CMP4336 – Introduction to Data Mining

Assignment II

Onur Guzel 1400369

Import required libraries

In [1]:
```
try:
    import os # accessing directory structure

    import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import seaborn as sns

    import warnings

    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import GaussianNB
    from sklearn.model_selection import KFold
    from sklearn.model_selection import cross_val_predict, cross_val_score
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    from sklearn.feature_selection import RFE
    from sklearn.linear_model import LogisticRegression


    import category_encoders as ce

    import statsmodels.api as sm


except ImportError as err:
    print("Some required files could not be found for program...",
          "\nPlease contact with the manufacturer!")

warnings.filterwarnings('ignore')
```

List datasets

In [2]:
```
for dirname, _, filenames in os.walk('..\Dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
..\Dataset\abalone_dataset.txt
```

Get data and give a name to all columns

In [3]:
```
df = pd.read_csv('../Dataset/abalone_dataset.txt', sep = "\t", header=None)

df.columns = ["Sex", "Length", "Diameter", "Height", "Whole weight", "Shucked weight", "Viscera weight", "Shell weight", "Rings"]
```

Print some information about dataset

In [4]:
```
number_of_data = df.shape[0]
number_of_attiribute = df.shape[1]

print("Number of data : ", number_of_data , "\n",
      "Number of attiribute : ", number_of_attiribute, sep = "")
```

```
Number of data : 4177
Number of attiribute : 9
```

Let's take a quick look at what the data looks like:

In [5]:
```
df
```

Out[5]:

	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	0.1500	3
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	1
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	2
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	2
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	1
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	2
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	2
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	2
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	2
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	3

4177 rows × 9 columns

Check for duplicated rows and missing values

In [6]:
```
if(df.duplicated().sum() == 0):
    print("No duplicated rows!")
else:
    print("Error! Duplicated row(s)!")

if(df.isnull().sum().sum() == 0):
    print("No missing value!")
else:
    print("Error! Missing Value(s)!")
    print("Number of missing value : ", df.isnull().sum().sum())
```

```
No duplicated rows!
No missing value!
```

Get information about Dtypes and Non-Null Counts

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

List categorical and numerical variables

In [8]:
```
# Find categorical variables
categorical = [var for var in df.columns if df[var].dtype=='object']

print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :\n\n', categorical)


# Find numerical variables
numerical = [var for var in df.columns if df[var].dtype!='object']

print('There are {} numerical variables\n'.format(len(numerical)))
print('The numerical variables are :', numerical)
```

```
There are 1 categorical variables

The categorical variables are :

 ['Sex']
There are 8 numerical variables

The numerical variables are : ['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight', 'Rings']
```

Calculate age column and drop Rings column

In [9]:
```
# Already applied in dataset
#df_age = pd.DataFrame(df["Rings"].apply(lambda x : 1 if x < 8 else (2 if x < 12 else 3)))

df_age = pd.DataFrame(df["Rings"])
df_age.columns = ["Age"]

# Drop Rings column
df.drop(["Rings"],axis=1,inplace=True)

# Convert M F I to 1 2 3 respectively for Sex column
df["Sex"] = df["Sex"].apply(lambda x : 1 if x == "M" else (2 if x == "F" else 3))

print(df.head())
print(df_age.head())
```

```
   Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
0    1   0.455     0.365   0.095        0.5140          0.2245
1    1   0.350     0.265   0.090        0.2255          0.0995
2    2   0.530     0.420   0.135        0.6770          0.2565
3    1   0.440     0.365   0.125        0.5160          0.2155
4    3   0.330     0.255   0.080        0.2050          0.0895

   Viscera weight  Shell weight
0          0.1010         0.150
1          0.0485         0.070
2          0.1415         0.210
3          0.1140         0.155
4          0.0395         0.055
   Age
0    3
1    1
2    2
3    2
4    1
```

Q1 Apply naive bayes classifier using all features as input, and

In [10]:
```
# Use all of features as input
X = df.values
y = df_age.values.ravel()

# Gaussian Naive Bayes Model
gnb = GaussianNB()
```

