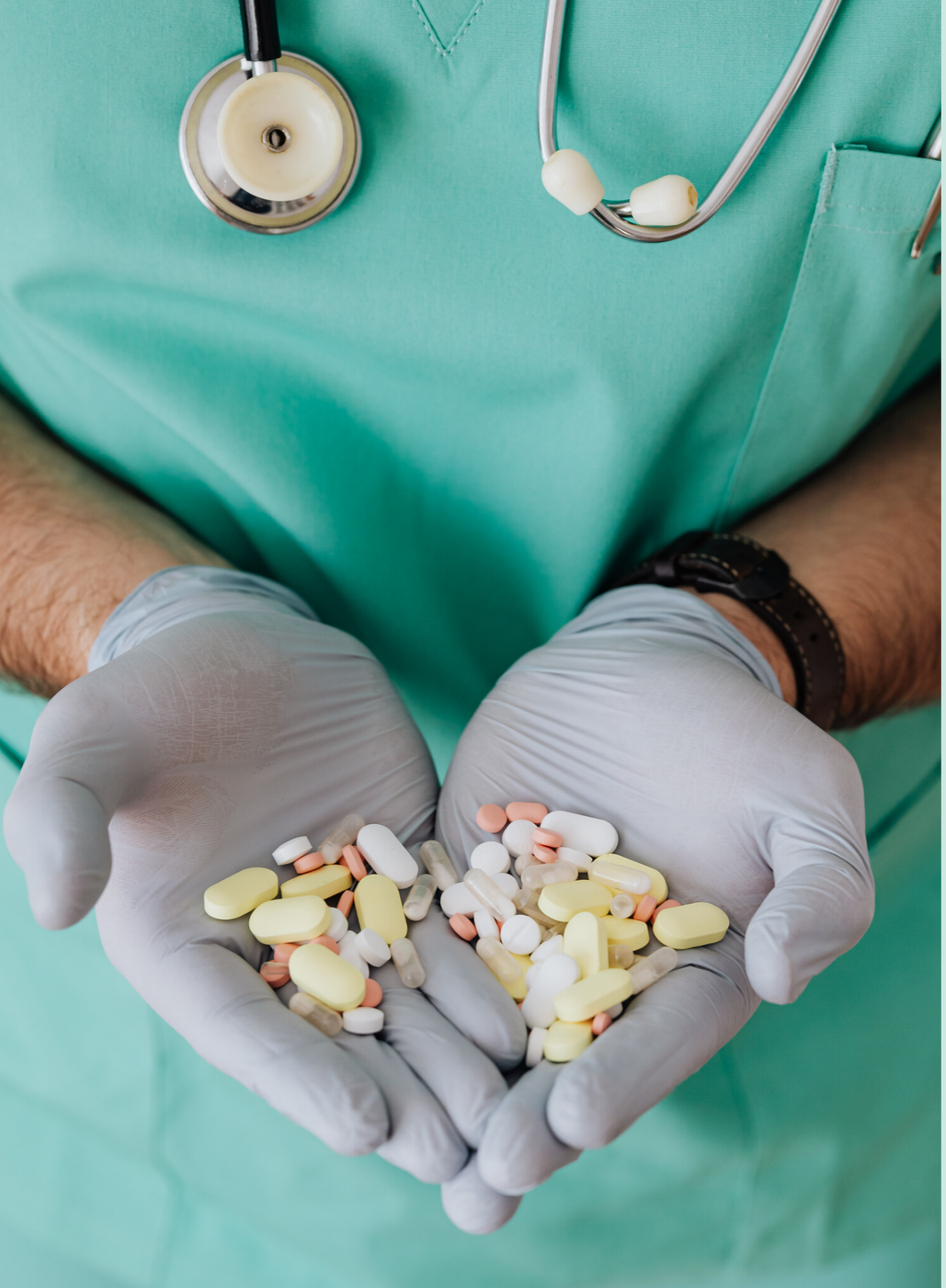


Life Expectancy Prediction

Presented by

Shlok Phulkar

GitHub- https://github.com/ShlokPhulkar

Linkdin- https://www.linkedin.com/feed/

Shlok Phulkar

Aspiring Data and Business Analyst

Education

BE Chemical Engineering

Post Graduaction
Diploma is Data Analyst
and Machine Learning

Skills

Pyhton Programming

Statistics

SQL

Machine Learning

Power BI

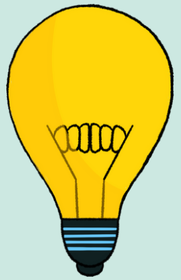
Conatct

9146560434

shlok33phulkar@gmail.com

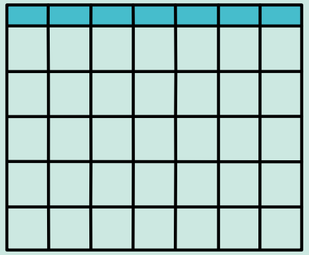
Index

1) Introduction to dataset
2) EDA
3) Missing values treatment
4) Outliers detection and treatment
5) Standardization of data
6) Implementaion of ML algorithms
7) Performance Parameters
8) Result
9) Conclusion



Introduction to dataset

Data points 58760

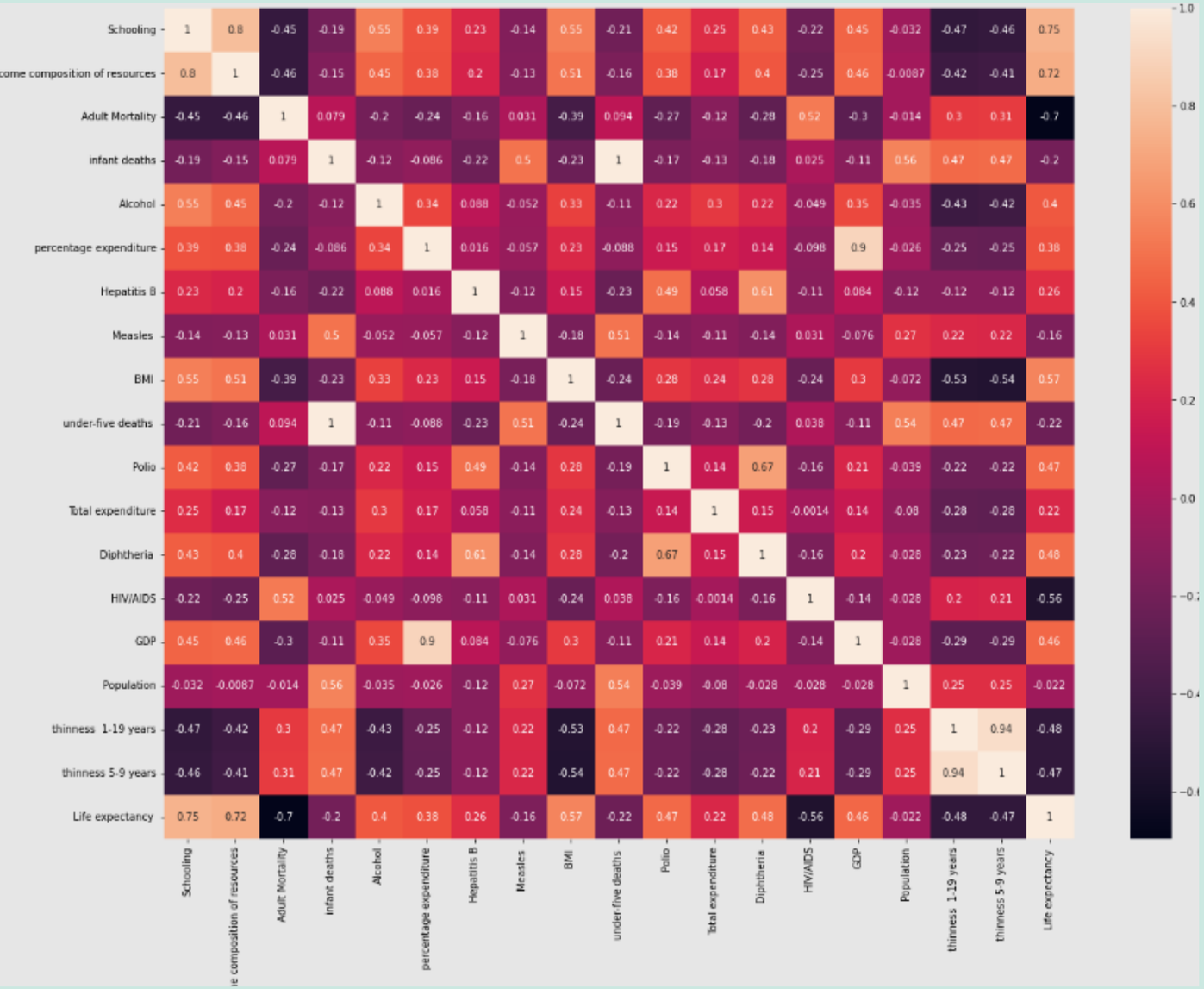


Shape 2938, 20

Columns are Schooling, Status, Income composition of resources Adult Mortality, infant deaths, Alcohol, percentage expenditure, Hepatitis B, Measles , BMI , under-five deaths , Polio, Total expenditure, Diphtheria ,HIV/AIDS, GDP, Population, thinness 1-19 years, thinness 5-9 years and Life expectancy is target column

EDA

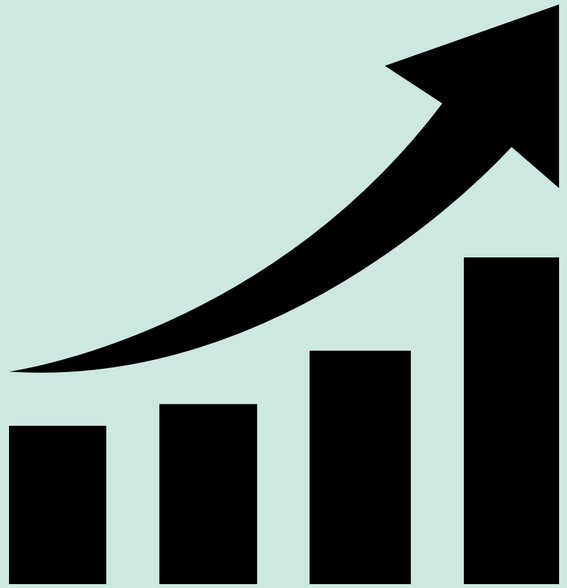
Correlation



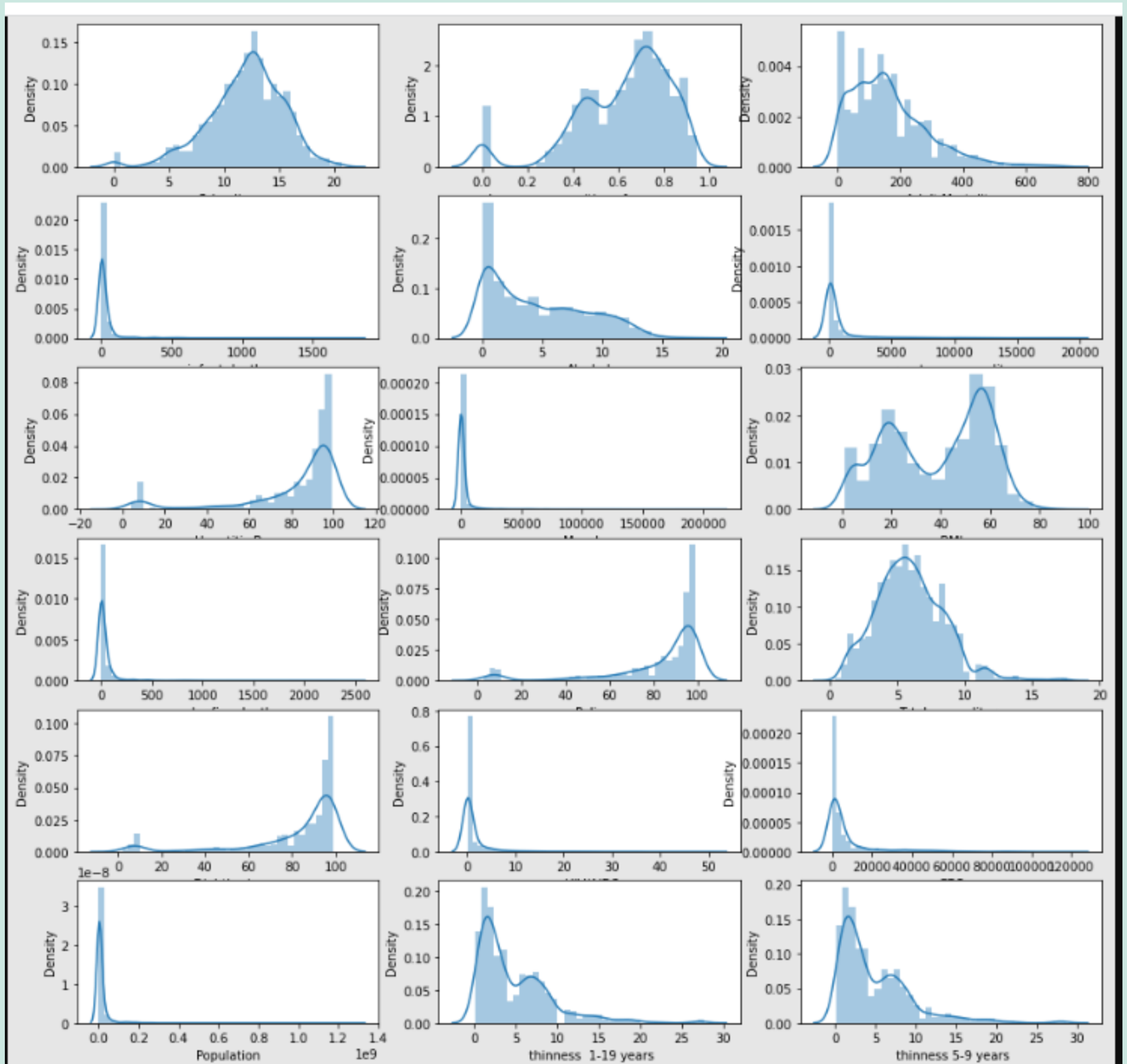
The target col has highest corr with schooling and income composition.

EDA Skewness

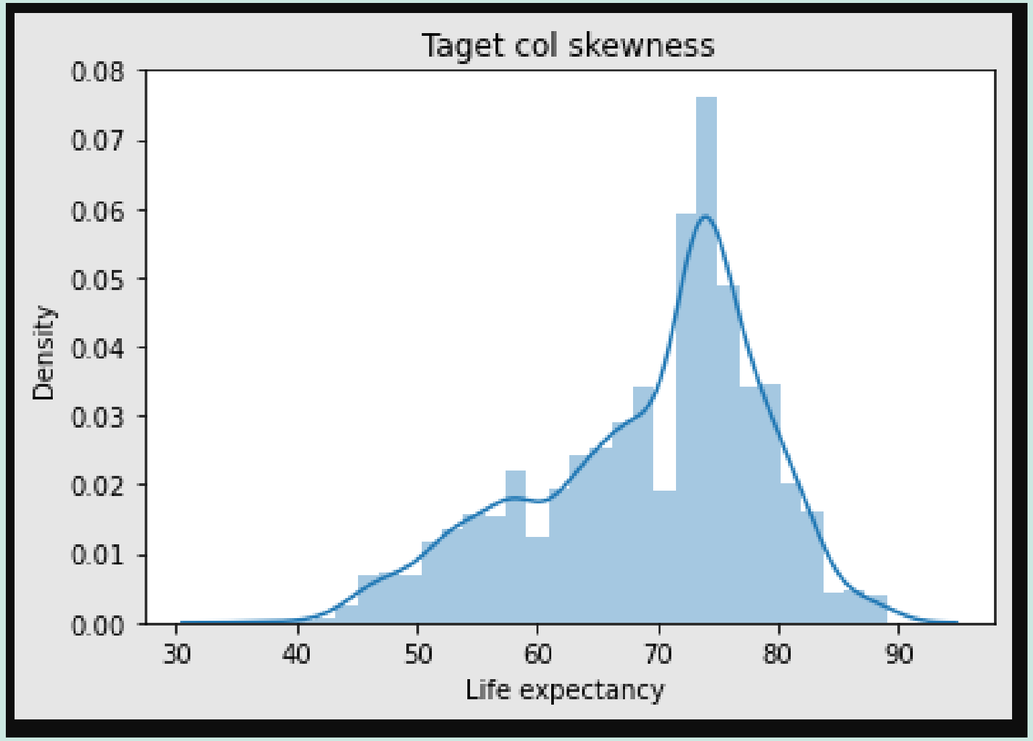
chooling	-0.602437
Income composition of resources	-1.143763
Adult Mortality	1.174369
infant deaths	9.786963
Alcohol	0.589563
percentage expenditure	4.652051
Hepatitis B	-1.930845
Measles	9.441332
BMI	-0.219312
under-five deaths	9.495065
Polio	-2.098053
Total expenditure	0.618686
Diphtheria	-2.072753
HIV/AIDS	5.396112
GDP	3.206655
Population	15.916236
thinness 1-19 years	1.711471
thinness 5-9 years	1.777424
Life expectancy	-0.638605



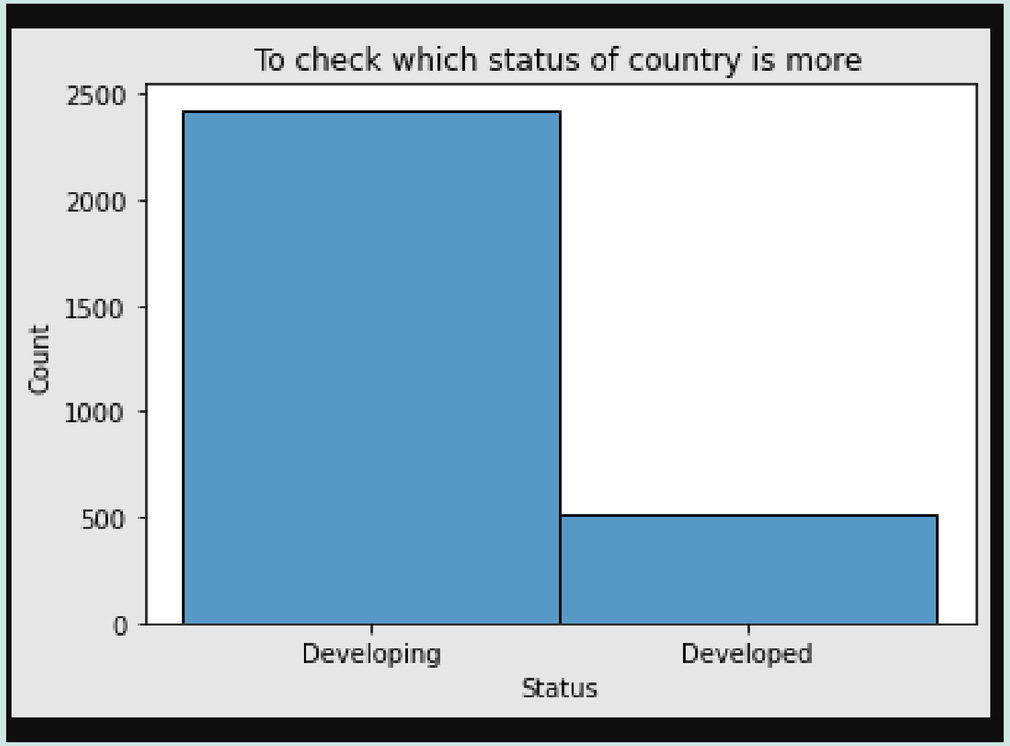
EDA Visualization of all independent



EDA Target column(Skewness)



EDA Check of country and development



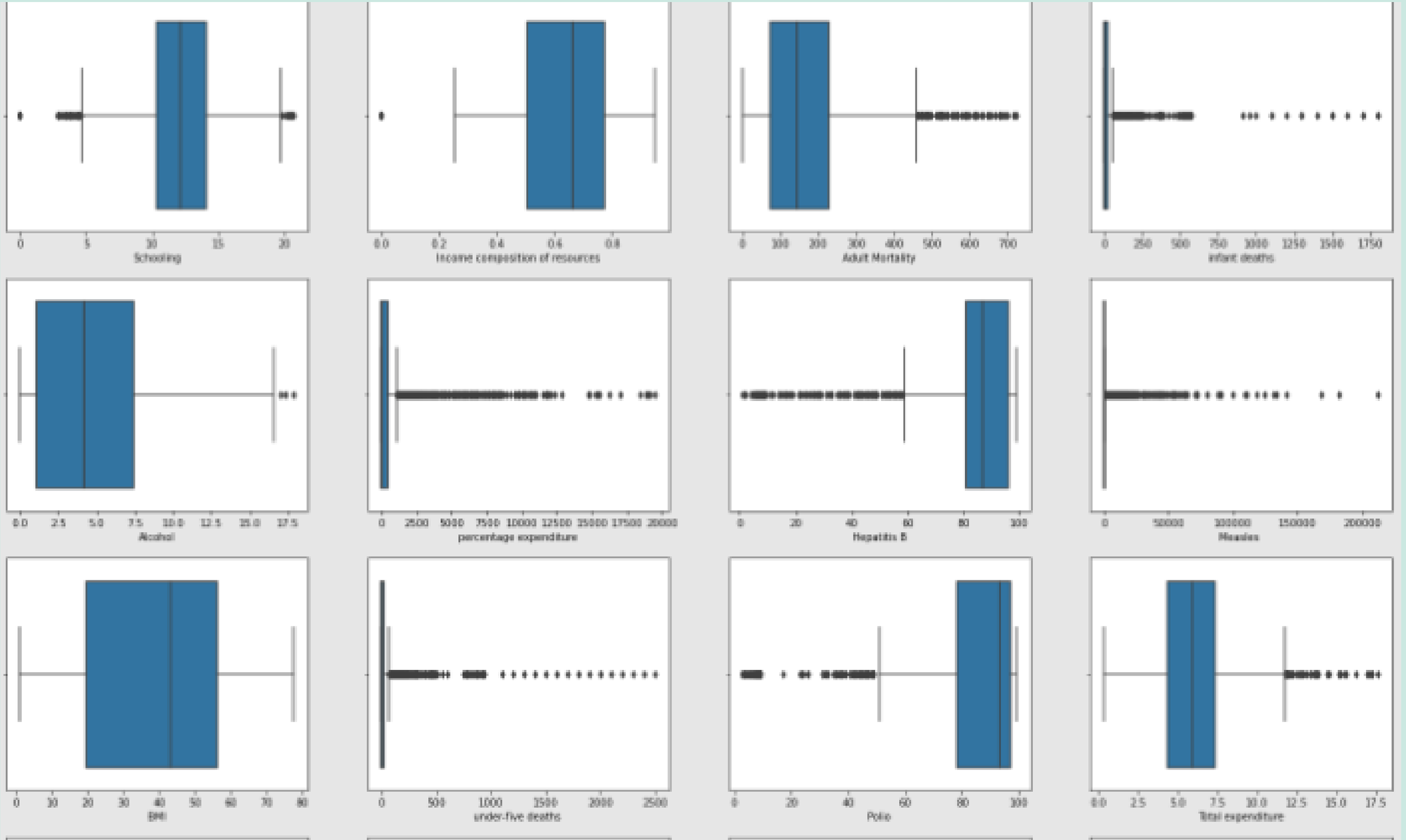
Check of missing values

Schooling	163
Status	0
Income composition of resources	167
Adult Mortality	10
infant deaths	0
Alcohol	194
percentage expenditure	0
Hepatitis B	553
Measles	0
BMI	34
under-five deaths	0
Polio	19
Total expenditure	226
Diphtheria	19
HIV/AIDS	0
GDP	448
Population	652
thinness 1-19 years	34
thinness 5-9 years	34
Life expectancy	10

Missing values Treatement

Schooling	0
Status	0
Income composition of resources	0
Adult Mortality	0
infant deaths	0
Alcohol	0
percentage expenditure	0
Hepatitis B	0
Measles	0
BMI	0
under-five deaths	0
Polio	0
Total expenditure	0
Diphtheria	0
HIV/AIDS	0
GDP	0
Population	0
thinness 1-19 years	0
thinness 5-9 years	0
Life expectancy	0

Check of outliers



Treatment of missing values

Standardization of Data

For each feature, the Standard Scaler scales the values such that the mean is 0 and the standard deviation is 1(or the variance)

x_scaled=x-mean/std_dev

Standard Scaler assumes that the distribution of the variable is normal

```
#Data standardization
X=df.drop("Life expectancy ",axis=1)
y=df["Life expectancy "]
ss=StandardScaler()
ScaledX=ss.fit_transform(X)
ScaledX
```

Implementation of ML algorithm

1) Multiple Regression
2) KNN
3) SVM
4) Decision tree
5) Random Forest
6) Bagging Regressor
7) Extra Trees Regression
8) Adaboost
9) Gradient Boost
10) XgBoosting

Multiple Regression

```
#Algoritm
MR=LinearRegression()
#Fit
MR.fit(X_train,y_train)
```

