Abalone Age PredictionData Description

Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

Attribute Information:
Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description
Sex / nominal / -- / M, F, and I (infant)

Length / continuous / mm / Longest shell measurement

Diameter / continuous / mm / perpendicular to length

Height / continuous / mm / with meat in shell

Whole weight / continuous / grams / whole abalone

Shucked weight / continuous / grams / weight of meat

Viscera weight / continuous / grams / gut weight (after bleeding)

Shell weight / continuous / grams / after being dried

Rings / integer / -- / +1.5 gives the age in years
IN(1):

```
import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns.
```

IN(2):

```
df = pd.read_csv('../input/abalone.csv')
```

IN(3):
df.head().

OUT(3):

	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

IN(4):
df.describe()

OUT(4):

	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

IN(5):
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
EDA
sns.heatmap(df.isnull())
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc468da358>

sns.pairplot(df)
<seaborn.axisgrid.PairGrid at 0x7fcc3caa8160>

df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
```

```
Data columns (total 9 columns):
Sex               4177 non-null object
Length            4177 non-null float64
Diameter          4177 non-null float64
Height            4177 non-null float64
Whole weight      4177 non-null float64
Shucked weight    4177 non-null float64
Viscera weight    4177 non-null float64
Shell weight      4177 non-null float64
age               4177 non-null float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
numerical_features
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
categorical_features
Index(['Sex'], dtype='object')
plt.figure(figsize = (20,7))
sns.heatmap(df[numerical_features].corr(),annot = True)
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc29714dd8>

Whole Weight is almost linearly varying with all other features except age
Heigh has least linearity with remaining features
Age is most linearly proprtional with Shell Weight followed by Diameter and length
Age is least correlated with Shucked Weight
Key insight:

All numerical features but 'sex'
  - Though features are not normaly distributed, are close to normality
  - None of the features have minimum = 0 except Height (requires re-check)
  - Each feature has difference scale range
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc26ba6748>

plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'age', data = df, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'age',data = df)
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc26b67b38>
```

Male : age majority lies in between 7.5 years to 19 years
Female: age majority lies in between 8 years to 19 years
Immature: age majority lies in between 6 years to < 10 years
Data Preprocessing

```
# outlier handling
df = pd.get_dummies(df)
dummy_df = df
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Viscera weight'] > 0.5) &
        (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<0.5) & (
df['age'] > 25)].index, inplace = True)
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Shell weight'] > 0.6) &
        (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Shell weight']<0.8) & (
df['age'] > 25)].index, inplace = True)
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Shucked weight'] >= 1) &
        (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<1) & (
df['age'] > 20)].index, inplace = True)
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Whole weight'] >= 2.5) &
        (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['age'] > 25)].index, inplace = True)
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Diameter'] <0.1) &
        (df['age'] < 5)].index, inplace = True)
```

```
df.drop(df[(df['Diameter']<0.6) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & (
df['age'] < 25)].index, inplace = True)
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Height'] > 0.4) &
         (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Length'] <0.1) &
         (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Length']<0.8) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Length']>=0.8) & (
df['age'] < 25)].index, inplace = True)
Feature Selection and Standardization
X = df.drop('age', axis = 1)
y = df['age']
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645:
DataConversionWarning: Data with input dtype uint8, float64 were all converted to float64 by
StandardScaler.
  return self.partial_fit(X, y)
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:464:  DataConversionWarning:  Data
with input dtype uint8, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
Model Selection
```

1)Linear regression

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
 from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)

p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
Mean Squared error of training set :3.551893
Mean Squared error of testing set :3.577687
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)

p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)
R2 Score of training set:0.54
R2 Score of testing set:0.53
```

2)Ridge

```
from sklearn.linear_model import Ridge
ridge_mod = Ridge(alpha=0.01, normalize=True)
ridge_mod.fit(X_train, y_train)
ridge_mod.fit(X_test, y_test)
ridge_model_pred = ridge_mod.predict(X_test)
ridge_mod.score(X_train, y_train)
0.5307346478347332
ridge_mod.score(X_test, y_test)
0.5272608729607438
plt.scatter(y_test, ridge_model_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
Text(0, 0.5, 'Predictions')
```