Q1.1 100 samples for training, and rest for validation set

In [11]:
```
# Build KFold Cross Validation Generator with 41 folds(Approximately 100 samples for training and the rest for validation)
kf = KFold(n_splits=41)

# Calculate cross validation scores
scores = cross_val_score(gnb, X, y, cv=kf)

# Generate cross-validated estimates for each input data point
y_pred = cross_val_predict(gnb, X, y, cv=kf)

# Build confusion matrix from predictions and the actual labels
conf_mat = confusion_matrix(y, y_pred)

# Print the confusion matrix, cross validation scores and the classification report
print("Confusion Matrix : \n", conf_mat,
      "\nCross Validation Scores : \n", scores,
      "\nClassification Report : \n", classification_report(y,y_pred))

# Print total misclassification errors
print("Total misclassification errors:")
classification_errors = conf_mat[0][1]  + conf_mat[0][2] + conf_mat[1][0] + conf_mat[1][2] + conf_mat[2][0] + conf_mat[2][1]
print(classification_errors)
```

```
Confusion Matrix :
 [[ 738   96    5]
 [ 472 1138  768]
 [  68  432  460]]
Cross Validation Scores :
 [0.62745098 0.6372549  0.58823529 0.64705882 0.50980392 0.37254902
 0.3627451  0.44117647 0.67647059 0.61764706 0.59803922 0.56862745
 0.69607843 0.47058824 0.57843137 0.7254902  0.48039216 0.52941176
 0.71568627 0.59803922 0.58823529 0.59803922 0.57843137 0.35294118
 0.48039216 0.62745098 0.53921569 0.6372549  0.61764706 0.60784314
 0.55882353 0.56862745 0.31372549 0.53921569 0.6372549  0.53921569
 0.4950495  0.57425743 0.47524752 0.55445545 0.6039604 ]
Classification Report :
               precision    recall  f1-score   support

           1       0.58      0.88      0.70       839
           2       0.68      0.48      0.56      2378
           3       0.37      0.48      0.42       960

    accuracy                           0.56      4177
   macro avg       0.54      0.61      0.56      4177
weighted avg       0.59      0.56      0.56      4177

Total misclassification errors:
1841
```

Q1.2 1000 samples for training, and rest for validation set

In [12]:
```
# Build KFold Cross Validation Generator with 4 folds(Approximately 1000 samples for training and the rest for validation)
kf = KFold(n_splits=4)

# Calculate cross validation scores
scores = cross_val_score(gnb, X, y, cv=kf)

# Generate cross-validated estimates for each input data point
y_pred = cross_val_predict(gnb, X, y, cv=kf)

# Build confusion matrix from predictions and the actual labels
conf_mat = confusion_matrix(y, y_pred)

# Print the confusion matrix, cross validation scores and the classification report
print("Confusion Matrix : \n", conf_mat,
      "\nCross Validation Scores : \n", scores,
      "\nClassification Report : \n", classification_report(y,y_pred))

# Print total misclassification errors
print("Total misclassification errors:")
classification_errors = conf_mat[0][1]  + conf_mat[0][2] + conf_mat[1][0] + conf_mat[1][2] + conf_mat[2][0] + conf_mat[2][1]
print(classification_errors)
```

```
Confusion Matrix :
 [[ 736   97    6]
 [ 476 1108  794]
 [  69  457  434]]
Cross Validation Scores :
 [0.52344498 0.55747126 0.56130268 0.53927203]
Classification Report :
               precision    recall  f1-score   support

           1       0.57      0.88      0.69       839
           2       0.67      0.47      0.55      2378
           3       0.35      0.45      0.40       960

    accuracy                           0.55      4177
   macro avg       0.53      0.60      0.55      4177
weighted avg       0.58      0.55      0.54      4177

Total misclassification errors:
1899
```

Q2 Apply bi-directional search feature selection algorithm to the dataset using Naive-Bayes as the baseline classification algorithm

In [13]:
```
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model, 3)

fit = rfe.fit(X, y)

print("Number of Selected Features: %d" % fit.n_features_)
```

```
Number of Selected Features: 3
```

Q2.1 Report the order of features selected by the algorithm.

In [14]:
```
print("Feature Ranking: %s" % fit.ranking_)

feature_number = 0
feature_list = []
for selected_feature in fit.support_:
    if selected_feature == True:
        feature_list.append(df.columns[feature_number])
    feature_number += 1

print(feature_list)
```

```
Feature Ranking: [6 2 3 4 1 1 5 1]
['Whole weight', 'Shucked weight', 'Shell weight']
```

Q2.2 Using top 3 selected features and 100 samples for training, apply naïve bayes classifier (the rest of the samples will be used for validation).

