PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

APRIL 2022: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE20MA251- LINEAR ALGEBRA

Project / Seminar

Session: Jan-May 2022

Branch : Computer Science and Engineering

Semester & Section : Semester IV Section K

Sl No.	Name of the Student	SRN	Marks Allotted (Out of 10)
1.	BHARGAV M V	PES1UG20CS660	
2.	MAHESH SHRIPAD BHAT	PES1UG20CS661	
3.	P CHAITANYA P S	PES1UG20CS669	
4.	MANOJ KUMAR DARSHANKAR	PES1UG20CS662	

Name of the Course Instructor :**Prof. NAGEGOWDA K S**

Signature of the Course Instructor

(with Date) : ______________________________

PROBLEM STATEMENT:

1. Clean your dataset, perform exploratory data analysis and come up with meaningful visualisations & statistical analysis to derive your initial hypothesis and note down your findings from the dataset.
2. Using PCA to be able to determine features that significantly determine the happiness index for that nation and provide reasonable explanation to some of the visible inferences that have come about due to these operations.
3. Build a multiple linear regression model to predict the happiness score given parameters for any country, for both the actual dataset and the dimension reduced dataset using PCA, and determine which is better.

Data parameter explanation (taken from the kaggle website, from the Details column at the beginning, and google)

- Country or region : Country name
- Score : The country's score in the hapiness index
- Overall rank : Rank of the country
- GDP per capita : GDP per capita is a measure of a country's economic output and how it affects the HI
- Social support : Social support means having friends and other people, including family, to turn to in times of need or crisis to support you, and how it affects the HI
- Healthy life expectancy :Healthy Life Expectancy is the average number of years that a newborn can expect to live, and how it affects the HI
- Freedom to make life choices : the freedom to follow their own paths and make their own decisions, and how it affects the HI
- Generosity:being kind and generous,and how it affects the HI
- Perception of corruption : Trust in government,how it affects the HI

DATASET:

Kaggle dataset of Happiness index of different countries in 2019.

MAJOR LIBRARIES USED:

- Scikit-learn
- Pandas

- NumPy
- PyPlot
- SciPy
- Seaborn

Model used:Multiple Linear regression model.

WHAT IS EDA?

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

1. maximize insight into a data set;
2. uncover underlying structure;
3. extract important variables;
4. detect outliers and anomalies;
5. test underlying assumptions;
6. develop parsimonious models; and
7. determine optimal factor settings.

What is Dimensionality Reduction?

The number of input features, variables, or columns present in a given dataset is known
as dimensionality, and the process to reduce these features is called dimensionality
reduction.
A dataset contains a huge number of input features in various cases, which makes the
predictive modelling task more complicated. Because it is very difficult to visualize or
make predictions for the training dataset with a high number of features, for such
cases, dimensionality reduction techniques are required to use.
Dimensionality reduction technique can be defined as, "It is a way of converting the
higher dimensions dataset into lesser dimensions dataset ensuring that it
provides similar information." These techniques are widely used in machine
learning for obtaining a better fit predictive model while solving the classification and

regression problems.
It is commonly used in the fields that deal with high-dimensional data, such as speech
recognition, signal processing, bioinformatics, etc. It can also be used for data
visualization, noise reduction, cluster analysis, etc

The Curse of Dimensionality

Handling the high-dimensional data is very difficult in practice, commonly known as
the curse of dimensionality. If the dimensionality of the input dataset increases, any
machine learning algorithm and model becomes more complex. As the number of
features increases, the number of samples also gets increased proportionally, and the
chance of overfitting also increases. If the machine learning model is trained on high-
dimensional data, it becomes overfitted and results in poor performance.
Hence, it is often required to reduce the number of features, which can be done with
dimensionality reduction.
Benefits of applying Dimensionality Reduction
Some benefits of applying dimensionality reduction technique to the given dataset are
given below:

- By reducing the dimensions of the features, the space required to store the
dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data
quickly.
- It removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of dimensionality Reduction
There are also some disadvantages of applying the

dimensionality reduction, which
are given below:

- Some data may be lost due to dimensionality reduction.
- In the PCA dimensionality reduction technique, sometimes the principal
components required to consider are unknown

Feature Selection:

Feature selection is the process of selecting the subset of the relevant features and
leaving out the irrelevant features present in a dataset to build a model of high
accuracy. In other words, it is a way of selecting the optimal features from the input
dataset.

Feature Extraction:

Feature extraction is the process of transforming the space containing many
dimensions into space with fewer dimensions. This approach is useful when we want
to keep the whole information but use fewer resources while processing the
information.
Some common feature extraction techniques are:
a) Principal Component Analysis
b) Linear Discriminant Analysis

SHORT NOTE ON PRINCIPAL COMPONENT ANALYSIS:

The principal components of a collection of points in a real coordinate space are a sequence of p unit vectors, where the i-th vector is the direction of a line that best fits the data while being orthogonal to the first i – 1 vectors. Here, a best-fitting line is defined as one that minimizes the average squared distance from the points to the line. These directions constitute an orthonormal basis in which different individual dimensions of the data are linearly uncorrelated. Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data. The i-th principal component can be taken as a direction orthogonal to the first i − 1 principal components that maximizes the variance of the projected data.

.

SHORT NOTE ON LINEAR REGRESSION:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for linear Regression:

y=mx+c

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best c and m values.