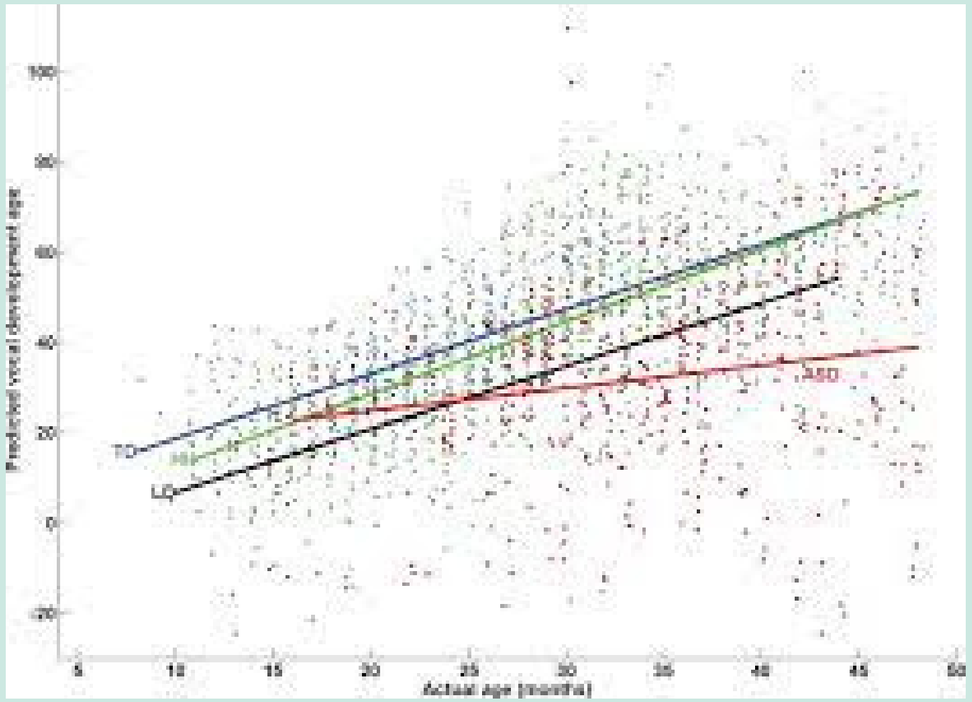
```
LinearRegression()
```

```
#Training and testing score
print("Training score is",MR.score(X_train,y_train))
print("Testing score is",MR.score(X_test,y_test))
```

```
Training score is 0.8513637008120423
Testing score is 0.858112607994263 4
```

```
#r2 score
predict_MR=MR.predict(X_test)
print("r2 score is",r2_score(y_test,predict_MR))
```

```
r2 score is 0.858112607994263 4
```



KNN

```
#Algorith
KNN=KNeighborsRegressor()
#fit
KNN.fit(X_train,y_train)
```

```
KNeighborsRegressor()
```

```
#Training and testing score
print("Training score is",KNN.score(X_train,y_train))
print("Testing score is",KNN.score(X_test,y_test))
```

```
Training score is 0.936510152932354
Testing score is 0.913087271863986
```

Concluion=As expected the traing and testing score for KNN is very good not not of a

```
#r2 score
predict_KNN=KNN.predict(X_test)
print("r2 score is",r2_score(y_test,predict_KNN))
```

SVM

```
#Algoritm
SVM=SVR()
#Fit
SVM.fit(X_train,y_train)
```

```
SVR()
```

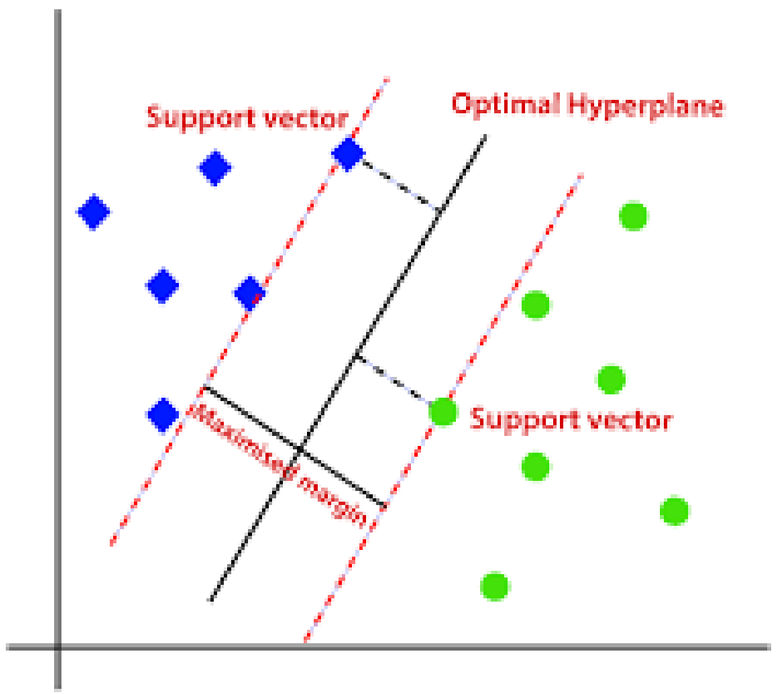
```
#Training and testing score
print("Training score is",SVM.score(X_train,y_train))
print("Testing score is",SVM.score(X_test,y_test))
```

```
Training score is 0.8830891709303101
Testing score is 0.8856348959623273
```

Concluion=The traing and testing score is good.

```
#r2 score
predict_SVM=SVM.predict(X_test)
print("r2 score is",r2_score(y_test,predict_SVM))
```

```
r2 score is 0.8856348959623273
```



Decision Tree

```
#Algoritm
DT=DecisionTreeRegressor()
#Fit
DT.fit(X_train,y_train)
```

```
DecisionTreeRegressor()
```

