [15]:
```
for i in range(3):
    x=df[df['Iris-setosa']==species[i]]
    plt.scatter(x['1.4'],x['0.2'],c=colors[i],label=species[i])
    plt.xlabel('1.4')
    plt.ylabel('0.2')
    plt.legend()
```



In [16]:
```
x=df.drop(['Iris-setosa'],axis=1)
y=df['Iris-setosa']
```

In [17]:
```
print(x)
```

```
     5.1  3.5  1.4  0.2
0    4.9  3.0  1.4  0.2
1    4.7  3.2  1.3  0.2
2    4.6  3.1  1.5  0.2
3    5.0  3.6  1.4  0.2
4    5.4  3.9  1.7  0.4
..   ...  ...  ...  ...
144  6.7  3.0  5.2  2.3
145  6.3  2.5  5.0  1.9
146  6.5  3.0  5.2  2.0
147  6.2  3.4  5.4  2.3
148  5.9  3.0  5.1  1.8

[149 rows x 4 columns]
```

In [18]:
```
print(y)
```

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

In [21]:
```
#seperate features and target
data=df.values
x=data[:,0:4]
y=data[:,4]
```

In [22]:
```
#model training
from sklearn.model_selection import train_test_split
X=df.drop(columns=['Iris-setosa'])
Y=df['Iris-setosa']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [23]:
```
from sklearn.linear_model import LogisticRegression
model= LogisticRegression()
```

In [24]:
```
model.fit(x_train, y_train)
```

Out[24]:
```
▾ LogisticRegression
LogisticRegression()
```

In [25]:
```
#print performance matric
print("Accuracy:",model.score(x_test,y_test)*100)
```

```
Accuracy: 93.33333333333333
```

In [ ]: