World Happiness Report

Context:

The World Happiness Report is a landmark survey of the state of global happiness.
The reports review the state of happiness in the world today and show how the new science of happiness explains personal and national variations in happiness.

Project parts

Data Exploration & Visualisation

Models

Data source: https://www.kaggle.com

Problem Statements

Data Exploration & Visualisation

- Cleaning the data
- Merge two datasets
- How does each srtong happiness factor correlate with one another in 2021?
- What is mean of corruption in all countries in Regional indicator in 2021?
- What the happiest and saddiest countries (top 10 and bottom 10 in 2021)?
- How is the Healthy life expectancy factor in the happiest and saddiest counries in 2021?

We have two datasets

First data name: data2021

Second data name: data

```
:  #Import two Datasets
   data2021=pd.read_csv('world-happiness-report-2021.csv')
   data=pd.read_csv('world-happiness-report.csv')
```

Data Exploration Analysis:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 20 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Country name                              149 non-null    object
 1   Regional indicator                        149 non-null    object
 2   Ladder score                              149 non-null    float64
 3   Standard error of ladder score            149 non-null    float64
 4   upperwhisker                              149 non-null    float64
 5   lowerwhisker                              149 non-null    float64
 6   Logged GDP per capita                     149 non-null    float64
 7   Social support                            149 non-null    float64
 8   Healthy life expectancy                   149 non-null    float64
 9   Freedom to make life choices              149 non-null    float64
 10  Generosity                                149 non-null    float64
 11  Perceptions of corruption                 149 non-null    float64
 12  Ladder score in Dystopia                  149 non-null    float64
 13  Explained by: Log GDP per capita          149 non-null    float64
 14  Explained by: Social support              149 non-null    float64
 15  Explained by: Healthy life expectancy     149 non-null    float64
 16  Explained by: Freedom to make life choices 149 non-null   float64
 17  Explained by: Generosity                  149 non-null    float64
 18  Explained by: Perceptions of corruption   149 non-null    float64
 19  Dystopia + residual                       149 non-null    float64
dtypes: float64(18), object(2)
memory usage: 23.4+ KB
```

Info() of first data (data2021)

Info() of second data (data)

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1949 entries, 0 to 1948
Data columns (total 11 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country name                      1949 non-null   object
 1   year                              1949 non-null   int64
 2   Life Ladder                       1949 non-null   float64
 3   Log GDP per capita                1913 non-null   float64
 4   Social support                    1936 non-null   float64
 5   Healthy life expectancy at birth  1894 non-null   float64
 6   Freedom to make life choices      1917 non-null   float64
 7   Generosity                        1860 non-null   float64
 8   Perceptions of corruption         1839 non-null   float64
 9   Positive affect                   1927 non-null   float64
 10  Negative affect                   1933 non-null   float64
dtypes: float64(9), int64(1), object(1)
memory usage: 167.6+ KB
```

Data Cleaning:

```
In [6]: #missing data
        data2021.isnull().sum()

Out[6]: Country name                                0
        Regional indicator                          0
        Ladder score                                0
        Standard error of ladder score              0
        upperwhisker                                0
        lowerwhisker                                0
        Logged GDP per capita                       0
        Social support                              0
        Healthy life expectancy                     0
        Freedom to make life choices                0
        Generosity                                  0
        Perceptions of corruption                   0
        Ladder score in Dystopia                    0
        Explained by: Log GDP per capita            0
        Explained by: Social support                0
        Explained by: Healthy life expectancy       0
        Explained by: Freedom to make life choices  0
        Explained by: Generosity                    0
        Explained by: Perceptions of corruption     0
        Dystopia + residual                         0
        dtype: int64
```

No NaN value in First data(data2021)

Remove NaN value in second data

```
In [16]: #drop missing
         data=data.dropna()

In [17]: data.isnull().sum()

Out[17]: Country name                        0
         year                                0
         Life Ladder                         0
         Log GDP per capita                  0
         Social support                      0
         Healthy life expectancy at birth    0
         Freedom to make life choices        0
         Generosity                          0
         Perceptions of corruption           0
         Positive affect                     0
         Negative affect                     0
         dtype: int64
```

```
ı [14]: #missing data
        data.isnull().sum()

ıt[14]: Country name                        0
        year                                0
        Life Ladder                         0
        Log GDP per capita                 36
        Social support                     13
        Healthy life expectancy at birth   55
        Freedom to make life choices       32
        Generosity                         89
        Perceptions of corruption         110
        Positive affect                    22
        Negative affect                    16
        dtype: int64
```

Data Cleaning:

drop unnecessary columns & rename some columns data.

```
data2021 = data2021.drop(['Standard error of ladder score', 'upperwhisker', 'lowerwhisker',
       'Explained by: Log GDP per capita', 'Explained by: Social support',
       'Explained by: Healthy life expectancy',
       'Explained by: Freedom to make life choices',
       'Explained by: Generosity', 'Explained by: Perceptions of corruption', 'Ladder score in Dystopia',
                         'Dystopia + residual'], axis = 1)
```

```
[16]: data2021.head()
```

t[16]:

click to scroll output; double click to hide

	Country name	Regional indicator	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	Finland	Western Europe	7.842	10.775	0.954	72.0	0.949	-0.098	0.186
1	Denmark	Western Europe	7.620	10.933	0.954	72.7	0.946	0.030	0.179
2			7.571	11.117	0.942	74.4	0.919	0.025	0.292
3	Iceland	Western Europe	7.554	10.878	0.983	73.0	0.955	0.160	0.673
4	Netherlands	Western Europe	7.464	10.932	0.942	72.4	0.913	0.175	0.338

```
#drop unnecessary columns
data= data.drop(['Positive affect', 'Negative affect'], axis = 1)
```

```
#rename some columns
data.rename(columns = {'Life Ladder': 'Ladder score',
                       'Log GDP per capita':'Logged GDP per capita'
                       ,'Healthy life expectancy at birth':'Healthy life expectancy'},inplace='True')
data
```

```
4]: data.head()
```

4]:

	Country name	year	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	Afghanistan	2008	3.724	7.370	0.451	50.80	0.718	0.168	0.882
1	Afghanistan	2009	4.402	7.540	0.552	51.20	0.679	0.190	0.850
2	Afghanistan	2010	4.758	7.647	0.539	51.60	0.600	0.121	0.707
3	Afghanistan	2011	3.832	7.620	0.521	51.92	0.496	0.162	0.731
4	Afghanistan	2012	3.783	7.705	0.521	52.24	0.531	0.236	0.776

Merge two datasets:

```
]: #drop a column we don't need
data2021 = data2021.drop(['Regional indicator'],axis=1)
#Add year feature for data2021
data2021['year']=2021
new_data=pd.merge(data,data2021,how='outer')
```

```
: new_data
```

:

	Country name	year	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	Afghanistan	2008	3.724	7.370	0.451	50.800	0.718	0.168	0.882
1	Afghanistan	2009	4.402	7.540	0.552	51.200	0.679	0.190	0.850
2	Afghanistan	2010	4.758	7.647	0.539	51.600	0.600	0.121	0.707
3	Afghanistan	2011	3.832	7.620	0.521	51.920	0.496	0.162	0.731
4	Afghanistan	2012	3.783	7.705	0.521	52.240	0.531	0.236	0.776
...	...	...	...	...	...	...	...	...	...
1852	Lesotho	2021	3.512	7.926	0.787	48.700	0.715	-0.131	0.915
1853	Botswana	2021	3.467	9.782	0.784	59.269	0.824	-0.246	0.801
1854	Rwanda	2021	3.415	7.676	0.552	61.400	0.897	0.061	0.167
1855	Zimbabwe	2021	3.145	7.943	0.750	56.201	0.677	-0.047	0.821
1856	Afghanistan	2021	2.523	7.695	0.463	52.493	0.382	-0.102	0.924

