Machine Learning Fundamentals Lab-7

Name: Gaurav Prasanna
Reg No: 19BEC1315

Aim:

a) To implement KNN algorithm on abalone dataset and find out the required predictions.
b) To implement KNN algorithm on same data as above and find the loss using mean squared error.

Software Required:

1) Jupyter Notebook
2) Anaconda Navigator

Libraries Required: Numpy, Matplotlib, Sci-kit Learn, Pandas.

Code and Outputs:

a) KNN Required Predictions:

In [ ]:
```
import pandas as pd
abalone=pd.read_csv('abalone.csv',header=None)
```

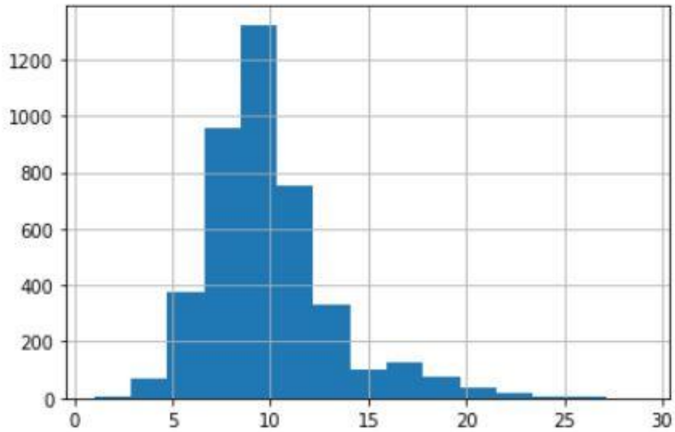
In [ ]:
```
abalone.head()
```

Out[18]:

	0	1	2	3	4	5	6	7	8
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

In [ ]:
```
abalone.columns= ["Sex", "Length", "Diameter", "Height", "Whole weight", "Shucked weight", "Viscera weight", "Shell weight", "Ri
abalone = abalone.drop("Sex", axis=1)
```

```
In [ ]: import matplotlib.pyplot as plt
        abalone["Rings"].hist(bins=15)
        plt.show()
```




```
In [ ]: correlation_matrix = abalone.corr()
        correlation_matrix["Rings"]

Out[21]: Length            0.556720
         Diameter          0.574660
         Height            0.557467
         Whole weight      0.540390
         Shucked weight    0.420884
         Viscera weight    0.503819
         Shell weight      0.627574
         Rings             1.000000
         Name: Rings, dtype: float64
```

```
In [ ]: X = abalone.drop("Rings", axis=1)
        X = X.values
        y = abalone["Rings"]
        y = y.values

In [ ]: import numpy as np
        new_data_point = np.array([0.569552,0.446407,0.154437,1.016849,0.439051,0.222526,0.291208])

In [ ]: distances = np.linalg.norm(X-new_data_point, axis=1)
        k = 7
        nearest_neighbor_ids = distances.argsort()[:k]
        nearest_neighbor_ids

Out[24]: array([4045, 1902, 1644, 1132, 1894, 3915, 3668])

In [ ]: nearest_neighbor_rings = y[nearest_neighbor_ids]
        nearest_neighbor_rings

Out[25]: array([ 9, 11, 10,  9, 11, 11, 10])

In [ ]: prediction = nearest_neighbor_rings.mean()
        prediction

Out[26]: 10.142857142857142
```

b) KNN Algorithm Loss:

In [9]:
```
import pandas as pd
data = pd.read_csv('abalone.csv',header=None)
data.columns=["Sex", "Length", "Diameter", "Height", "Whole weight", "shucked weight", "Viscera weight", "shell weight", "Rings"]
data = data.drop ("Sex", axis=1)
data.head()
```

Out[9]:

	Length	Diameter	Height	Whole weight	shucked weight	Viscera weight	shell weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [3]:
```
X=data.drop ("Rings", axis=1)
X=X.values
y=data["Rings"]
y=y.values
```

In [4]:
```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train, y_test=train_test_split (X, y, test_size=0.2, random_state=12345)
```

In [5]:
```
from sklearn.neighbors import KNeighborsRegressor
knn_model=KNeighborsRegressor(n_neighbors=7)
```

In [5]:
```
from sklearn.neighbors import KNeighborsRegressor
knn_model=KNeighborsRegressor(n_neighbors=7)
```


In [6]:
```
knn_model.fit(X_train, y_train)
```

Out[6]: `KNeighborsRegressor(n_neighbors=7)`

In [8]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
train_preds = knn_model.predict (X_train)
mse = mean_squared_error (y_train, train_preds)
rmse = sqrt(mse)
rmse
```

Out[8]: `1.9191665035671284`

Inference: KNN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

Result: KNN is implemented and visualized using Jupyter notebook and the required plots are shown.