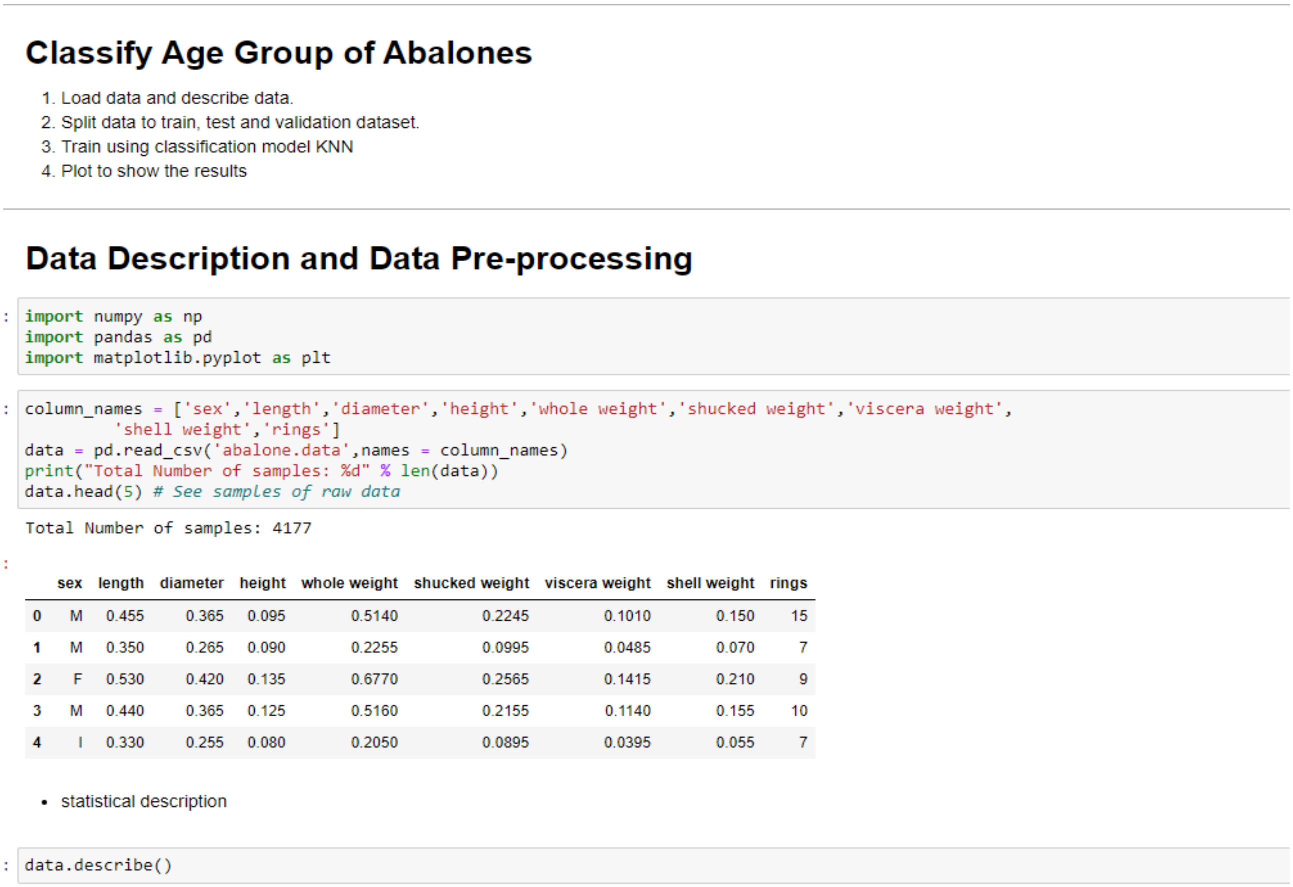
Koushik Sahu

118CS0597

Soft Computing Lab – 9

21st March 2022

Code:



Classify Age Group of Abalones

1. Load data and describe data.
2. Split data to train, test and validation dataset.
3. Train using classification model KNN
4. Plot to show the results