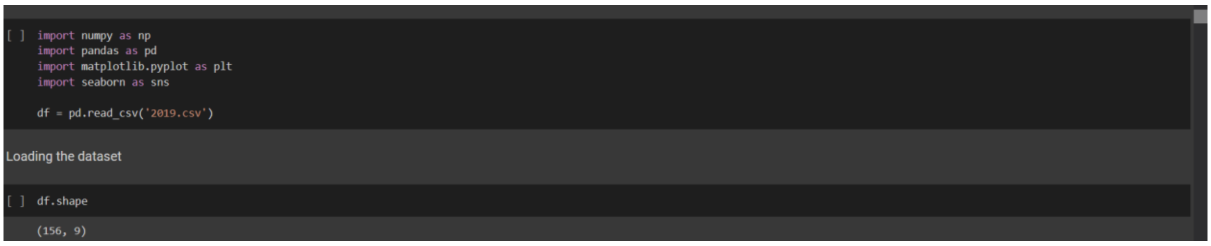
c: intercept
m: coefficient of x

Once we find the best c and m values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

LINK TO OUR NOTEBOOK OF PROJECT:

https://colab.research.google.com/drive/1TX6hWhBtkOoAdXs1iZZkla0LV6OreF3p#scrollTo=ijNf6WPjmSui

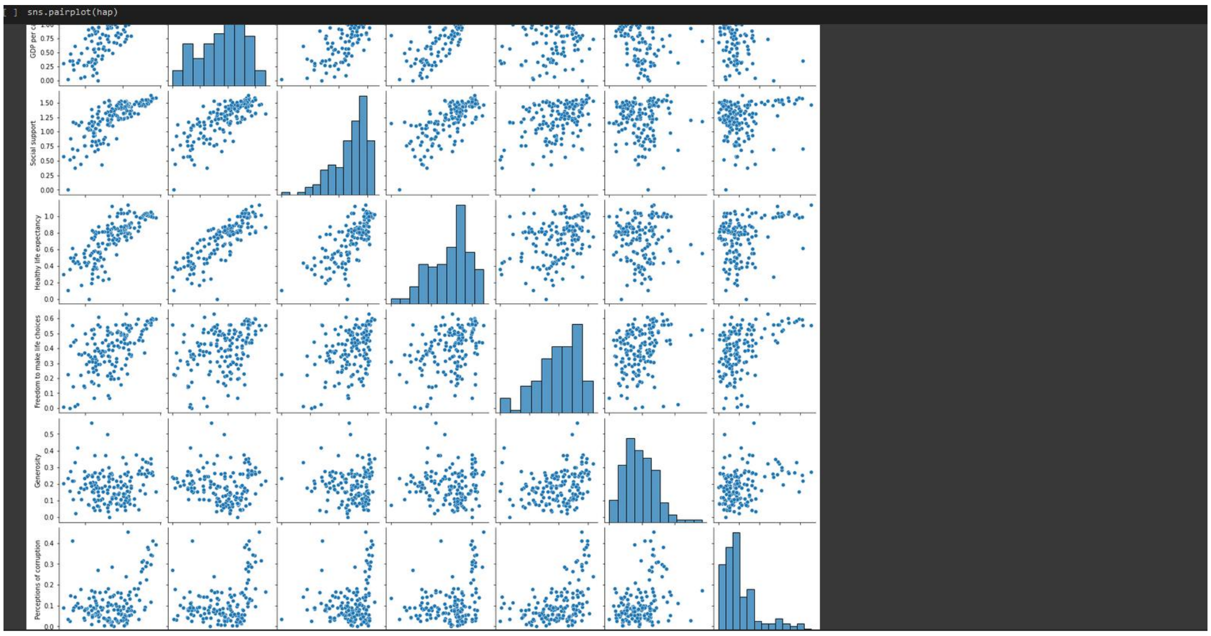
OUTPUT SCREENSHOT:



```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     df = pd.read_csv('2019.csv')
```

Loading the dataset

```
[ ]  df.shape

     (156, 9)
```



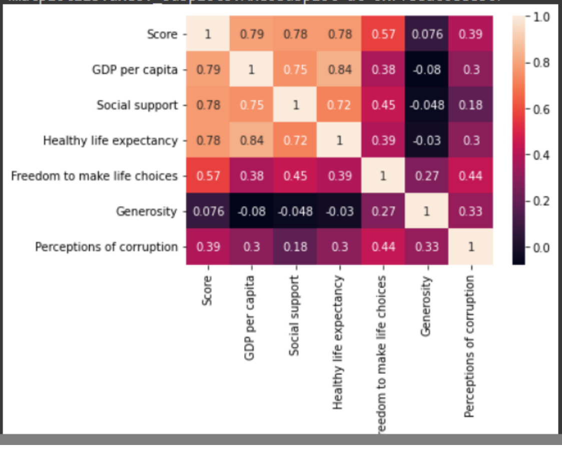
```
[ ]  sns.pairplot(hap)
[ ]  hap=df.drop(['Country or region', 'Overall rank'], axis=1)
```

Columns like Country or Overall Rank doesnot affect the people's happiness.

```
[ ]  corelation=hap.corr()
```

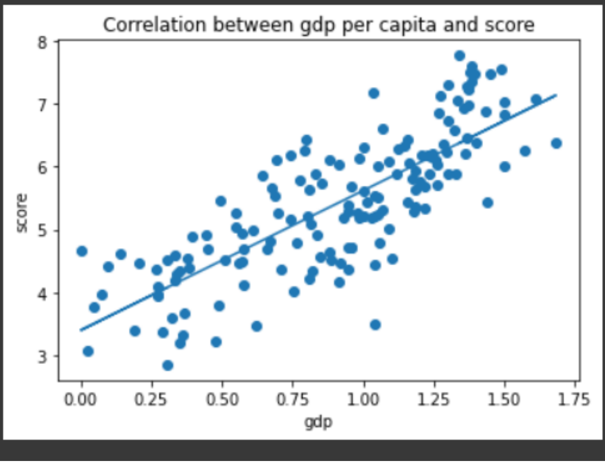
```
sns.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f88ac088850>
```



