```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import  confusion_matrix,accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import  skew
dataset=pd.read_csv(r"E:\prg files\ibm\abalone.csv")
dataset.head()
dataset.info()
dataset.describe()
dataset.shape
dataset['Age']=dataset['Rings']+1.5
dataset=dataset.drop('Rings',axis=1)
dataset.head()
dataset.describe()
dataset.hist(figsize=(20,10), grid=False, layout=(2,5), bins=30)
plt.show()
Numerical = dataset.select_dtypes(include=[np.number]).columns
Categorical = dataset.select_dtypes(include=[np.object]).columns
kew_values = skew(dataset[Numerical], nan_policy = 'omit')
dummy = pd.concat([pd.DataFrame(list(Numerical), columns=['Features']),
pd.DataFrame(list(skew_values), columns=['Skewness degree'])], axis = 1)
dummy.sort_values(by = 'Skewness degree' , ascending = False)
dataset.boxplot(figsize=(20,20))
plt.show()
plt.figure(figsize=(20,10))
sns.distplot(dataset['Age'])
plt.show()
dataset.head()
fig,axes=plt.subplots(4,2, figsize=(20,20))
axes=axes.flatten()
for i  in range(1,len(dataset.columns)-1):
    sns.scatterplot(x=dataset.iloc[:,i],y=dataset['Age'],ax=axes[i])
plt.show()
plt.figure(figsize=(10,10))
sns.boxenplot(y=dataset['Age'], x=dataset['Sex'])
plt.grid()
plt.show()
dataset.groupby('Sex')['Age'].describe()
plt.figure(figsize=(15,10))
sns.heatmap(dataset.corr(),annot=True)
plt.show()
sns.pairplot(dataset)
plt.show()
dataset.mean()
dataset.mode()
dataset.median()
dataset.isna()
dataset.isna().any()
dataset.skew()
print(sns.distplot(dataset['Age']))
dataset.kurt()
dataset.var()
dataset.std()
sns.boxplot(dataset['Length'])
qnt=dataset.quantile(q=(0.30,0.45))
qnt
```

```
iqr=qnt.loc[0.45]-qnt.loc[0.30]
iqr
lower=qnt.loc[0.30]-1.5*iqr
lower
upper=qnt.loc[0.45]+1.5*iqr
upper
dataset['Length']=np.where(dataset['Length']>45,31,dataset['Length'])
sns.boxplot(dataset['Length'])
dataset=pd.read_csv(r"E:\prg files\ibm\Harilone.csv")
dataset.head()
dataset['Sex'].replace( {'F':1,'M':0},inplace=True)
dataset.head()
y=dataset['Height']
y.head()
x=dataset.drop(columns=['Height'],axis=1)
x.head()
dataset=pd.get_dummies(dataset,columns=['Height'])
dataset.head()
#encoding
dataset = pd.get_dummies(dataset, drop_first=True)
dataset.head()
from sklearn.preprocessing import scale
scalex=scale(x)
x
x.mean()
x.std()
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
x_train.shape
x_test.shape
y_train.shape
y_test.shape
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
dataset=pd.get_dummies(dataset,drop_first=True)
dataset.head()
X  = dataset.drop('Height_1.13',axis=1)
y = dataset['Height_1.13']

from sklearn.model_selection import train_test_split
X_train, X_test, y_test, y_test = train_test_split(X, y, test_size=0.33)

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

X_trains = ss.fit_transform(X_train)
X_tests = ss.transform(X_test)
#Base model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_trains, y_train)
pred = lr.predict(X_tests)

from sklearn.metrics import r2_score, roc_auc_score, mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, pred))
r2 = r2_score(y_test, pred)
```

```
print("The root mean Sq error calculated from the base model is:",rmse)
print("The r2-score is:",r2)
The root mean Sq error calculated from the base model
is:0.026928853702142326 The r2-score is: 0.0

#selecting best feautre
from sklearn.feature_selection import RFE
lr = LinearRegression()
n = [{'n_features_to_select':list(range(1,10))}]
rfe = RFE(lr)

from sklearn.model_selection import GridSearchCV
gsearch = GridSearchCV(rfe, param_grid=n, cv=3)
gsearch.fit(X, y)

gsearch.best_params_
lr=LinearRegression()
rf = RFE(lr,n_features_to_select=8)
rf.fit(X,y)
pd.DataFrame(rf.ranking_,  index=X.columns, columns=['Class'])

from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_predict
from sklearn import model_selection

models=[ SVR(), RandomForestRegressor(), GradientBoostingRegressor(),
KNeighborsRegressor(n_neighbors = 4)]
results = []
names=['SVM','RF','GB','K-NN']
for model,names in zip(models,names):
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train,
cv=kfold)
    rmse = np.sqrt(mean_squared_error(y, cross_val_predict(model, X , y,
cv=3)))
    results.append(rmse)
    names.append(name)
    m = "%s: %f" % (name, rmse)
    print(m)
```