1857 rows × 9 columns

The correlation of the new_data

```
#finding the correlation of the data
corr=new_data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	year	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
year	1.000000	0.064171	0.090051	0.013482	0.185613	0.264187	-0.041746	-0.100054
Ladder score	0.064171	1.000000	0.792625	0.716365	0.755120	0.528963	0.167975	-0.446527
Logged GDP per capita	0.090051	0.792625	1.000000	0.711879	0.859453	0.358373	-0.038199	-0.343938
Social support	0.013482	0.716365	0.711879	1.000000	0.623813	0.414582	0.043773	-0.225320
Healthy life expectancy	0.185613	0.755120	0.859453	0.623813	1.000000	0.392673	0.004725	-0.338215
Freedom to make life choices	0.264187	0.528963	0.358373	0.414582	0.392673	1.000000	0.312524	-0.482894
Generosity	-0.041746	0.167975	-0.038199	0.043773	0.004725	0.312524	1.000000	-0.278394
Perceptions of corruption	-0.100054	-0.446527	-0.343938	-0.225320	-0.338215	-0.482894	-0.278394	1.000000

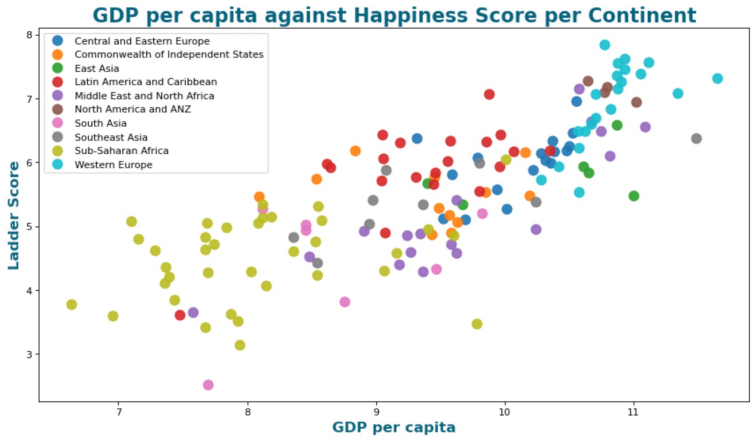
Description of columns in dataset

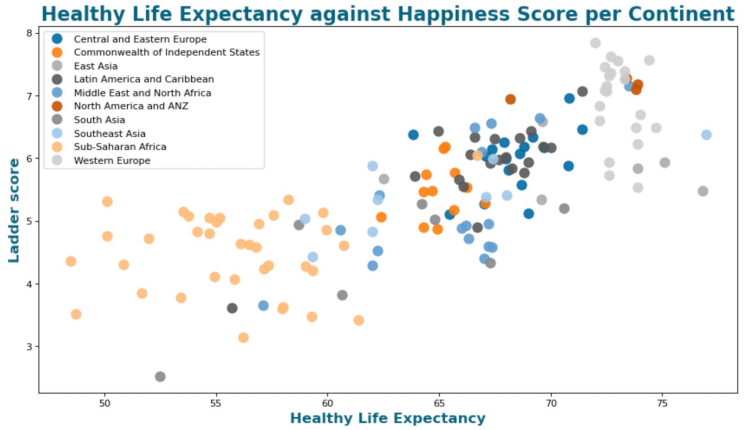
[Country name]:
Name of each Country
[Life Ladder] :
A metric measured by asking the sampled people the question: "How would you rate your happiness on a scale of 0 to 10 where 10 is the happiest."
[Logged GDP per capita]:
GDP per Capita of each Country in terms of Puchasing Power Parity (PPP) (in USD)