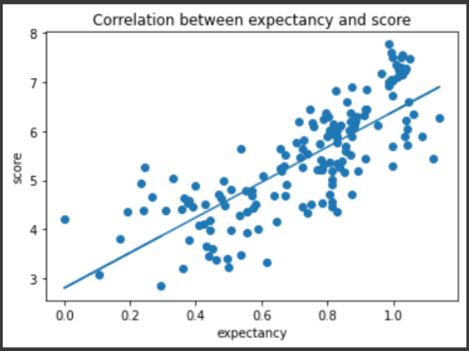
```
]  fig, ax = plt.subplots()
   x=df['GDP per capita']
   y=df['Score']
   ax.scatter(x,y)
   title = f'Correlation between gdp per capita and score'
   ax.set(xlabel='gdp',ylabel='score',title=title)
   m, b = np.polyfit(x, y, 1)
   coeff = np.corrcoef(x,y)[0][1]
   print("Correlation = {0:.2f}".format(coeff))
   print()
   plt.plot(x, m*x + b)
   plt.show()
```

```
Correlation = 0.79
```



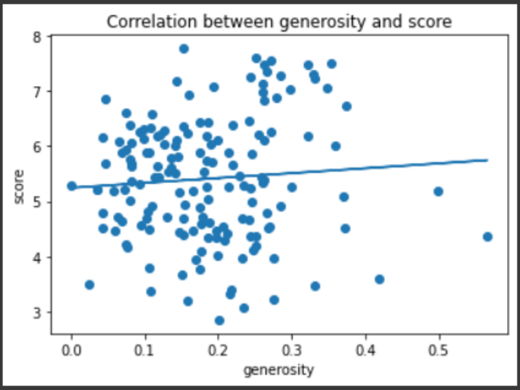
```
fig, ax = plt.subplots()
x=df['Healthy life expectancy']
y=df['Score']
ax.scatter(x,y)
title = f'Correlation between expectancy and score'
ax.set(xlabel='expectancy',ylabel='score',title=title)
m, b = np.polyfit(x, y, 1)
coeff = np.corrcoef(x,y)[0][1]
print("Correlation = {0:.2f}".format(coeff))
print()
plt.plot(x, m*x + b)
plt.show()
```

```
Correlation = 0.78
```



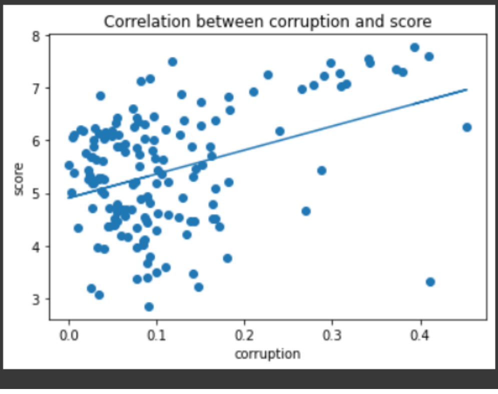
```
[ ] fig, ax = plt.subplots()
x=df['Generosity']
y=df['Score']
ax.scatter(x,y)
title = f'Correlation between generosity and score'
ax.set(xlabel='generosity',ylabel='score',title=title)
m, b = np.polyfit(x, y, 1)
coeff = np.corrcoef(x,y)[0][1]
print("Correlation = {0:.2f}".format(coeff))
print()
plt.plot(x, m*x + b)
plt.show()
```

```
Correlation = 0.08
```



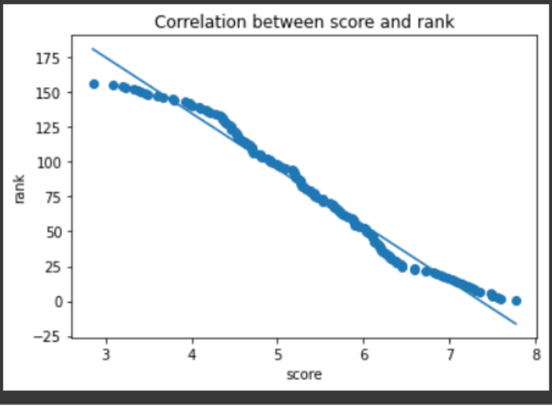
```
fig, ax = plt.subplots()
x=df['Perceptions of corruption']
y=df['Score']
ax.scatter(x,y)
title = f'Correlation between corruption and score'
ax.set(xlabel='corruption',ylabel='score',title=title)
m, b = np.polyfit(x, y, 1)
coeff = np.corrcoef(x,y)[0][1]
print("Correlation = {0:.2f}".format(coeff))
print()
plt.plot(x, m*x + b)
plt.show()
```