3)Support vector Regression

```
from sklearn.svm import SVR
# LINEAR KERNEL

svr = SVR(kernel = 'linear')
svr.fit(X_train, y_train)
```

```
svr.fit(X_test, y_test)
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
  gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
  tol=0.001, verbose=False)
y_train_pred = svr.predict(X_train)
y_test_pred = svr.predict(X_test)

svr.score(X_train, y_train)
0.4461014542389635
svr.score(X_test, y_test)
0.43681391121982105
4) RandomForestRegression
 from sklearn.ensemble import RandomForestRegressor
 regr = RandomForestRegressor(max_depth=2, random_state=0,
                              n_estimators=100)
regr.fit(X_train, y_train)
regr.fit(X_test, y_test)
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
y_train_pred = regr.predict(X_train)
y_test_pred = regr.predict(X_test)

regr.score(X_train, y_train)
0.4287379777803546
regr.score(X_test, y_test)
0.43753106247261264
5)Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
gbr.fit(X_test, y_test)
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, n_iter_no_change=None, presort='auto',
             random_state=None, subsample=1.0, tol=0.0001,
             validation_fraction=0.1, verbose=0, warm_start=False)
y_train_pred = regr.predict(X_train)
y_test_pred = regr.predict(X_test)
```

```
regr.score(X_train, y_train)
0.4287379777803546
regr.score(X_test, y_test)
0.43753106247261264
```
6)KNeighborsRegressor
```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors =4 )
knn.fit(X_train, y_train)
knn.fit(X_test, y_test)
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=4, p=2,
          weights='uniform')
y_train_pred = knn.predict(X_train)
y_test_pred = knn.predict(X_test)


knn.score(X_train, y_train)
0.4677575709446231
knn.score(X_test, y_test)
0.6856343678141352
```
you have seen the performance of each one of above model.

so according to you which model should we start or choose?

"Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better. Another way of saying it is that the more assumptions you have to make, the more unlikely an explanation." Hence, starting with the simplest model Ridge, for various reasons:

- Feature Dimension is less
- No misisng values
- Few categorical features

Hyperparameter Tunning Using GridSearchCV
```
# Hyperparameter Tuning using GridSearchCV

from sklearn.model_selection import  GridSearchCV
param  = {'alpha':[0.01, 0.1, 1,10,100],
         'solver' : ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']}
glrm0 = GridSearchCV(estimator = Ridge(random_state=10,),
param_grid = param,scoring= 'r2' ,cv = 5,  n_jobs = -1)
glrm0.fit(X_train, y_train)
glrm0.best_params_, glrm0.best_score_
({'alpha': 0.1, 'solver': 'sag'}, 0.5308712206007367)
ridge_mod = Ridge(alpha=0.001,solver = 'sag', random_state = 10, normalize=True)
ridge_mod.fit(X_train, y_train)
ridge_mod.fit(X_test, y_test)
```

ridge_model_pred = ridge_mod.predict(X_test)
ridge_mod.score(X_train, y_train)
0.5331463016536355
ridge_mod.score(X_test, y_test)
0.534651599310652
After hyperparameter tuning, CV score has improve slightly while, R2_Score has decreased showing base model was overfit.

we use tuning on different different model.

Exploratory Analysis Using Univariate, Bivariate, and Multivariate Analysis Techniques. This article was published as a part of the Data Science Blogathon.

Introduction

Data is everywhere around us, in spreadsheets, on various social media platforms, in survey forms, and more. The process of cleaning, transforming, interpreting, analyzing, and visualizing this data to extract useful information and gain valuable insights to make more effective business decisions is called Data Analysis.

Data Analysis can be organized into 6 types

- Exploratory Analysis
- Descriptive Analysis
- Inferential Analysis
- Predictive Analysis
- Causal Analysis
- Mechanistic Analysis

Here, we will dive deep into Exploratory Analysis,

Exploratory Analysis

The preliminary analysis of data to discover relationships between measures in the data and to gain an insight on the trends, patterns, and relationships among various entities present in the data set with the help of statistics and visualization tools is called Exploratory Data Analysis (EDA).