Data Description and Data Pre-processing

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

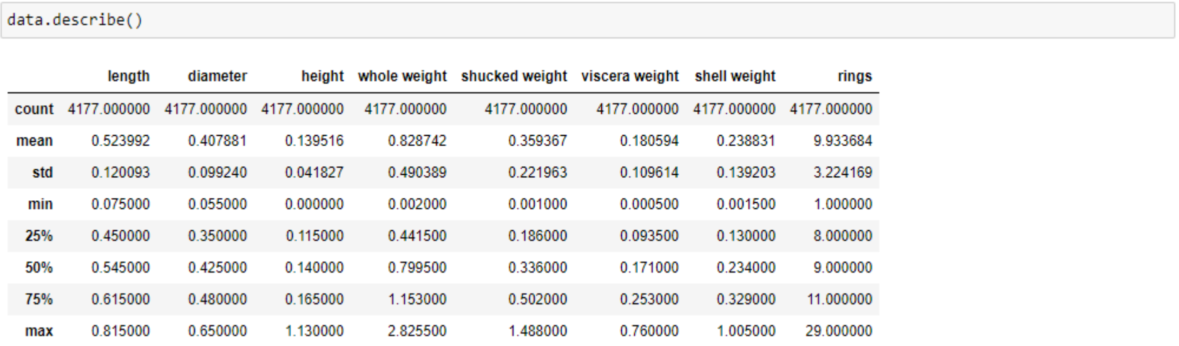
```
column_names = ['sex','length','diameter','height','whole weight','shucked weight','viscera weight',
         'shell weight','rings']
data = pd.read_csv('abalone.data',names = column_names)
print("Total Number of samples: %d" % len(data))
data.head(5) # See samples of raw data
```

```
Total Number of samples: 4177
```