[Healthy life expectancy] :
Healthy Life Expectancy at birth are constructed based on data from the World Health Organization (WHO) and WDI.
[Social support]:
National average of the binary responses (either 0 or 1) to the question “If you were in trouble, do you have relatives or friends you can count on to help you whenever you need them, or not?”
[Freedom to make life choices] :
National average of binary responses to the question “Are you satisfied or dissatisfied with your freedom to choose what you do with your life?”
[Generosity]:
Generosity is the residual of regressing the national average of responses to the question “Have you donated money to a charity in the past month?” on GDP per capita.
[Perceptions of corruption]:
Perceptions of corruption are the average of binary answers to two GWP questions: “Is corruption widespread throughout the government or not?” and “Is corruption widespread within businesses or not?”

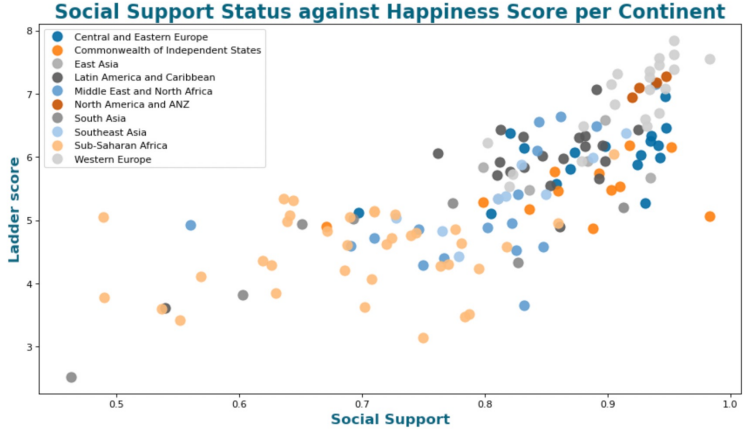
General Analysis :

how each factor affects overall happiness or one another

Logged GDP per capita, Social Support, and Healthy Life Expectancy have a strong degree of relationship with the Ladder Score. As each of these factors increases, the overall Ladder Score increases as well.







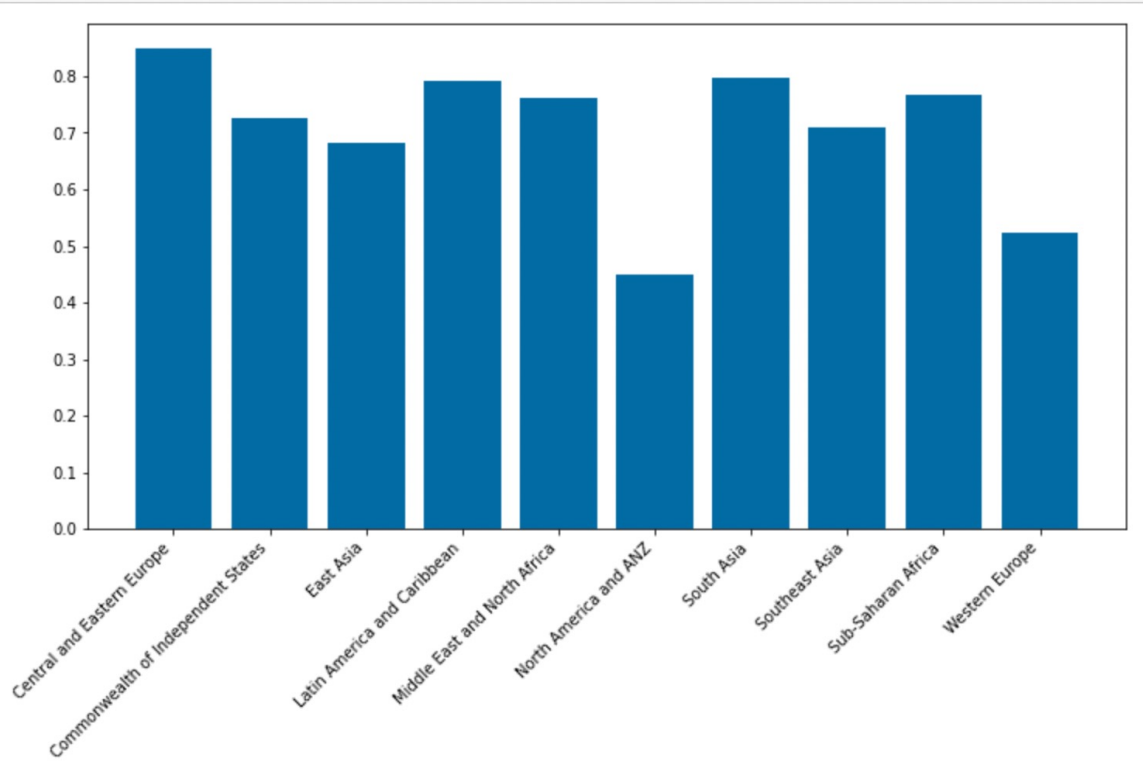
Factor to factor analysis shows strong degree of relationship

mean of corruption