Exploratory data analysis is cross-classified in two different ways where each method is either graphical or non-graphical. And then, each method is either univariate, bivariate or multivariate.

Univariate Analysis

Uni means one and variate means variable, so in univariate analysis, there is only one dependable variable. The objective of univariate analysis is to derive the data, define and summarize it, and analyze the pattern present in it. In a dataset, it explores each variable separately. It is possible for two kinds of variables- Categorical and Numerical.

Some patterns that can be easily identified with univariate analysis are Central Tendency (mean, mode and median), Dispersion (range, variance), Quartiles (interquartile range), and Standard deviation.

Univariate data can be described through:

Ø Frequency Distribution Tables

The frequency distribution table reflects how often an occurrence has taken place in the data. It gives a brief idea of the data and makes it easier to find patterns.

Example:

The list of IQ scores is: 118, 139, 124, 125, 127, 128, 129, 130, 130, 133, 136, 138, 141, 142, 149, 130, 154.

IQ Range	Number
118-125	3
126-133	7
134-141	4
142-149	2
150-157	1

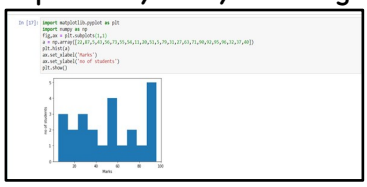
Ø Bar Charts

The bar graph is very convenient while comparing categories of data or different groups of data. It helps to track changes over time. It is best for visualizing discrete data.

Ø Histograms

Histograms are similar to bar charts and display the same categorical variables against the category of data. Histograms display these categories as bins which indicate the number of data points in a range. It is best for visualizing continuous data.

exploratory analysis histogram



Ø Pie Charts

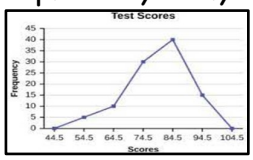
Pie charts are mainly used to comprehend how a group is broken down into smaller pieces. The whole pie represents 100 percent, and the slices denote the relative size of that particular category.

exploratory analysis pie chart

Ø Frequency Polygons

Similar to histograms, a frequency polygon is used for comparing datasets or displaying the cumulative frequency distribution.

exploratory analysis - frequency plit



Bivariate Analysis

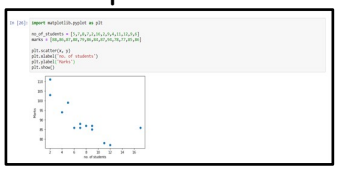
Bi means two and variate means variable, so here there are two variables. The analysis is related to cause and the relationship between the two variables. There are three types of bivariate analysis.

Bivariate Analysis of two Numerical Variables (Numerical-Numerical)

Ø Scatter Plot

A scatter plot represents individual pieces of data using dots. These plots make it easier to see if two variables are related to each other. The resulting pattern indicates the type (linear or non-linear) and strength of the relationship between two variables.

scatter plot



Ø Linear Correlation

Linear Correlation represents the strength of a linear relationship between two numerical variables. If there is no correlation between the two variables, there is no tendency to change along with the values of the second quantity.

$$r = \frac{Covar(x,y)}{\sqrt{Var(x)Var(y)}}$$

$$Covar(x,y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{n}$$

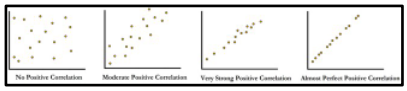
$$Var(x) = \frac{\sum(x-\bar{x})^2}{n}$$

$$Var(y) = \frac{\sum(y-\bar{y})^2}{n}$$

r : Linear Correlation
$Covar$: Covariance
Var : Variance

linear correlation

Here, r measures the strength of a linear relationship and is always between -1 and 1 where -1 denotes perfect negative linear correlation and +1 denotes perfect positive linear correlation and zero denotes no linear correlation



Bivariate Analysis of two categorical Variables (Categorical-Categorical)

Ø Chi-square Test

The chi-square test is used for determining the association between categorical variables. It is calculated based on the difference between expected frequencies and the observed frequencies in one or more categories of the frequency table. A probability of zero indicates a complete dependency between two categorical variables and a probability of one indicates that two categorical variables are completely independent.

