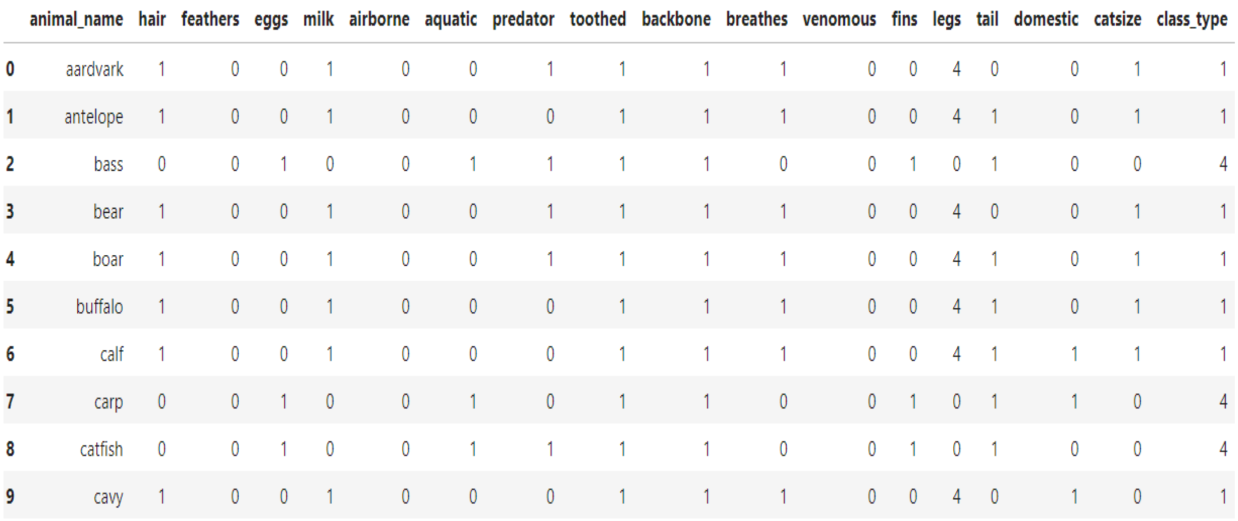
—-------import libraries—---------------

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold,cross_val_score,GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score
from sklearn.preprocessing import StandardScaler
```

—---------read dataset—-----------------

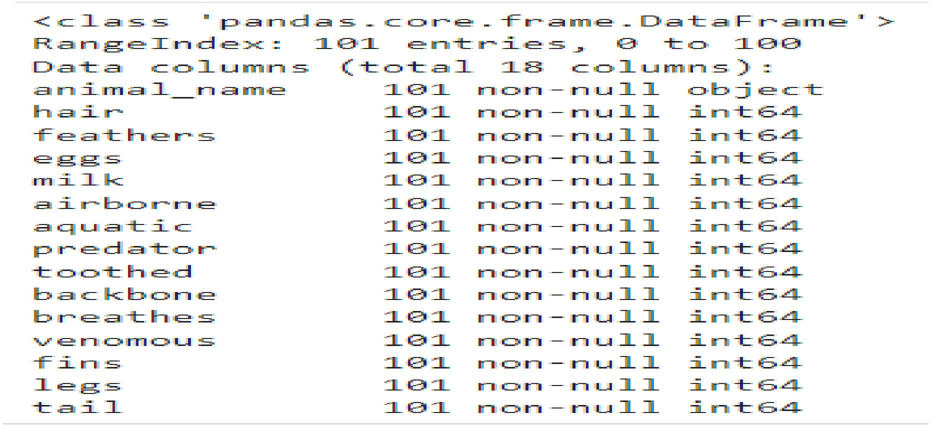
```
data = pd.read_csv('Downloads/Zoo.csv')
data.head(6)
```



	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	class_type
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
5	buffalo	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
6	calf	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
7	carp	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	4
8	catfish	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
9	cavy	1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	1

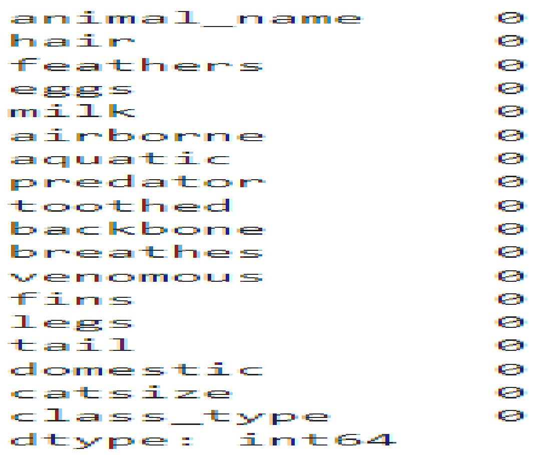
—----------info—-------------