```
corruption=data2021.groupby('Regional indicator')[['Perceptions of corruption']].mean()
```

```
corruption
```

Regional indicator	Perceptions of corruption
Central and Eastern Europe	0.850529
Commonwealth of Independent States	0.725083
East Asia	0.683333
Latin America and Caribbean	0.792600
Middle East and North Africa	0.762235
North America and ANZ	0.449250
South Asia	0.797429
Southeast Asia	0.709111
Sub-Saharan Africa	0.765944
Western Europe	0.523095



The mean of corruption in all countries in Regional indicator in 2021?

Top 10 and Bottom 10 countries:

```
# Sort the dataset in 2021
top_10 = data2021.sort_values(by='Ladder score', ascending=False).head(10)
bottom_10 = data2021.sort_values(by='Ladder score', ascending=False).tail(10)
```

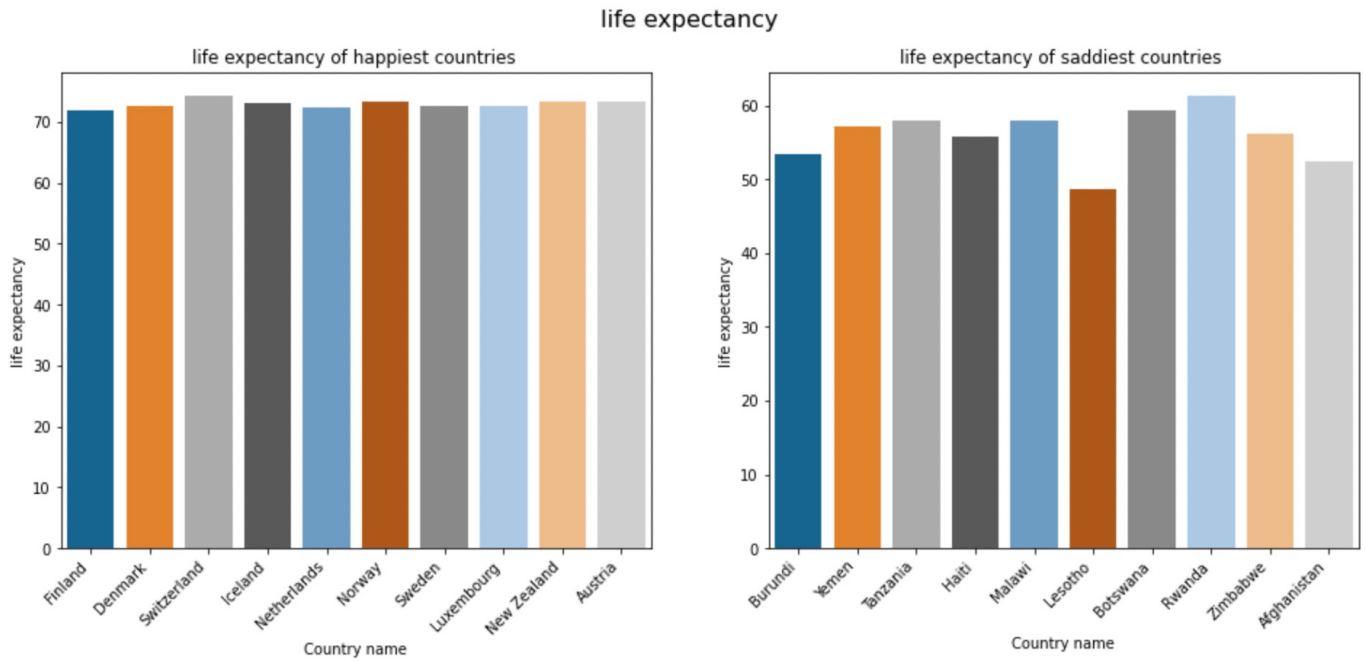
```
#Top 10 of happiest country
top_10
```

	Country name	Regional indicator	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	Finland	Western Europe	7.842	10.775	0.954	72.0	0.949	-0.098	0.186
1	Denmark	Western Europe	7.620	10.933	0.954	72.7	0.946	0.030	0.179
2	Switzerland	Western Europe	7.571	11.117	0.942	74.4	0.919	0.025	0.292
3	Iceland	Western Europe	7.554	10.878	0.983	73.0	0.955	0.160	0.673
4	Netherlands	Western Europe	7.464	10.932	0.942	72.4	0.913	0.175	0.338
5	Norway	Western Europe	7.392	11.053	0.954	73.3	0.960	0.093	0.270
6	Sweden	Western Europe	7.363	10.867	0.934	72.7	0.945	0.086	0.237
7	Luxembourg	Western Europe	7.324	11.647	0.908	72.6	0.907	-0.034	0.386
8	New Zealand	North America and ANZ	7.277	10.643	0.948	73.4	0.929	0.134	0.242
9	Austria	Western Europe	7.268	10.906	0.934	73.3	0.908	0.042	0.481

```
#Top 10 of saddest country
bottom_10
```

	Country name	Regional indicator	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
139	Burundi	Sub-Saharan Africa	3.775	6.635	0.490	53.400	0.626	-0.024	0.607
140	Yemen	Middle East and North Africa	3.658	7.578	0.832	57.122	0.602	-0.147	0.800
141	Tanzania	Sub-Saharan Africa	3.623	7.876	0.702	57.999	0.833	0.183	0.577
142	Haiti	Latin America and Caribbean	3.615	7.477	0.540	55.700	0.593	0.422	0.721
143	Malawi	Sub-Saharan Africa	3.600	6.958	0.537	57.948	0.780	0.038	0.729
144	Lesotho	Sub-Saharan Africa	3.512	7.926	0.787	48.700	0.715	-0.131	0.915
145	Botswana	Sub-Saharan Africa	3.467	9.782	0.784	59.269	0.824	-0.246	0.801
146	Rwanda	Sub-Saharan Africa	3.415	7.676	0.552	61.400	0.897	0.061	0.167
147	Zimbabwe	Sub-Saharan Africa	3.145	7.943	0.750	56.201	0.677	-0.047	0.821
148	Afghanistan	South Asia	2.523	7.695	0.463	52.493	0.382	-0.102	0.924

Healthy life expectancy in happiest countries and saddiest countries



models

- **KNN Regressor Algorithm Implementation**
- **Linear Regression**
- **Support Vector Regressor**
- **Decision Tree Regressor**
- **Random Forest Regression**
- **XG_BOOST_REGRESSOR**

Preprocessing of data

```
x=data2021.drop(["Ladder score"],axis=1)
y=data2021[["Ladder score"]]
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df1=le.fit_transform(x["Country name"])
x["Country name"]= df1
df1=le.fit_transform(x["Regional indicator"])
x["Regional indicator"]= df1
```

Preprocessing of data