	sex	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

- statistical description

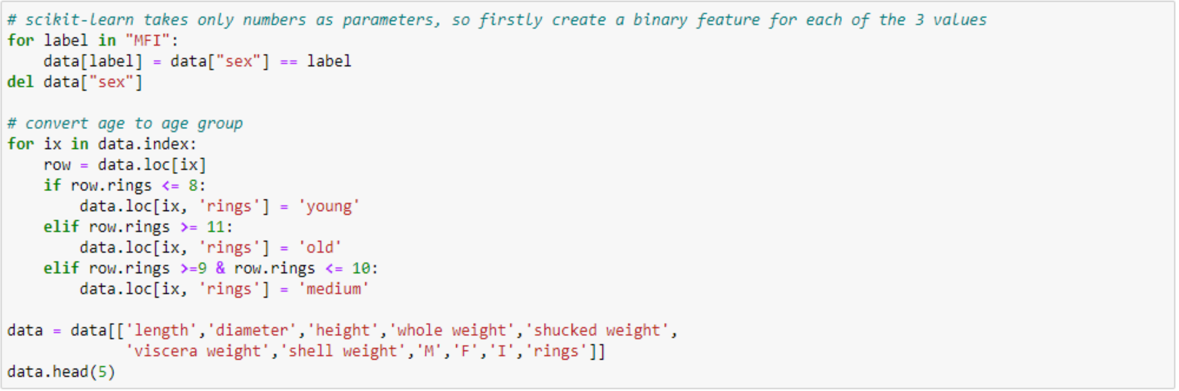
```
data.describe()
```



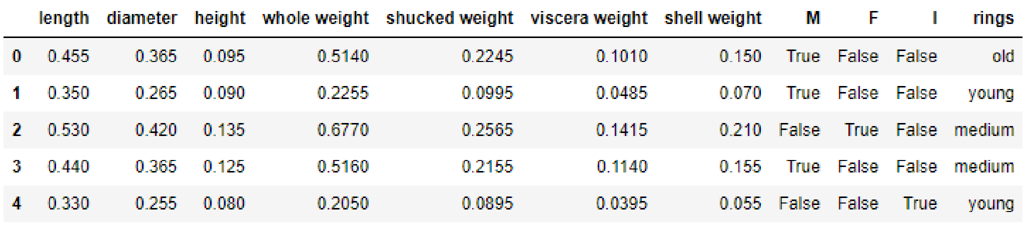
```
data.describe()
```

	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

- Data pre-processing

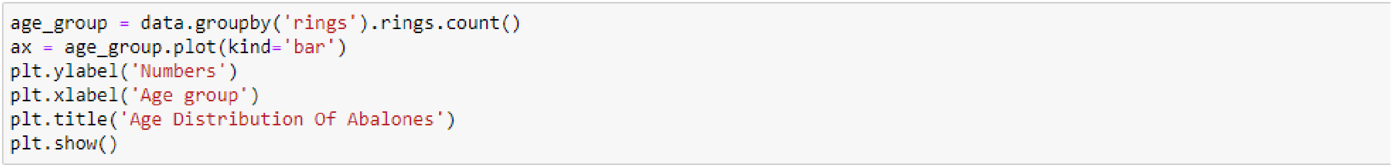


```
# scikit-learn takes only numbers as parameters, so firstly create a binary feature for each of the 3 values
for label in "MFI":
    data[label] = data["sex"] == label
del data["sex"]

# convert age to age group
for ix in data.index:
    row = data.loc[ix]
    if row.rings <= 8:
        data.loc[ix, 'rings'] = 'young'
    elif row.rings >= 11:
        data.loc[ix, 'rings'] = 'old'
    elif row.rings >=9 & row.rings <= 10:
        data.loc[ix, 'rings'] = 'medium'

data = data[['length','diameter','height','whole weight','shucked weight',
             'viscera weight','shell weight','M','F','I','rings']]
data.head(5)
```

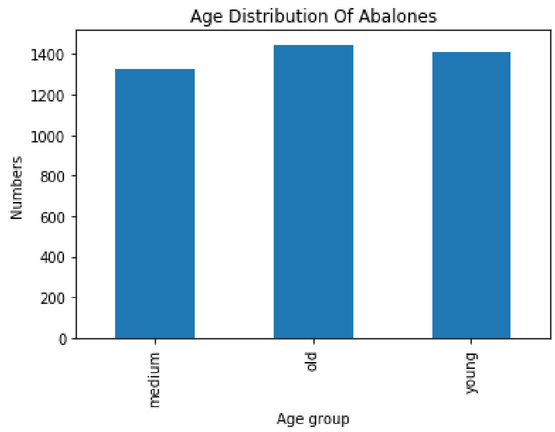


	length	diameter	height	whole weight	shucked weight	viscera weight	shell weight	M	F	I	rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	True	False	False	old
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	True	False	False	young
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	False	True	False	medium
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	True	False	False	medium
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	False	False	True	young