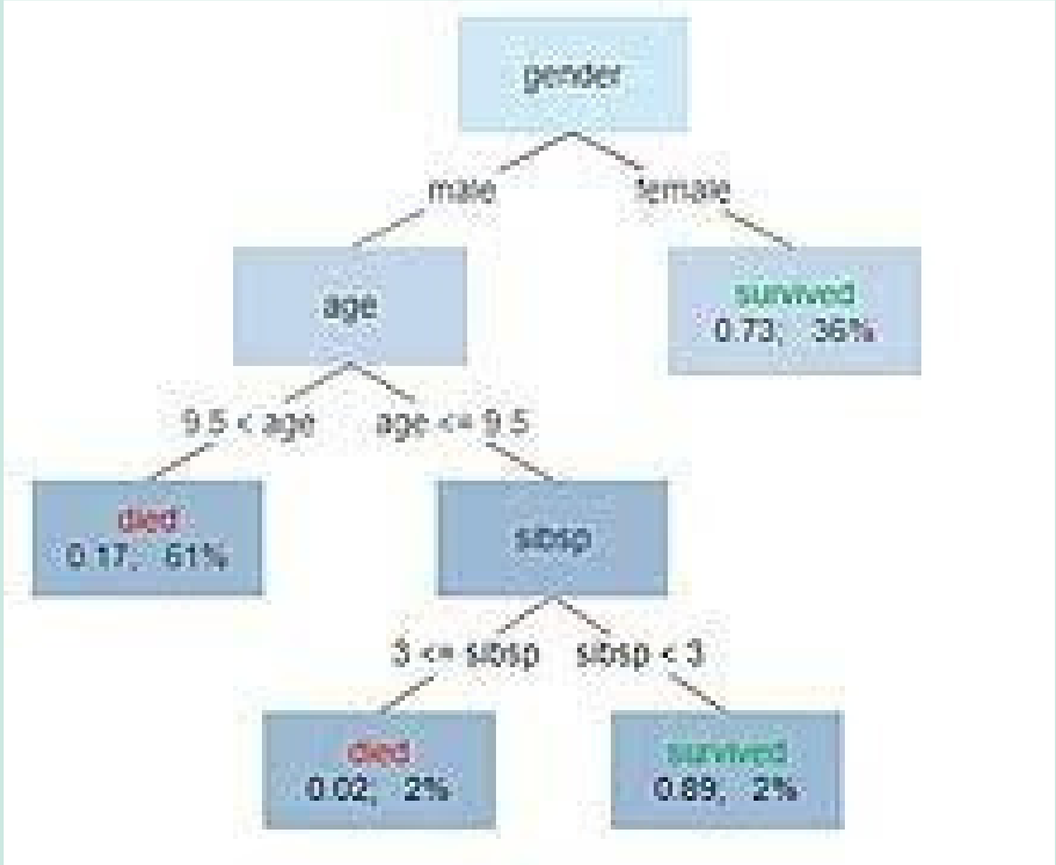
```
#Traning and testing score
print("Training score is",DT.score(X_train,y_train))
print("Testing score is",DT.score(X_test,y_test))
```

```
Training score is 1.0
Testing score is 0.9229677967937002
```

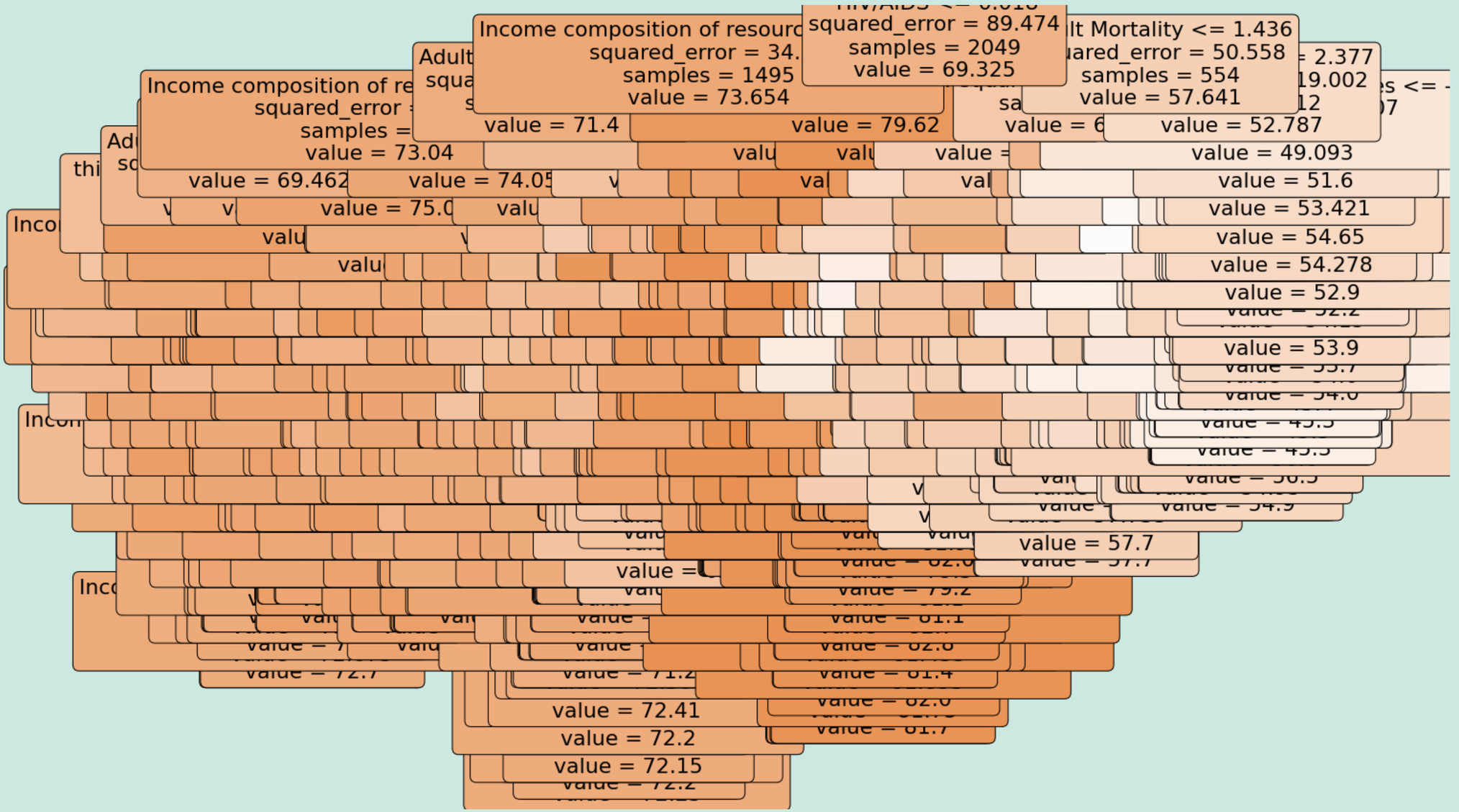
Concluion=The traning and testing score are very good.

```
#r2 score
predict_DT=DT.predict(X_test)
print("r2 score is",r2_score(y_test,predict_DT))
```

```
r2 score is 0.9229677967937002
```