```
data.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 18 columns):
animal_name    101 non-null object
hair           101 non-null int64
feathers       101 non-null int64
eggs           101 non-null int64
milk           101 non-null int64
airborne       101 non-null int64
aquatic        101 non-null int64
predator       101 non-null int64
toothed        101 non-null int64
backbone       101 non-null int64
breathes       101 non-null int64
venomous       101 non-null int64
fins           101 non-null int64
legs           101 non-null int64
tail           101 non-null int64
```

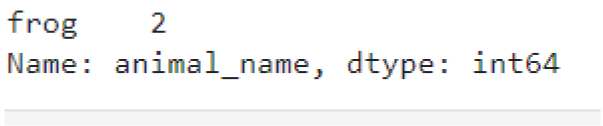
```
data.isna().sum()
```



```
animal_name    0
hair           0
feathers       0
eggs           0
milk           0
airborne       0
aquatic        0
predator       0
toothed        0
backbone       0
breathes       0
venomous       0
fins           0
legs           0
tail           0
domestic       0
catsize        0
class_type     0
dtype: int64
```

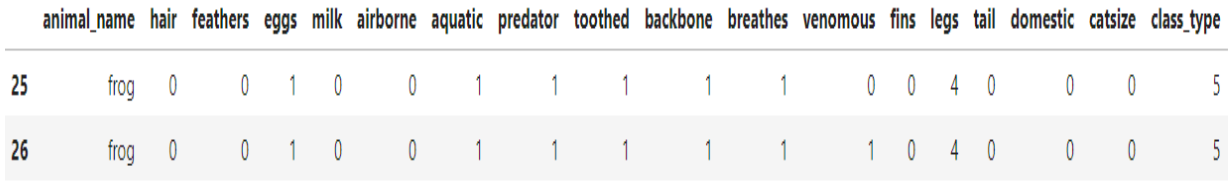
—----duplicates—----------------

duplicates = data.animal_name.value_counts()
duplicates(duplicates>1)