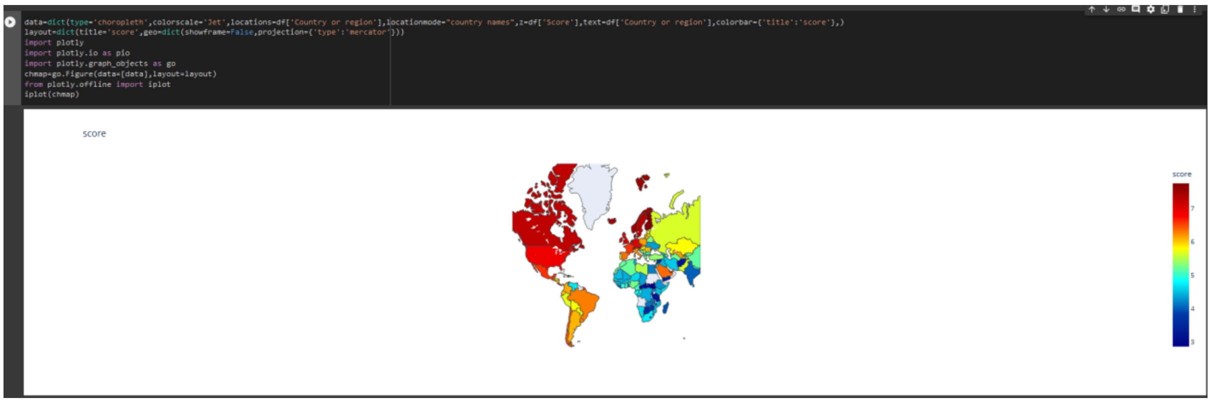
```
Correlation = 0.39
```



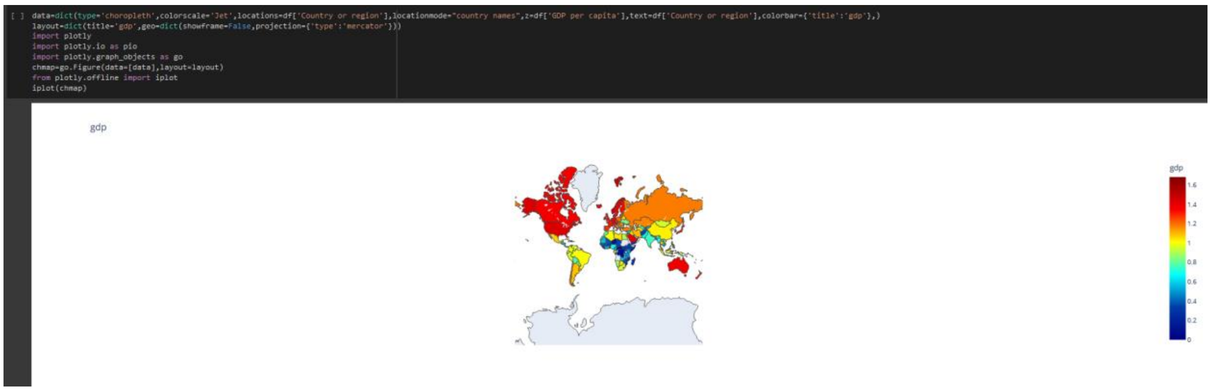
```
fig, ax = plt.subplots()
x=df['Score']
y=df['Overall rank']
ax.scatter(x,y)
title = f'Correlation between score and rank'
ax.set(xlabel='score',ylabel='rank',title=title)
m, b = np.polyfit(x, y, 1)
coeff = np.corrcoef(x,y)[0][1]
print("Correlation = {0:.2f}".format(coeff))
print()
plt.plot(x, m*x + b)
plt.show()
```

```
Correlation = -0.99
```

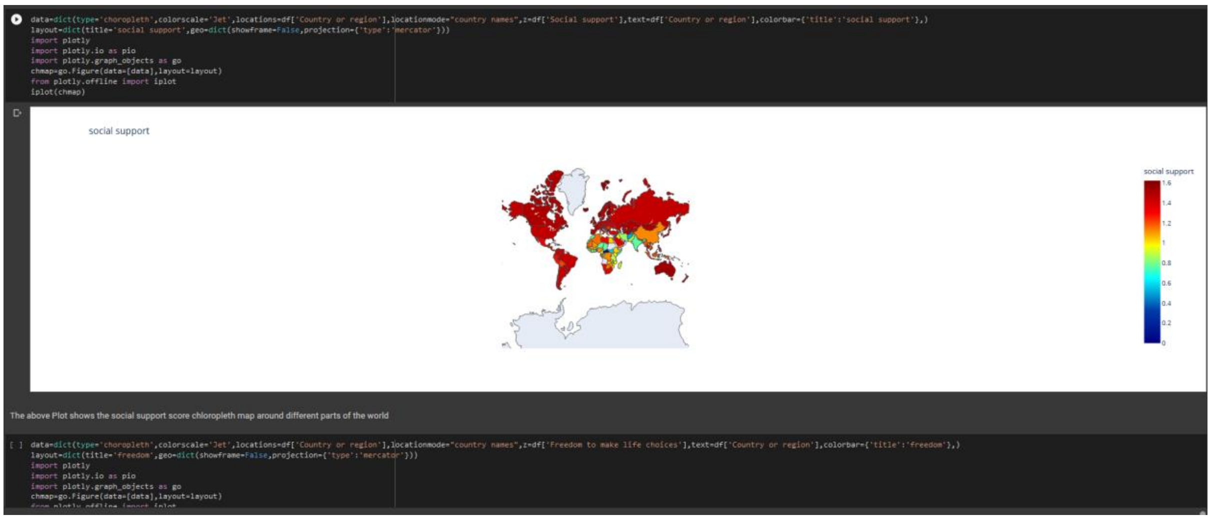




```
data=dict(type='choropleth',colorscale='Jet',locations=df['Country or region'],locationmode="country names",z=df['Score'],text=df['Country or region'],colorbar={'title':'score'},)
layout=dict(title='score',geo=dict(showframe=False,projection={'type':'mercator'}))
import plotly
import plotly.io as pio
import plotly.graph_objects as go
chmap=go.Figure(data=[data],layout=layout)
from plotly.offline import iplot
iplot(chmap)
```