Decision Tree



Decision Tree Hyper-parametric tuning

```
#DT after pruning
#Algoritm
DT_P=DecisionTreeRegressor(criterion='absolute_error', max_depth=10, min_samples_leaf=5, min_samples_
#fit
DT_P.fit(X_train,y_train)
```

```
DecisionTreeRegressor(criterion='absolute_error', max_depth=10,
                      min_samples_leaf=5, min_samples_split=10)
```

```
#Traning and testing score
print("Training score is",DT_P.score(X_train,y_train))
print("Testing score is",DT_P.score(X_test,y_test))
```

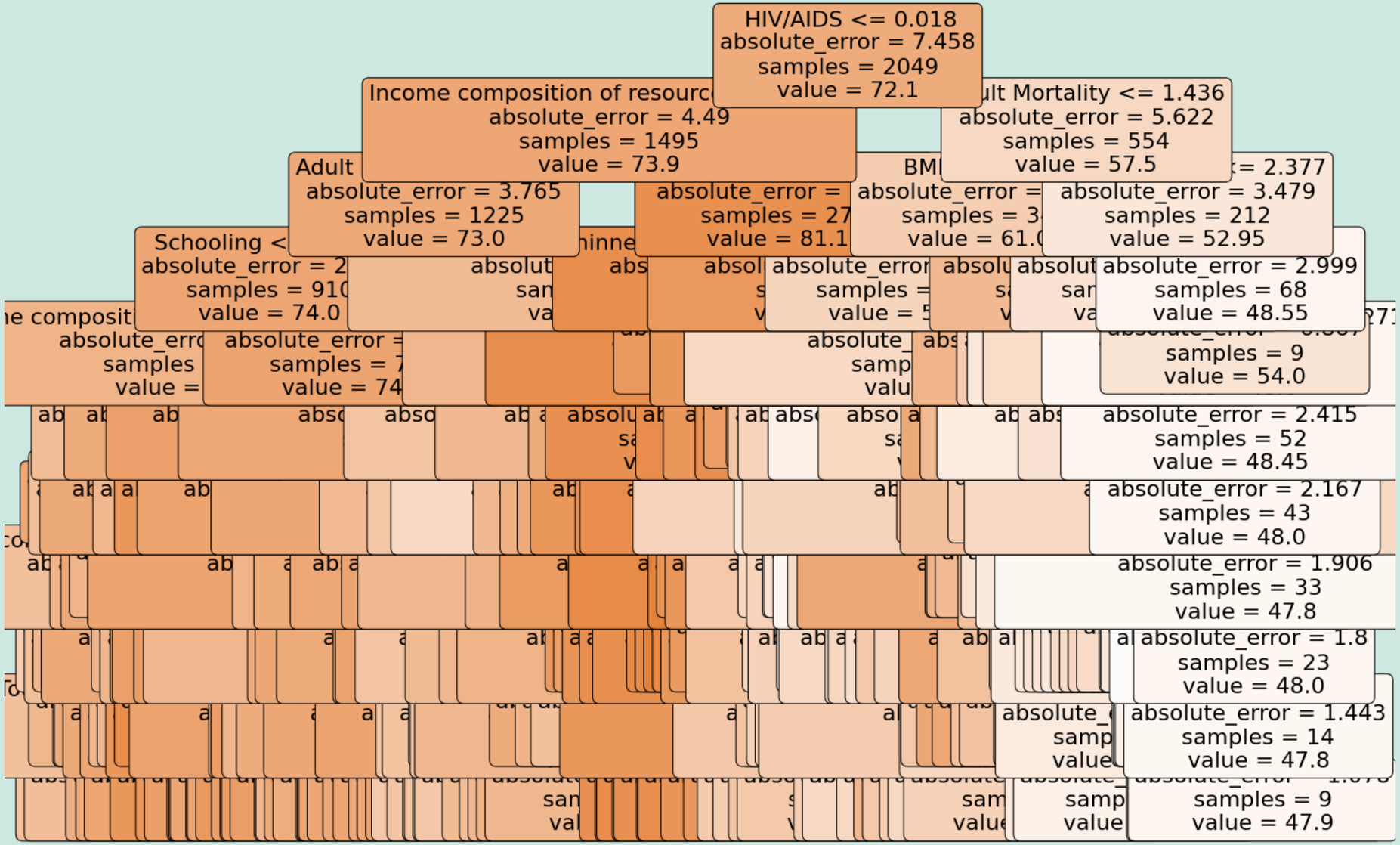
```
Training score is 0.9626384125926909
Testing score is 0.9165141346816498
```

Concluion=The traning has reduced but good enough.

```
#r2 score
predict_DT=DT_P.predict(X_test)
print("r2 score is",r2_score(y_test,predict_DT))
```

```
r2 score is 0.9165141346816498
```

Decision Tree After hyperparametric Tunning



Random Forest

```
#Algorithm
RF=RandomForestRegressor()
#Fit
RF.fit(X_train,y_train)
```

```
RandomForestRegressor()
```

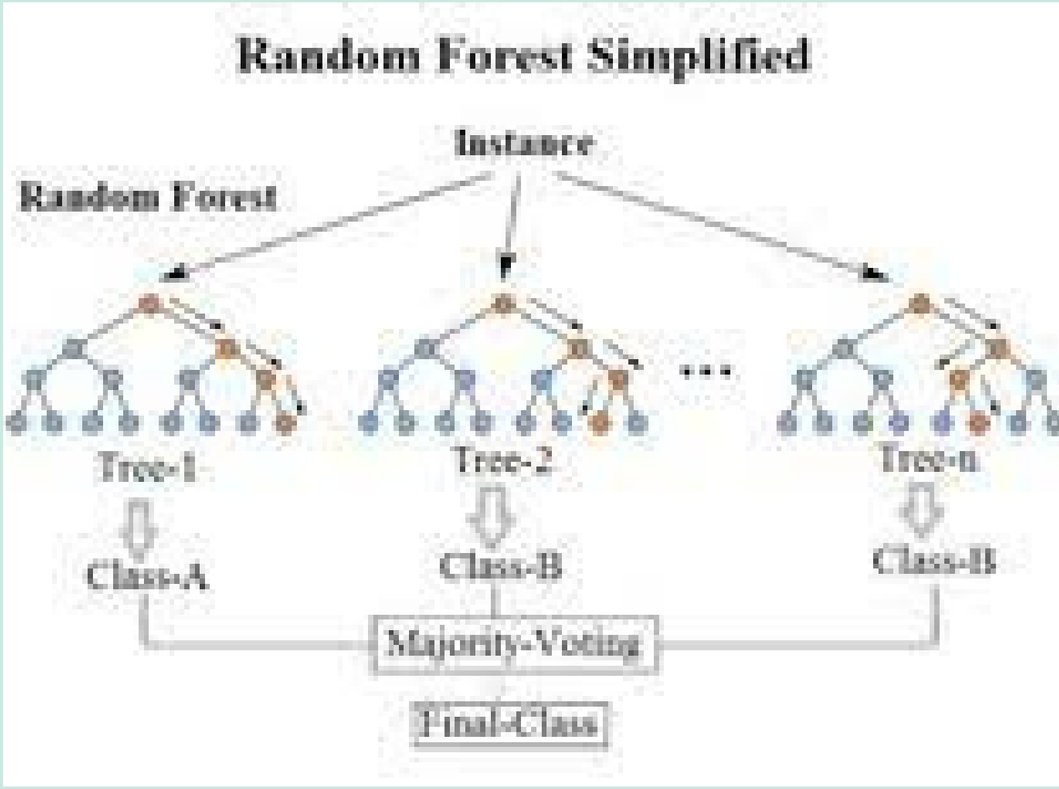
```
#Traning and testing score
print("Training score is",RF.score(X_train,y_train))
print("Testing score is",RF.score(X_test,y_test))
```

```
Training score is 0.9940526733592869
Testing score is 0.959739342505321
```