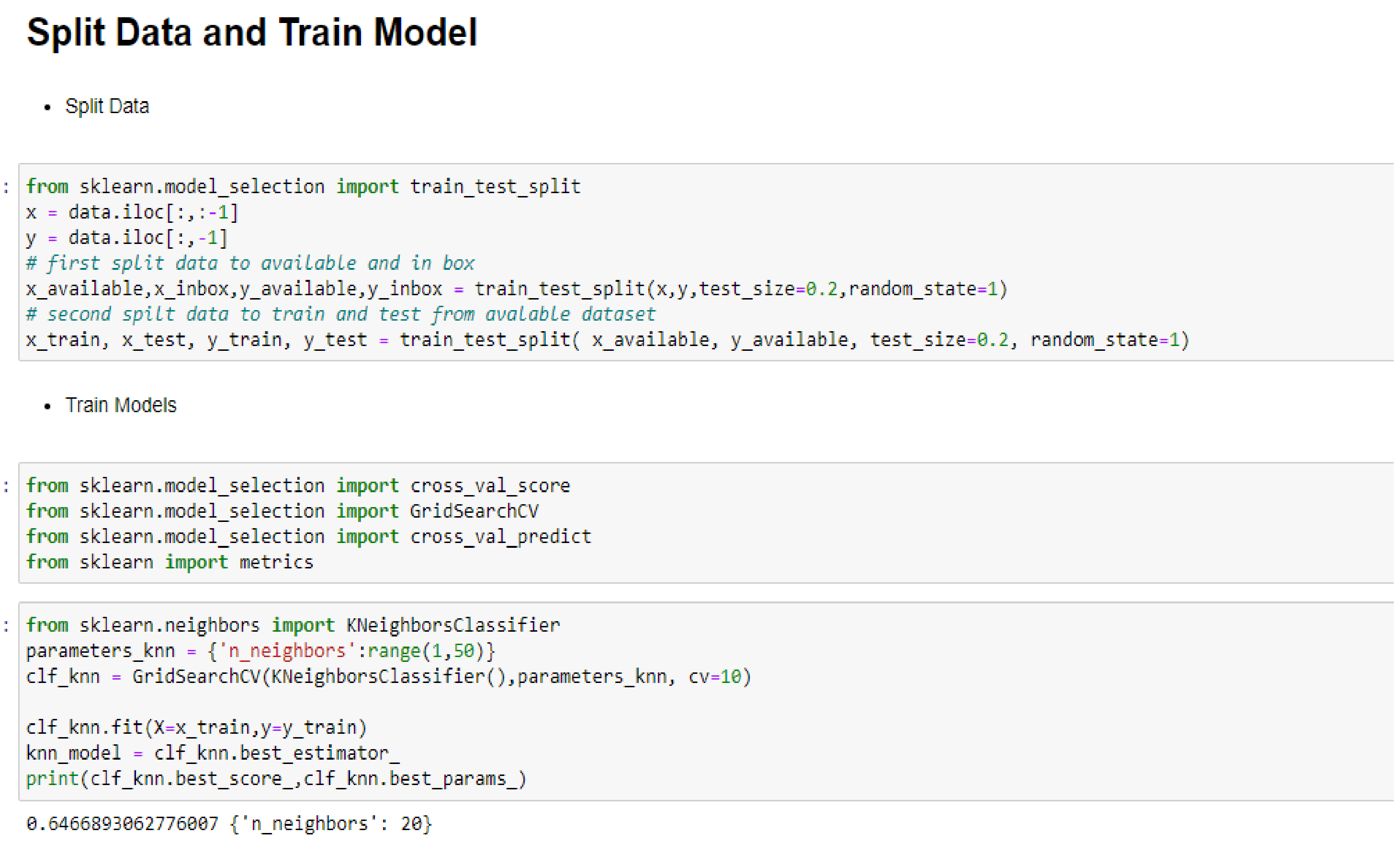
- To see if the dataset has the problem of unbalanced data



```
age_group = data.groupby('rings').rings.count()
ax = age_group.plot(kind='bar')
plt.ylabel('Numbers')
plt.xlabel('Age group')
plt.title('Age Distribution Of Abalones')
plt.show()
```