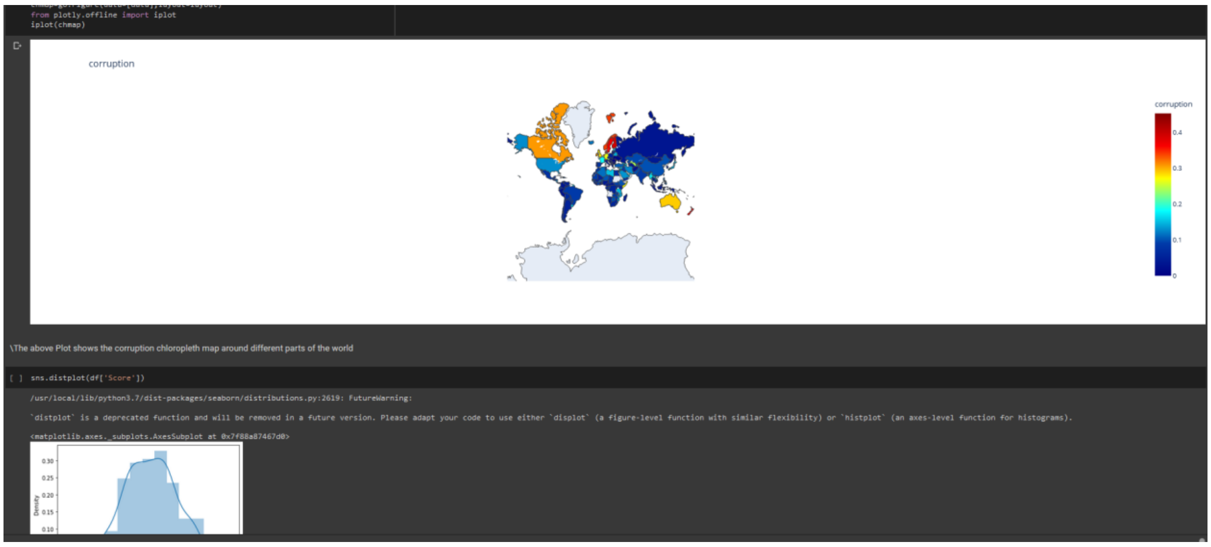
```
[ ] data=dict(type='choropleth',colorscale='Jet',locations=df['Country or region'],locationmode="country names",z=df['GDP per capita'],text=df['Country or region'],colorbar={'title':'gdp'},)
    layout=dict(title='gdp',geo=dict(showframe=False,projection={'type':'mercator'}))
    import plotly
    import plotly.io as pio
    import plotly.graph_objects as go
    chmap=go.Figure(data=[data],layout=layout)
    from plotly.offline import iplot
    iplot(chmap)
```



```
data=dict(type='choropleth',colorscale='Jet',locations=df['Country or region'],locationmode="country names",z=df['Social support'],text=df['Country or region'],colorbar={'title':'social support'},)
layout=dict(title='social support',geo=dict(showframe=False,projection={'type':'mercator'}))
import plotly
import plotly.io as pio
import plotly.graph_objects as go
chmap=go.Figure(data=[data],layout=layout)
from plotly.offline import iplot
iplot(chmap)
```

The above Plot shows the social support score chloropleth map around different parts of the world

```
[ ] data=dict(type='choropleth',colorscale='Jet',locations=df['Country or region'],locationmode="country names",z=df['Freedom to make life choices'],text=df['Country or region'],colorbar={'title':'freedom'},)
    layout=dict(title='freedom',geo=dict(showframe=False,projection={'type':'mercator'}))
    import plotly
    import plotly.io as pio
    import plotly.graph_objects as go
    chmap=go.Figure(data=[data],layout=layout)
```



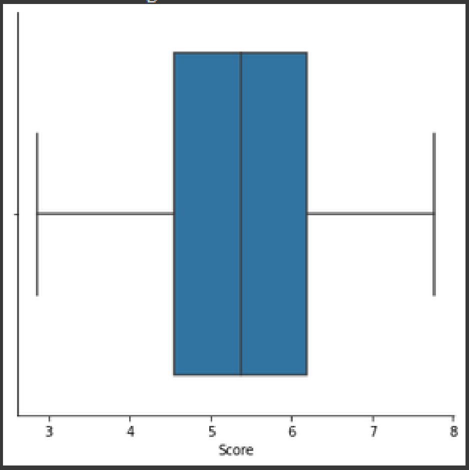
```
    from plotly.offline import iplot
    iplot(chmap)
```

\The above Plot shows the corruption chloropleth map around different parts of the world

```
[ ] sns.distplot(df['Score'])

    /usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:

    `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

    <matplotlib.axes._subplots.AxesSubplot at 0x7f88a87467d0>
```

```
[ ] sns.catplot(x='Score',kind='box', data=hap)
```

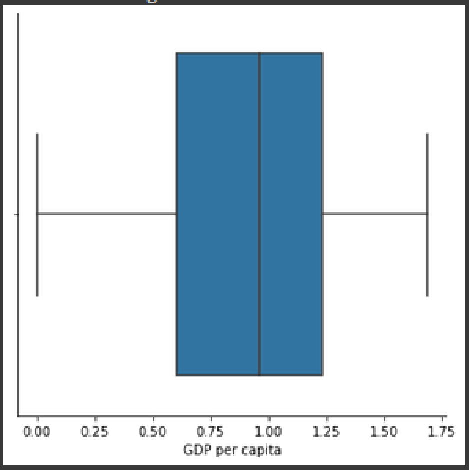
```
<seaborn.axisgrid.FacetGrid at 0x7f88a8513d50>
```