Conclusion=Good traning and testing.

```
#r2 score
predict_RF=RF.predict(X_test)
print("r2 score is",r2_score(y_test,predict_RF))
```

```
r2 score is 0.959739342505321
```



Extra Trees

```
#Algoritm
ET=ExtraTreesRegressor()
#Fit
ET.fit(X_train,y_train)
```

```
ExtraTreesRegressor()
```

```
#Traning and testing score
print("Training score is",ET.score(X_train,y_train))
print("Testing score is",ET.score(X_test,y_test))
```

```
Training score is 0.9999999995963625
Testing score is 0.966414542711537
```

Conclusion=Very good traning and testing.

```
#r2 score
predict_ET=ET.predict(X_test)
print("r2 score is",r2_score(y_test,predict_ET))
```

```
r2 score is 0.966414542711537
```

Bagging Regressor

```
#Algoritm
BR=BaggingRegressor()
#Fit
BR.fit(X_train,y_train)
```

```
BaggingRegressor()
```

```
#Traning and testing score
print("Training score is",BR.score(X_train,y_train))
print("Testing score is",BR.score(X_test,y_test))
```

```
Training score is 0.990952182247769 9
Testing score is 0.952158300761296 4
```

Conclusion=The Traning and testing both are very good.

```
#r2 score
predict_BR=BR.predict(X_test)
print("r2 score is",r2_score(y_test,predict_BR))
```

```
r2 score is 0.952158300761296 4
```

Adaboost

```
#Algorithm
AD=AdaBoostRegressor()
#Fit
AD.fit(X_train,y_train)
```

```
AdaBoostRegressor()
```

```
#Traning and testing score
print("Training score is",AD.score(X_train,y_train))
print("Testing score is",AD.score(X_test,y_test))
```

```
Training score is 0.9052760804926714
Testing score is 0.8968590577337301
```

```
#r2 score
predict_AD=AD.predict(X_test)
print("r2 score is",r2_score(y_test,predict_AD))
```

```
r2 score is 0.8968590577337301
```

Gradient Boost

```
#Algorithm
GB=GradientBoostingRegressor()
#Fit
GB.fit(X_train,y_train)
```

```
GradientBoostingRegressor()
```

```
#Traning and testing score
print("Training score is",GB.score(X_train,y_train))
print("Testing score is",GB.score(X_test,y_test))
```

```
Training score is 0.957530295102421
Testing score is 0.9405191514414325
```

Conclusion=Low bias and low varirance.

```
#r2 score
predict_GB =GB.predict(X_test)
print("r2 score is",r2_score(y_test,predict_GB))
```

```
r2 score is 0.9405191514414325
```

XgBoost

```
#Traning and testing score
print("Training score is",XB.score(X_train,y_train))
print("Testing score is",XB.score(X_test,y_test))
```

```
Training score is 0.9917518668376926
Testing score is 0.9575588665902822
```

Conclusion=Very good traning and very good testing.

```
#r2 score
predict_XB=XB.predict(X_test)
print("r2 score is",r2_score(y_test,predict_XB))
```

```
r2 score is 0.9575588665902822
```

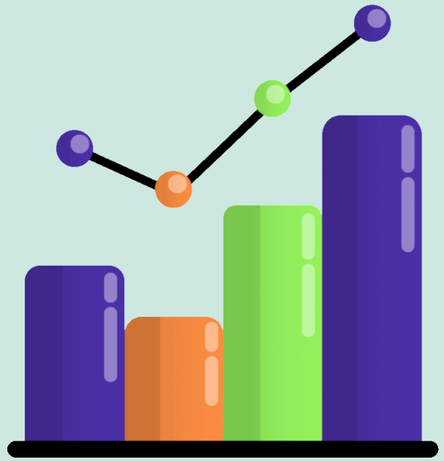
Result

	Algorithm	Traning Score	Testing Score	r2 value
1	Multiple Regression	0.85	0.850	0.850
2	KNN	0.93	0.910	0.913
3	SVM	0.88	0.880	0.880
4	DT	1.00	0.910	0.910
5	DT after hyperparametric Tunning	0.92	0.910	0.910
6	Random Forest	0.99	0.950	0.950
7	Bagging Regressor	0.98	0.955	0.955
8	Extra Trees Regressor	0.98	0.955	0.955
9	Adaboost Regressor	0.90	0.890	0.890
10	Gradient Boosting	0.95	0.940	0.940
11	XgBoost	0.99	0.950	0.957

Conclusion

XgBoost has the highest r2 score which is the best performance parameter in case of regression.

Hence XgBoost will be choosen for further regressions.



Prediction Comparison

	Actual value	Predicted value	Percentage error
2399	56.5	53.877747	0.048670
196	73.0	70.913506	0.029423
2316	82.5	81.858452	0.007837
1735	75.6	76.198807	-0.007858
1102	53.0	52.716797	0.005372
1194	65.2	65.668610	-0.007136
1507	72.4	73.308167	-0.012388
1161	74.1	73.859474	0.003257
322	77.0	75.604187	0.018462
1352	67.8	68.277275	-0.006990