Here, subscript c indicates the degrees of freedom, O indicates observed value, and E indicates expected value.

chi-square

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Bivariate Analysis of one numerical and one categorical variable (Numerical-Categorical)

Ø Z-test and t-test

Z and T-tests are important to calculate if the difference between a sample and population is substantial.

z-test

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

$\bar{x}$ = sample mean
μ = population mean
σ =population standard deviation
n =sample size

If the probability of Z is small, the difference between the two averages is more significant.

T-Test

$$t = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad \text{where} \quad s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

where

μ = Proposed constant for the population mean
$\bar{x}$ = Sample mean
n = Sample size (i.e., number of observations)
s = Sample standard deviation
$s_{\bar{x}}$ = Estimated standard error of the mean (s/sqrt(n))

If the sample size is large enough, then we use a Z-test, and for a small sample size, we use a T-test.

Ø ANALYSIS OF VARIANCE (ANOVA)

The ANOVA test is used to determine whether there is a significant difference among the averages of more than two groups that are statistically different from each other. This analysis is appropriate for comparing the averages of a numerical variable for more than two categories of a categorical variable.

anova

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Squares (MS)	F
Within	$SSW = \sum_{j=1}^{k}\sum_{i=1}^{n_j}(x-\bar{x}_j)^2$	$df_w = k-1$	$MSW = \frac{SSW}{df_w}$	$F = \frac{MSB}{MSW}$
Between	$SSB = \sum_{j=1}^{k}(\bar{x}_j-\bar{x})^2$	$df_b = n-k$	$MSB = \frac{SSB}{df_b}$	
Total	$SST = \sum_{j=1}^{k}(\bar{x}_j-\bar{x})^2$	$df_t = n-1$		

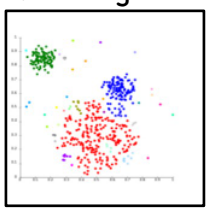
Multivariate Analysis

Multivariate analysis is required when more than two variables have to be analyzed simultaneously. It is a tremendously hard task for the human brain to visualize a relationship among 4 variables in a graph and thus multivariate analysis is used to study more complex sets of data. Types of Multivariate Analysis include Cluster Analysis, Factor Analysis, Multiple Regression Analysis, Principal Component Analysis, etc. More than 20 different ways to perform multivariate analysis exist and which one to choose depends upon the type of data and the end goal to achieve. The most common ways are:

Ø Cluster Analysis

Cluster Analysis classifies different objects into clusters in a way that the similarity between two objects from the same group is maximum and minimal otherwise. It is used when rows and columns of the data table represent the same units and the measure represents distance or similarity.

clustering

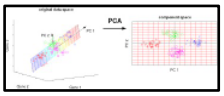


Ø Principal Component Analysis (PCA)

Principal Components Analysis (or PCA) is used for reducing the dimensionality of a data table with a large number of interrelated measures. Here, the original variables are converted into a new set of variables, which are known as the "Principal Components" of Principal Component Analysis.

PCA is used for the dataset that shows multicollinearity. Although least squares estimates are biased, the distance between variances and their actual value can be really large. So, PCA adds some bias and reduces standard error for the regression model.

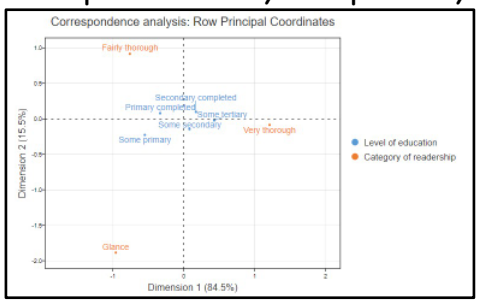
PCA



Ø Correspondence Analysis

Correspondence Analysis using the data from a contingency table shows relative relationships between and among two different groups of variables. A contingency table is a 2D table with rows and columns as groups of variables.

correspondence analysis Exploratory Analysis



Conclusion

I hope you now have a better understanding of various techniques used in Univariate, Bivariate, and Multivariate Analysis.