```
from sklearn.preprocessing import StandardScaler
scaler1=StandardScaler()
X=pd.DataFrame(scaler1.fit_transform(x),columns=x.columns)
X.head()
```

	Country name	Regional indicator	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	-0.790484	1.254062	1.162885	1.216171	1.039750	1.393550	-0.551886	-3.031228
1	-0.953231	1.254062	1.299717	1.216171	1.143618	1.366990	0.300594	-3.070416
2	1.255474	1.254062	1.459064	1.111370	1.395869	1.127948	0.267294	-2.437802
3	-0.464991	1.254062	1.252086	1.469440	1.188133	1.446671	1.166393	-0.304829
4	0.511490	1.254062	1.298851	1.111370	1.099103	1.074828	1.266293	-2.180278

Train_Test_split

Train_Test_split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=20)
print(X_train.shape)
print(X_test.shape)
```

```
(119, 8)
(30, 8)
```

KNN Regressor Algorithm Implementation

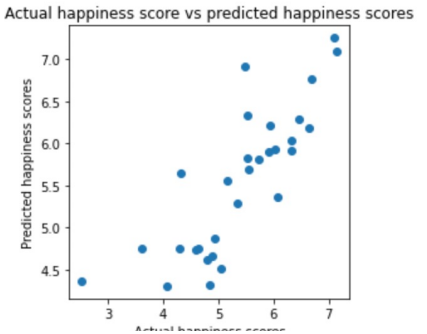
	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.424187	0.381555	0.639938

```
from sklearn.neighbors import KNeighborsRegressor
KNR = KNeighborsRegressor()
```

```
knnreg = KNR.fit(X_train, y_train)
```

```
predictions = knnreg.predict(X_test)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
plt.scatter(y_test,predictions)
plt.title("Actual happiness score vs predicted happiness scores")
plt.xlabel("Actual happiness scores")
plt.ylabel("Predicted happiness scores")
plt.show()
```




```
#checking the errors and r2 error
MAE_KNN= mean_absolute_error(y_test, predictions)
MSE_KNN= mean_squared_error(y_test, predictions)
R2_KNN = r2_score(y_test, predictions)

print("THe MAE of the KNN_regression is:", MAE_KNN)
print("THe MSE of the KNN_regression is:", MSE_KNN)
print("THe R^2 of the KNN_regression is:", R2_KNN)
```

```
THe MAE of the KNN_regression is: 0.4241866666666665
THe MSE of the KNN_regression is: 0.3815546079999999
THe R^2 of the KNN_regression is: 0.6399383594749849
```


```
#Linear Regressor
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_train,y_train)
#print(x1_train.shape, x1_test.shape, y1_train.shape, y1_test.shape)
```

```
LinearRegression()
```

```
predictions_1= lin_reg.predict(X_test)
#lin_reg.predict([[43,1.340,1.587,0.986,0.596,0.393]])
```

```
#finding the intercept
intercept = lin_reg.intercept_[0]
print("The intercept for the model is {}".format(intercept))
```

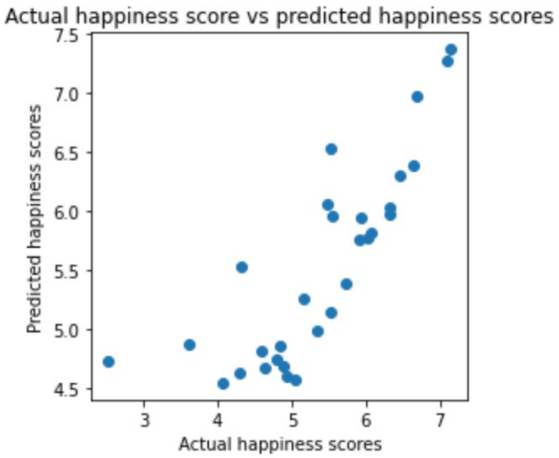
```
The intercept for the model is 5.550445853303329
```

```
#checking the errors and r2 error

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

MAE_LR= mean_absolute_error(y_test, predictions_1)
MSE_LR= mean_squared_error(y_test, predictions_1)
R2_LR = r2_score(y_test, predictions_1)

print("THe MAE of the Linear_regression is:", MAE_LR)
print("THe MSE of the Linear_regression is:", MSE_LR)
print("THe R^2 of the Linear_regression is:", R2_LR)
```

```
THe MAE of the Linear_regression is: 0.3842156833370932
THe MSE of the Linear_regression is: 0.26028264413195784
THe R^2 of the Linear_regression is: 0.7543790747605348
```