The plot shows that the data distributed balanced



Split Data and Train Model

- Split Data

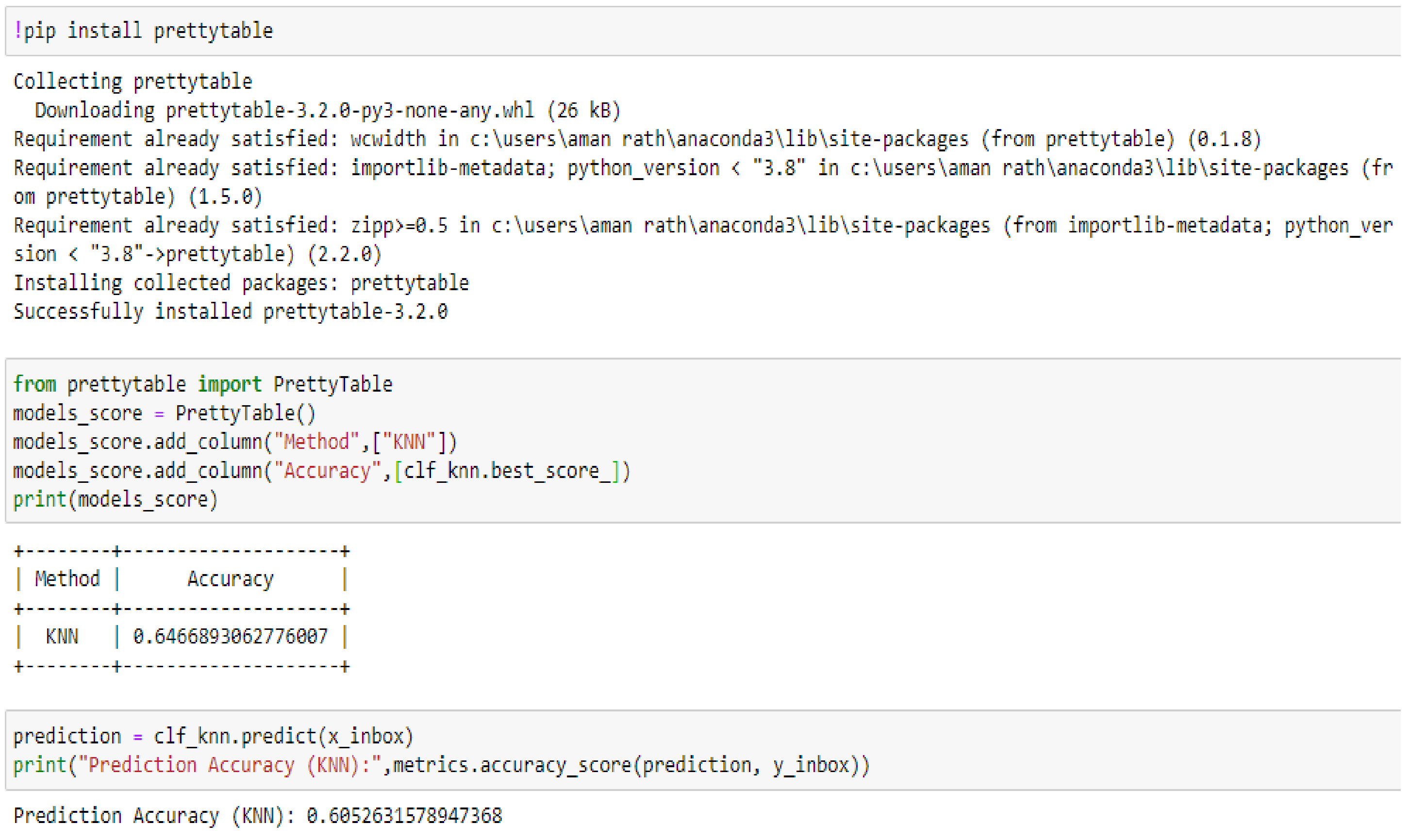
```
from sklearn.model_selection import train_test_split
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
# first split data to available and in box
x_available,x_inbox,y_available,y_inbox = train_test_split(x,y,test_size=0.2,random_state=1)
# second split data to train and test from avalable dataset
x_train, x_test, y_train, y_test = train_test_split( x_available, y_available, test_size=0.2, random_state=1)
```

- Train Models

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_predict
from sklearn import metrics
```

```
from sklearn.neighbors import KNeighborsClassifier
parameters_knn = {'n_neighbors':range(1,50)}
clf_knn = GridSearchCV(KNeighborsClassifier(),parameters_knn, cv=10)

clf_knn.fit(X=x_train,y=y_train)
knn_model = clf_knn.best_estimator_
print(clf_knn.best_score_,clf_knn.best_params_)
```

```
0.6466893062776007 {'n_neighbors': 20}
```



```
!pip install prettytable
```

```
Collecting prettytable
  Downloading prettytable-3.2.0-py3-none-any.whl (26 kB)
Requirement already satisfied: wcwidth in c:\users\aman rath\anaconda3\lib\site-packages (from prettytable) (0.1.8)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in c:\users\aman rath\anaconda3\lib\site-packages (fr
om prettytable) (1.5.0)
Requirement already satisfied: zipp>=0.5 in c:\users\aman rath\anaconda3\lib\site-packages (from importlib-metadata; python_ver
sion < "3.8"->prettytable) (2.2.0)
Installing collected packages: prettytable
Successfully installed prettytable-3.2.0
```

```
from prettytable import PrettyTable
models_score = PrettyTable()
models_score.add_column("Method",["KNN"])
models_score.add_column("Accuracy",[clf_knn.best_score_])
print(models_score)
```

```
+--------+--------------------+
| Method |      Accuracy      |
+--------+--------------------+
|  KNN   | 0.6466893062776007 |
+--------+--------------------+
```

```
prediction = clf_knn.predict(x_inbox)
print("Prediction Accuracy (KNN):",metrics.accuracy_score(prediction, y_inbox))
```

```
Prediction Accuracy (KNN): 0.6052631578947368
```