In [15]:
```
# Use top 3 selected features as input data
X = df[feature_list].values
y = df_age.values.ravel()

# Build Gaussian Naive Bayes Model
gnb = GaussianNB()

# Build KFold Cross Validation Generator with 41 folds(Approximately 100 samples for training and the rest for validation)
kf = KFold(n_splits=41)

# Calculate cross validation scores
scores = cross_val_score(gnb, X, y, cv=kf)

# Generate cross-validated estimates for each input data point
y_pred = cross_val_predict(gnb, X, y, cv=kf)

# Build confusion matrix from predictions and the actual labels
conf_mat = confusion_matrix(y, y_pred)

# Print the confusion matrix, cross validation scores and the classification report
print("Confusion Matrix : \n", conf_mat,
      "\nCross Validation Scores : \n", scores,
      "\nClassification Report : \n", classification_report(y,y_pred))

# Print total misclassification errors
print("Total misclassification errors:")
classification_errors = conf_mat[0][1]  + conf_mat[0][2] + conf_mat[1][0] + conf_mat[1][2] + conf_mat[2][0] + conf_mat[2][1]
print(classification_errors)
```

```
Confusion Matrix :
 [[ 736  102    1]
 [ 517 1531  330]
 [  75  593  292]]
Cross Validation Scores :
 [0.54901961 0.55882353 0.43137255 0.51960784 0.35294118 0.35294118
 0.32352941 0.34313725 0.79411765 0.7745098  0.71568627 0.75490196
 0.70588235 0.68627451 0.73529412 0.7254902  0.7745098  0.6372549
 0.80392157 0.71568627 0.60784314 0.52941176 0.44117647 0.33333333
 0.51960784 0.76470588 0.71568627 0.85294118 0.78431373 0.70588235
 0.66666667 0.48039216 0.2254902  0.59803922 0.70588235 0.70588235
 0.69306931 0.65346535 0.43564356 0.71287129 0.73267327]
Classification Report :
               precision    recall  f1-score   support

           1       0.55      0.88      0.68       839
           2       0.69      0.64      0.67      2378
           3       0.47      0.30      0.37       960

    accuracy                           0.61      4177
   macro avg       0.57      0.61      0.57      4177
weighted avg       0.61      0.61      0.60      4177

Total misclassification errors:
1618
```

Q2.3 Using top 3 selected features and 1000 samples for training, apply naïve bayes classifier (the rest of the samples will be used for validation).

In [16]:
```
# Use top 3 selected features as input data
X = df[feature_list].values
y = df_age.values.ravel()

# Build Gaussian Naive Bayes Model
gnb = GaussianNB()

# Build KFold Cross Validation Generator with 4 folds(Approximately 1000 samples for training and the rest for validation)
kf = KFold(n_splits=4)

# Calculate cross validation scores
scores = cross_val_score(gnb, X, y, cv=kf)

# Generate cross-validated estimates for each input data point
y_pred = cross_val_predict(gnb, X, y, cv=kf)

# Build confusion matrix from predictions and the actual labels
conf_mat = confusion_matrix(y, y_pred)

# Print the confusion matrix, cross validation scores and the classification report
print("Confusion Matrix : \n", conf_mat,
      "\nCross Validation Scores : \n", scores,
      "\nClassification Report : \n", classification_report(y,y_pred))

# Print total misclassification errors
print("Total misclassification errors:")
classification_errors = conf_mat[0][1]  + conf_mat[0][2] + conf_mat[1][0] + conf_mat[1][2] + conf_mat[2][0] + conf_mat[2][1]
print(classification_errors)
```

```
Confusion Matrix :
 [[ 734  103    2]
 [ 510 1503  365]
 [  76  600  284]]
Cross Validation Scores :
 [0.48995215 0.69252874 0.6302682  0.60153257]
Classification Report :
               precision    recall  f1-score   support

           1       0.56      0.87      0.68       839
           2       0.68      0.63      0.66      2378
           3       0.44      0.30      0.35       960

    accuracy                           0.60      4177
   macro avg       0.56      0.60      0.56      4177
weighted avg       0.60      0.60      0.59      4177

Total misclassification errors:
1656
```

Conclusion

The model that uses top 3 selected features which are 'Whole weight', 'Shucked weight' and 'Shell weight' as input and 41 folds for cross validation provides the highest accuracy levels (0.61). This indicates that these 3 parameters are more related to the age of the abalone than all of the inputs combined. Also it was expected to get better results with 41 folds than 4 folds as the model is able to generalize better without overfitting.

To summarize briefly advantages and disadvantages of Naive Bayes:

```
Advantages :
    - Simple and easy to apply.
    - It can be used with continuous and discrete data.
    - It can also be used in an unbalanced data set.
    - It can be used in real time systems due to its speed.

Disadvantages
    -Relationships between variables cannot be modeled because operations are carried out by assuming that properties are independent from each other.
    - You may be faced with the Zero Probability problem.
```