Linear Regression

	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.424187	0.381555	0.639938
1	LINEAR_REGRESSOR	0.384216	0.260283	0.754379




```
from sklearn.svm import SVR
regressor=SVR(kernel="rbf")
regressor.fit(X_train,np.array(y_train).ravel())
```

```
SVR()
```

```
predictions_2=regressor.predict(X_test)
```

```
MAE_SVR = mean_absolute_error(y_test, predictions_2)
MSE_SVR = mean_squared_error(y_test, predictions_2)
R2_SVR = r2_score(y_test, predictions_2)
print("THe MAE of the Support_Vector_Regressor is:", MAE_SVR)
print("THe MSE of the Support_Vector_Regressor is:", MSE_SVR)
print("THe R^2 of the Support_Vector_Regressor is:", R2_SVR)
```

```
THe MAE of the Support_Vector_Regressor is: 0.4134109889571923
THe MSE of the Support_Vector_Regressor is: 0.37223174235472484
THe R^2 of the Support_Vector_Regressor is: 0.6487360681862686
```

Support Vector Regressor

	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.424187	0.381555	0.639938
1	LINEAR_REGRESSOR	0.384216	0.260283	0.754379
2	SUPPORT_VECTOR_REGRESSOR	0.413411	0.372232	0.648736

Decision Tree Regressor

	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.398280	0.341891	0.677368
1	LINEAR_REGRESSOR	0.380402	0.240183	0.773347
2	SUPPORT_VECTOR_REGRESSOR	0.415027	0.381574	0.639920
3	DECISION_TREE_REGRESSOR	0.653567	0.617777	0.417022

```
#decision Tree Regressor

from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor()
regressor.fit(X_train,y_train)
```

```
DecisionTreeRegressor()
```

```
predictions_3=regressor.predict(X_test)
```

```
MAE_DTR = mean_absolute_error(y_test, predictions_3)
MSE_DTR = mean_squared_error(y_test, predictions_3)
R2_DTR = r2_score(y_test, predictions_3)
print("THe MAE of the Decision_Tree is:", MAE_DTR)
print("THe MSE of the Decision_Tree is:", MSE_DTR)
print("THe R^2 of the Decision_Tree is:", R2_DTR)
```

```
THe MAE of the Decision_Tree is: 0.6618333333333334
THe MSE of the Decision_Tree is: 0.6550536333333334
THe R^2 of the Decision_Tree is: 0.3818455316627396
```

Random Forest Regression

	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.398280	0.341891	0.677368
1	LINEAR_REGRESSOR	0.380402	0.240183	0.773347
2	SUPPORT_VECTOR_REGRESSOR	0.415027	0.381574	0.639920
3	DECISION_TREE_REGRESSOR	0.653567	0.617777	0.417022
4	RANDOM_FOREST_REGRESSOR	0.461017	0.337967	0.681071

```
]: from sklearn.ensemble import RandomForestRegressor
   from sklearn.model_selection import GridSearchCV
   params_to_test = {
       'n_estimators':range(1,11)
   }

   rf_model=RandomForestRegressor()
```

```
]: grid_search=GridSearchCV(rf_model, param_grid=params_to_test)

   grid_search.fit(X_train,np.array(y_train).ravel())
```

```
]: GridSearchCV(estimator=RandomForestRegressor(),
                  param_grid={'n_estimators': range(1, 11)})
```

```
]: predictions_4=grid_search.predict(X_test)
```

```
]: MAE_RF = mean_absolute_error(y_test, predictions_4)
   MSE_RF = mean_squared_error(y_test, predictions_4)
   R2_RF = r2_score(y_test, predictions_4)
   print("THe MAE of the Random_forest is:",MAE_RF)
   print("THe MSE of the Random_forest is:", MSE_RF)
   print("THe R^2 of the Random_forest is:", R2_RF)
```

```
THe MAE of the Random_forest is: 0.3779199999999998
THe MSE of the Random_forest is: 0.25609053599999976
THe R^2 of the Random_forest is: 0.7583350414809028
```


```
from xgboost import XGBRegressor
xgreg = XGBRegressor()
xgreg.fit(X_train, y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
predictions_5 = xgreg.predict(X_test)
```