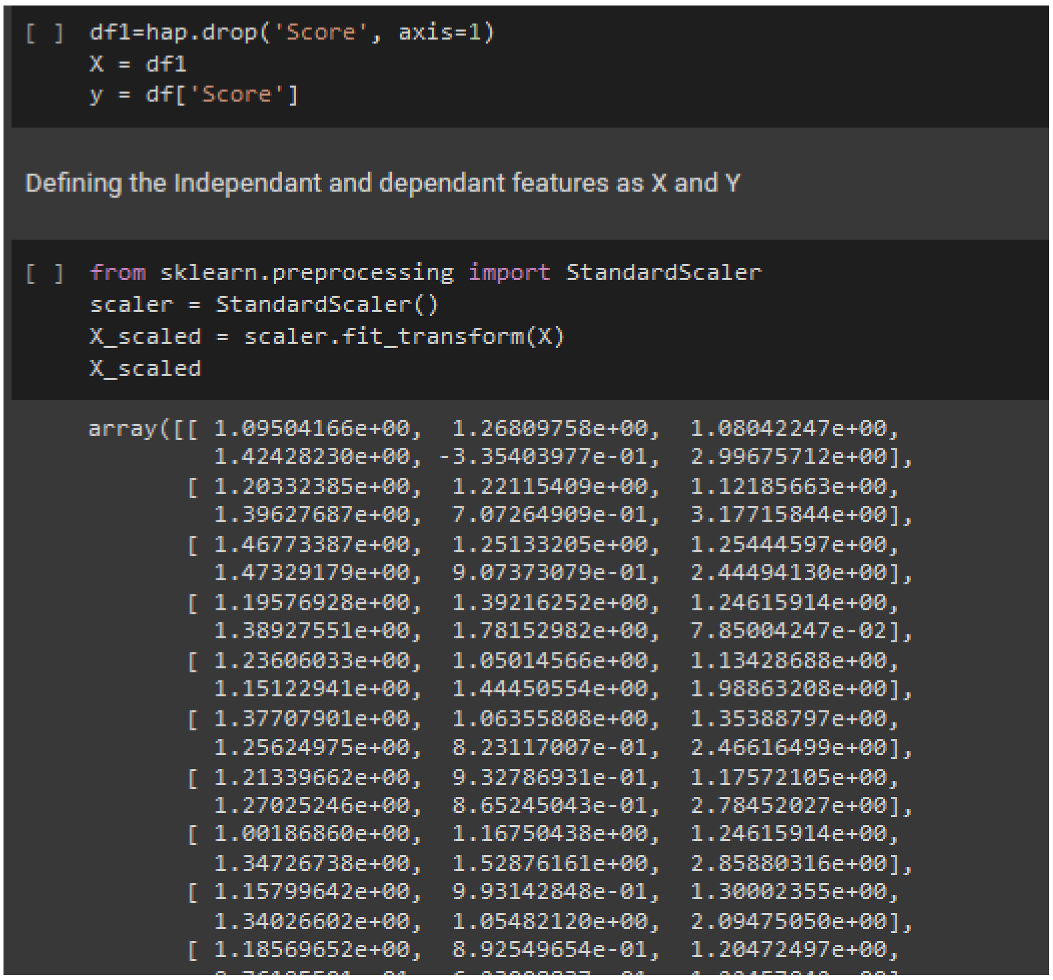
Most values of Score fall between 4.5 and 6.3, without any outliers

```
[ ] sns.catplot(x='GDP per capita',kind='box', data=hap)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f88a8480490>
```



Most of the values of GDP fall between 0.5 and 1.25

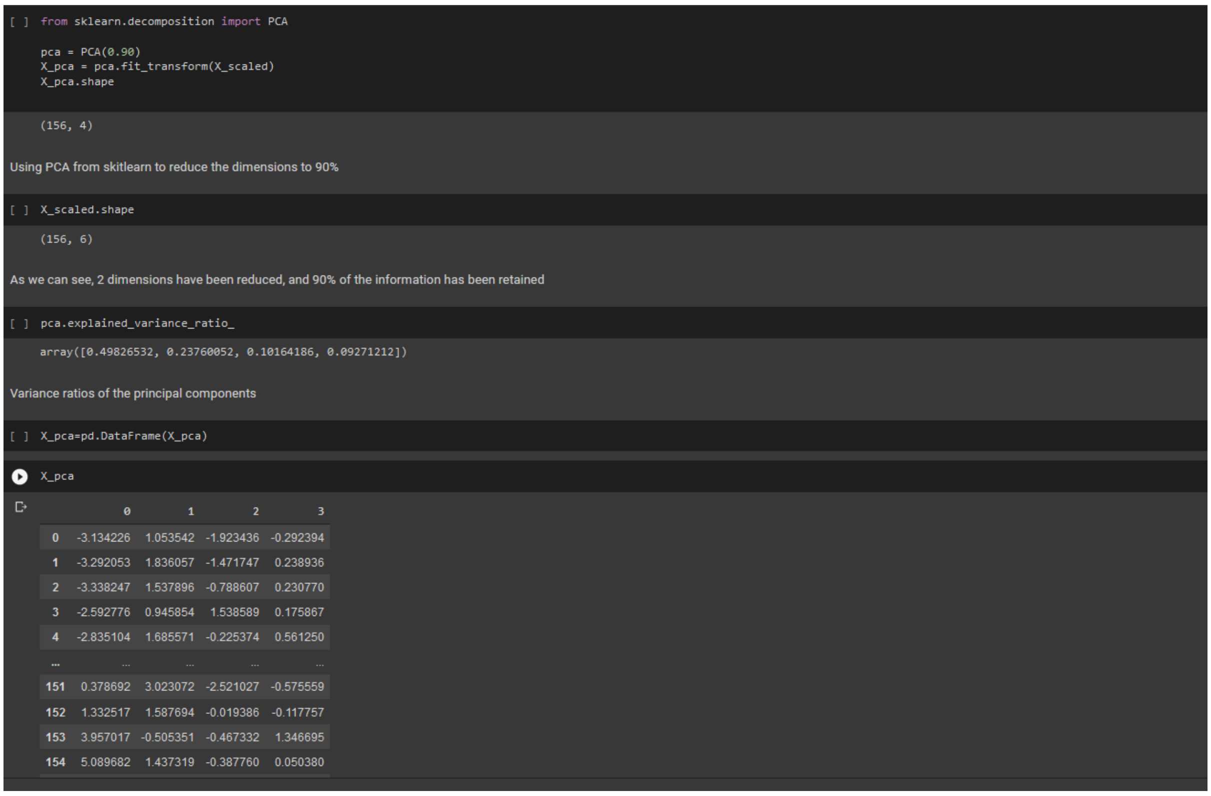


```
df1=hap.drop('Score', axis=1)
X = df1
y = df['Score']
```