```
frog    2
Name: animal_name, dtype: int64
```

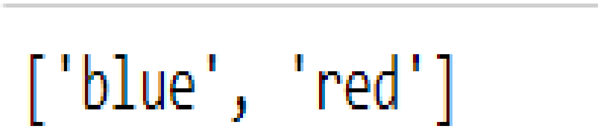
frog = data.loc[data['animal_name']=='frog']
frog



	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	class_type
25	frog	0	0	1	0	0	1	1	1	1	1	0	0	4	0	0	0	5
26	frog	0	0	1	0	0	1	1	1	1	1	1	0	4	0	0	0	5

data['animal_name'][(data.venomous==1) & (data.animal_name=='frog')] = 'frog2'

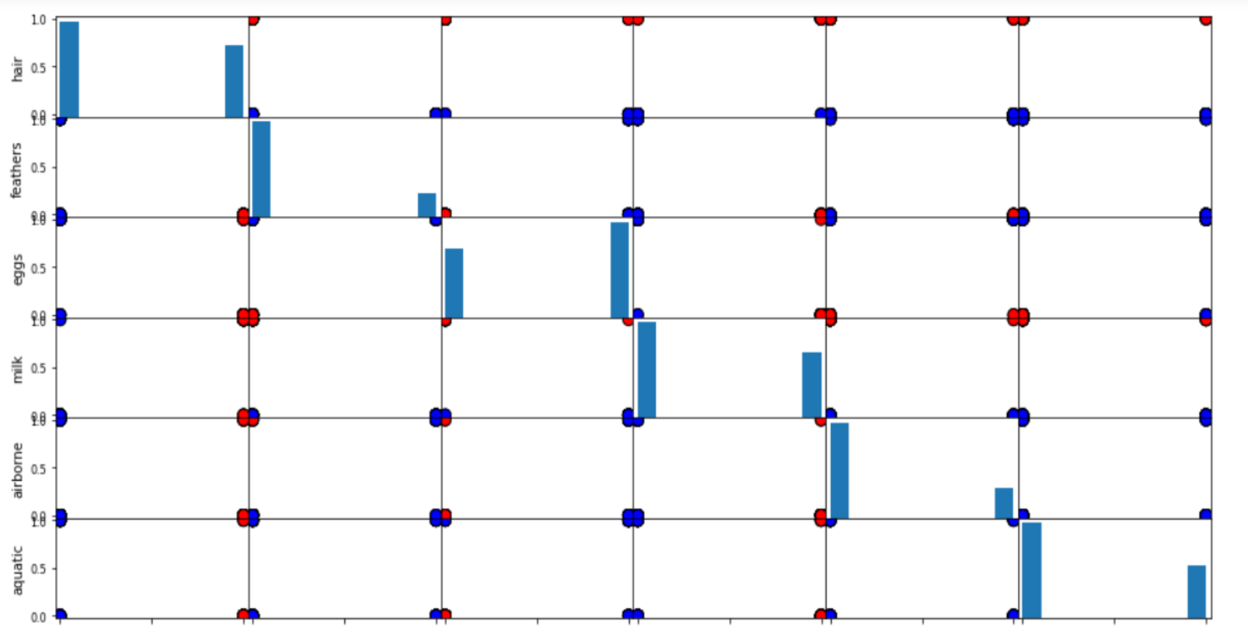
color_list = [('red' if i==1 else 'blue' if i==0 else 'yellow') for i in data.hair]
unique_color = list(set(color_list))
unique_color



```
['blue', 'red']
```

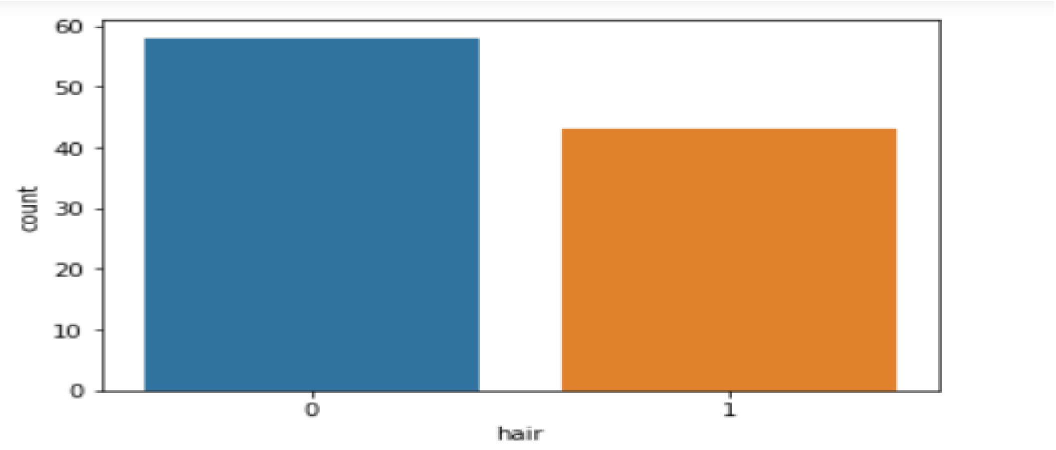
—----------plot—--------------

pd.plotting.scatter_matrix(data.iloc[:,:7],c=color_list,figsize=(16,8),diagonal='hist',alpha=1,s=300, edgecolor='black')



—-----------plot2—---------------