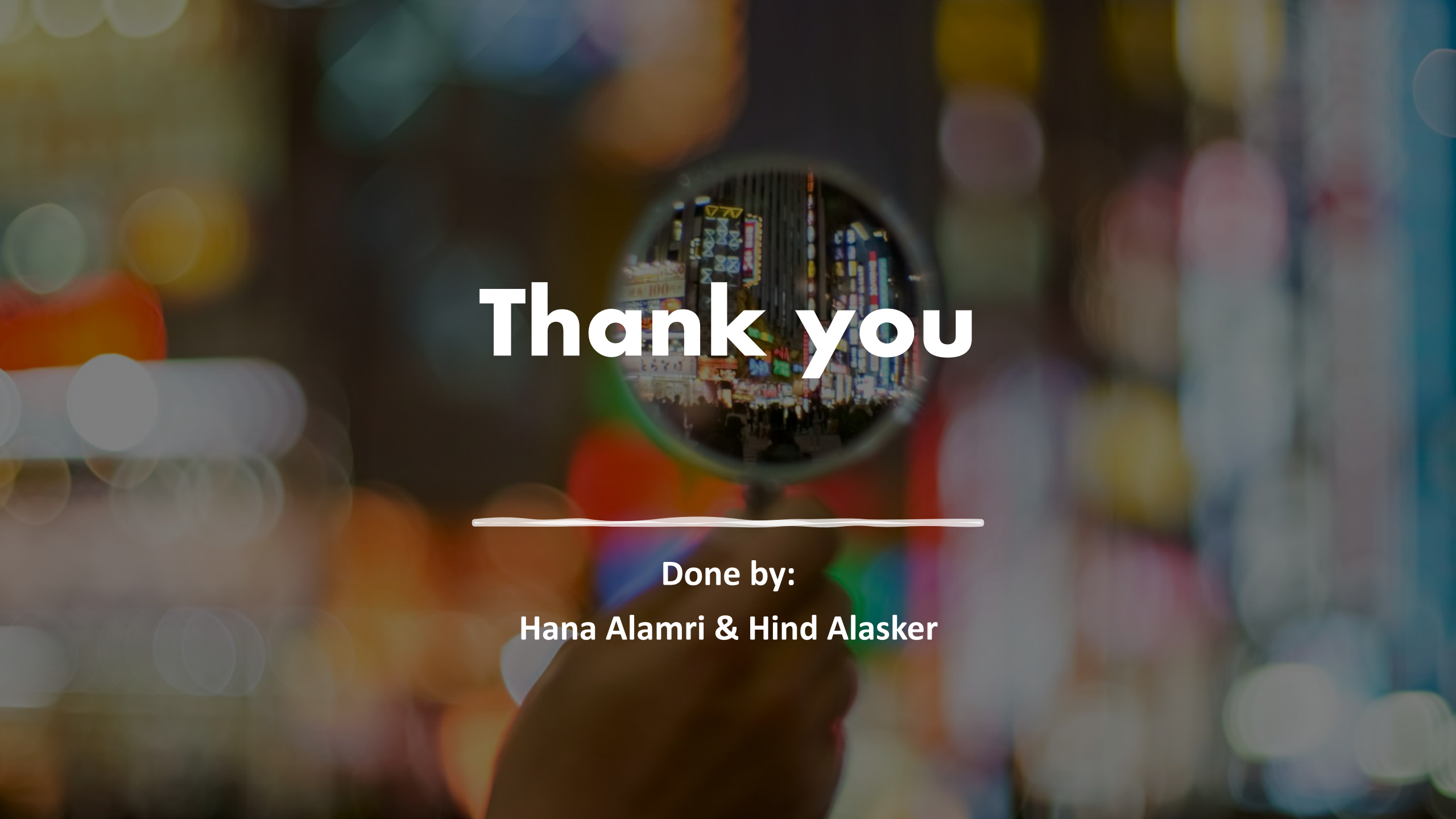
```
MAE_RF = mean_absolute_error(y_test, predictions_5)
MSE_RF = mean_squared_error(y_test, predictions_5)
R2_RF = r2_score(y_test, predictions_5)
print("THe MAE of the XG Boost Regressor is:",MAE_RF)
print("THe MSE of the XG Boost Regressor  is:", MSE_RF)
print("THe R^2 of the XG Boost Regressor  is:", R2_RF)
```

```
THe MAE of the XG Boost Regressor is: 0.33036000518798825
THe MSE of the XG Boost Regressor  is: 0.20255680351365604
THe R^2 of the XG Boost Regressor  is: 0.8088532193204959
```

XG_BOOST_REGRESSOR

	Model	Mean Absolute Error	Mean Squared Error	R Squared Error
0	KNN_REGRESSOR	0.424187	0.381555	0.639938
1	LINEAR_REGRESSOR	0.384216	0.260283	0.754379
2	SUPPORT_VECTOR_REGRESSOR	0.413411	0.372232	0.648736
3	DECISION_TREE_REGRESSOR	0.695900	0.697670	0.341629
4	RANDOM_FOREST_REGRESSOR	0.377920	0.256091	0.758335
5	XG_BOOST_REGRESSOR	0.333956	0.198637	0.812553

XG_BOOST_REGRESSOR
is giving the best results



Thank you

Done by:

Hana Alamri & Hind Alasker