Defining the Independant and dependant features as X and Y

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

```
array([[ 1.09504166e+00,  1.26809758e+00,  1.08042247e+00,
         1.42428230e+00, -3.35403977e-01,  2.99675712e+00],
       [ 1.20332385e+00,  1.22115409e+00,  1.12185663e+00,
         1.39627687e+00,  7.07264909e-01,  3.17715844e+00],
       [ 1.46773387e+00,  1.25133205e+00,  1.25444597e+00,
         1.47329179e+00,  9.07373079e-01,  2.44494130e+00],
       [ 1.19576928e+00,  1.39216252e+00,  1.24615914e+00,
         1.38927551e+00,  1.78152982e+00,  7.85004247e-02],
       [ 1.23606033e+00,  1.05014566e+00,  1.13428688e+00,
         1.15122941e+00,  1.44450554e+00,  1.98863208e+00],
       [ 1.37707901e+00,  1.06355808e+00,  1.35388797e+00,
         1.25624975e+00,  8.23117007e-01,  2.46616499e+00],
       [ 1.21339662e+00,  9.32786931e-01,  1.17572105e+00,
         1.27025246e+00,  8.65245043e-01,  2.78452027e+00],
       [ 1.00186860e+00,  1.16750438e+00,  1.24615914e+00,
         1.34726738e+00,  1.52876161e+00,  2.85880316e+00],
       [ 1.15799642e+00,  9.93142848e-01,  1.30002355e+00,
         1.34026602e+00,  1.05482120e+00,  2.09475050e+00],
       [ 1.18569652e+00,  8.92549654e-01,  1.20472497e+00,
```



```
from sklearn.decomposition import PCA

pca = PCA(0.90)
X_pca = pca.fit_transform(X_scaled)
X_pca.shape
```

```
(156, 4)
```

Using PCA from skitlearn to reduce the dimensions to 90%

```
X_scaled.shape
```

```
(156, 6)
```

As we can see, 2 dimensions have been reduced, and 90% of the information has been retained

```
pca.explained_variance_ratio_
```

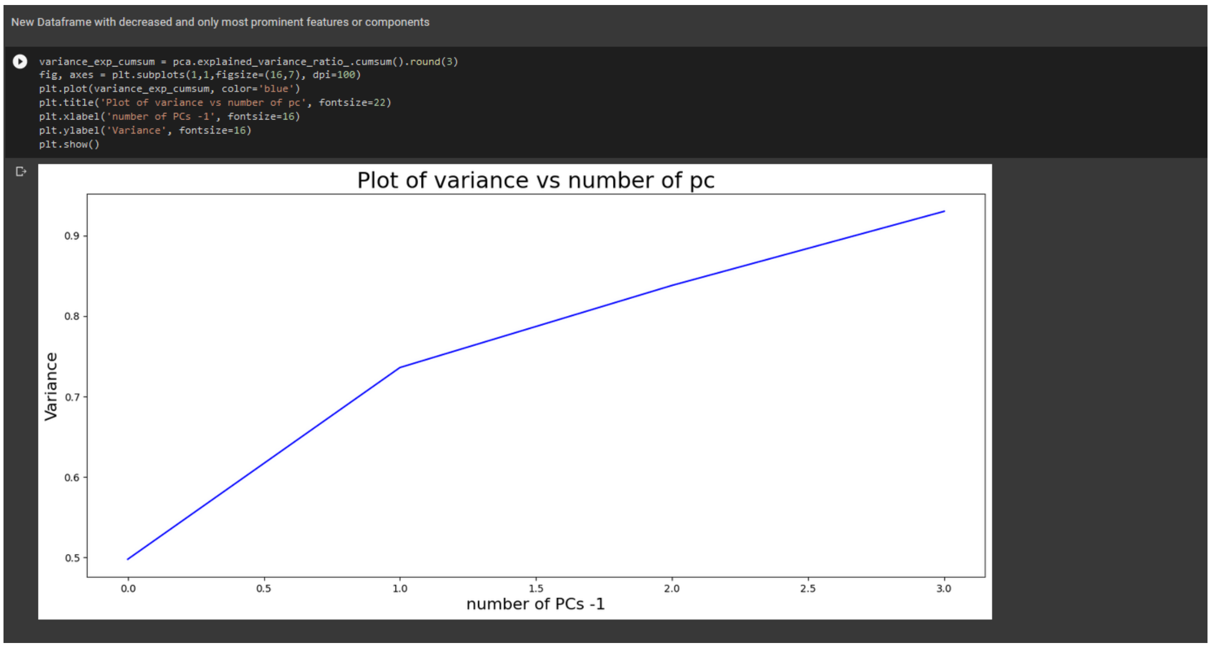
```
array([0.49826532, 0.23760052, 0.10164186, 0.09271212])
```