```
sns.countplot(x='hair',data=data)
plt.xlabel('hair')
plt.ylabel('count')
```

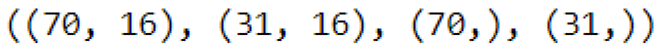


—---------divide—-----------------------

```
x = data.iloc[:,1:17]
y = data.iloc[:,17]
```

—--------traintest—-----------------

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

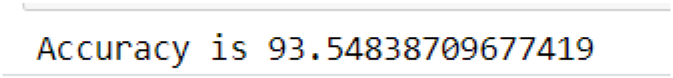


```
((70, 16), (31, 16), (70,), (31,))
```

—---------knn—----------------------

```
clf = KNeighborsClassifier(n_neighbors=3)
```

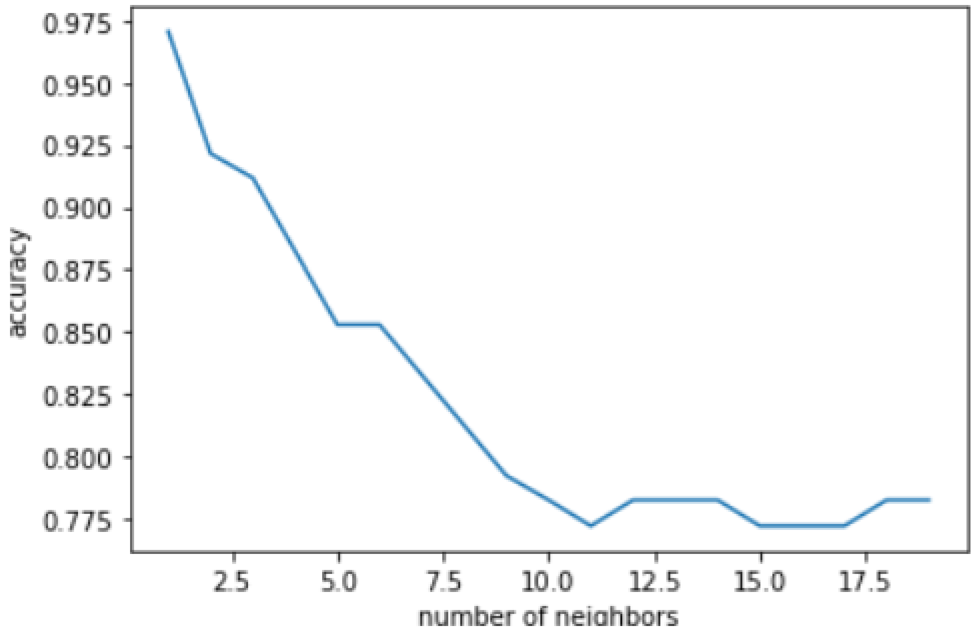
```
clf.fit(x_train,y_train)
predict = clf.predict(x_test)
a = accuracy_score(predict,y_test)*100
print('Accuracy is',a)
```



```
Accuracy is 93.54838709677419
```

```
—----------plot—------------------------
krange = np.arange(1,20)
kscore = []
for k in krange:
   knn = KNeighborsClassifier(n_neighbors=k)
   score = cross_val_score(knn,x,y,cv=6)
   kscore.append(score.mean())
plt.plot(krange,kscore)
plt.xlabel('number of neighbors')
plt.ylabel('accuracy')
```

```
Text(0, 0.5, 'accuracy')
```


accuracy
number of neighbors