Variance ratios of the principal components

```
X_pca=pd.DataFrame(X_pca)
```

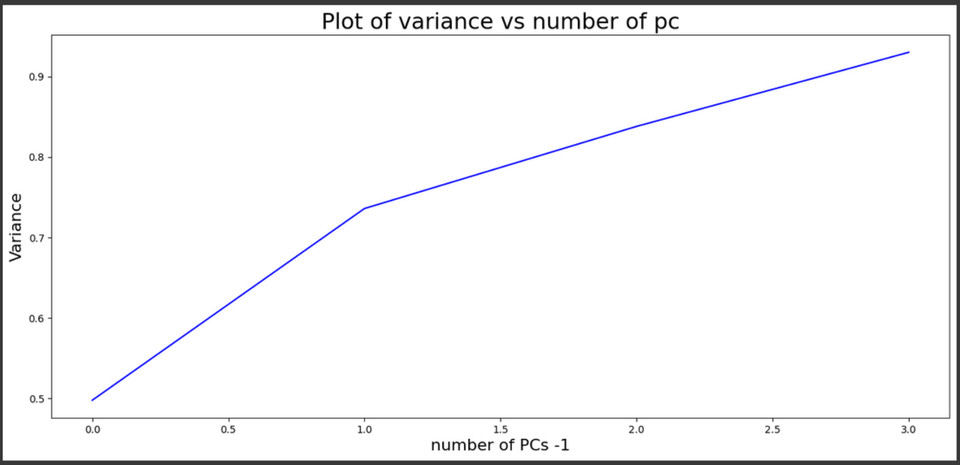
```
X_pca
```

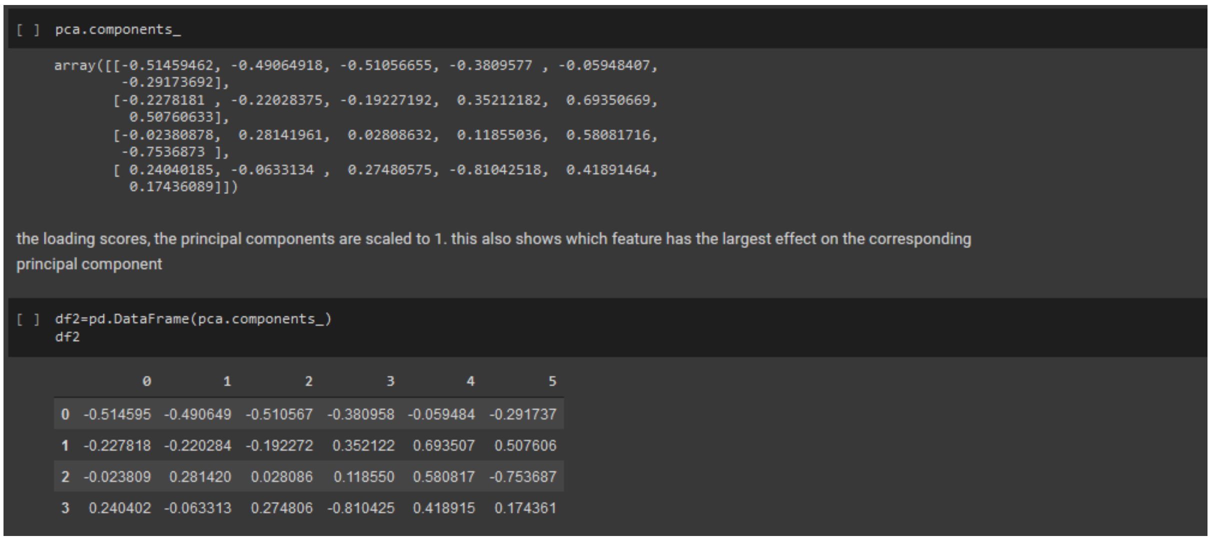
	0	1	2	3
0	-3.134226	1.053542	-1.923436	-0.292394
1	-3.292053	1.836057	-1.471747	0.238936
2	-3.338247	1.537896	-0.788607	0.230770
3	-2.592776	0.945854	1.538589	0.175867
4	-2.835104	1.685571	-0.225374	0.561250
...	...	...	...	...
151	0.378692	3.023072	-2.521027	-0.575559
152	1.332517	1.587694	-0.019386	-0.117757
153	3.957017	-0.505351	-0.467332	1.346695
154	5.089682	1.437319	-0.387760	0.050380



New Dataframe with decreased and only most prominent features or components

```
variance_exp_cumsum = pca.explained_variance_ratio_.cumsum().round(3)
fig, axes = plt.subplots(1,1,figsize=(16,7), dpi=100)
plt.plot(variance_exp_cumsum, color='blue')
plt.title('Plot of variance vs number of pc', fontsize=22)
plt.xlabel('number of PCs -1', fontsize=16)
plt.ylabel('Variance', fontsize=16)
plt.show()
```





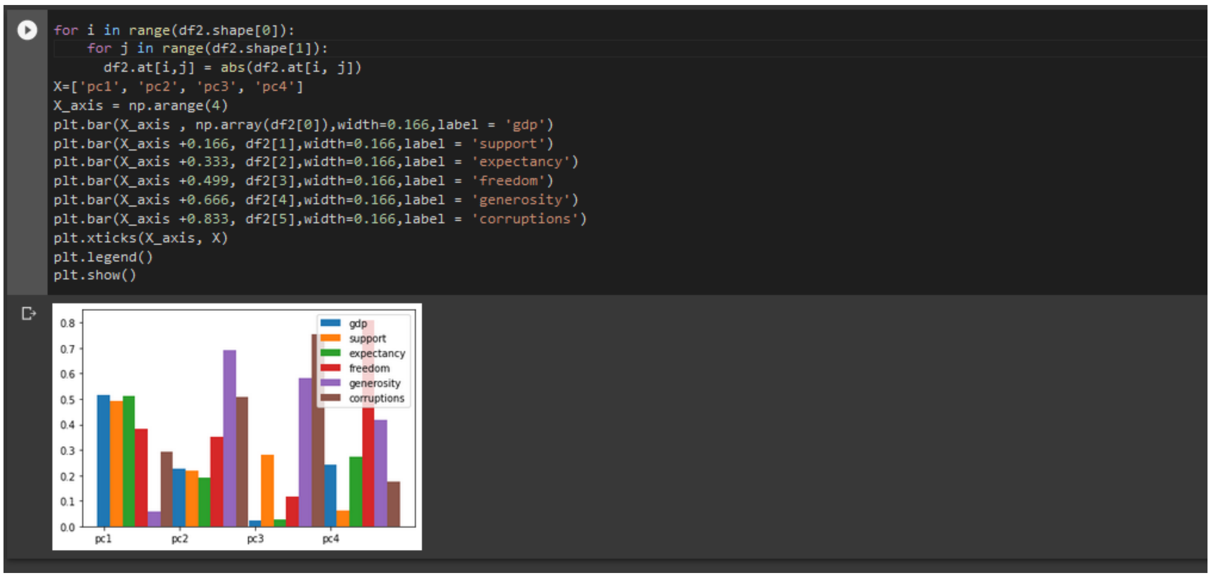
```
[ ] pca.components_
```

```
array([[-0.51459462, -0.49064918, -0.51056655, -0.3809577 , -0.05948407,
        -0.29173692],
       [-0.2278181 , -0.22028375, -0.19227192,  0.35212182,  0.69350669,
         0.50760633],
       [-0.02380878,  0.28141961,  0.02808632,  0.11855036,  0.58081716,
        -0.7536873 ],
       [ 0.24040185, -0.0633134 ,  0.27480575, -0.81042518,  0.41891464,
         0.17436089]])
```

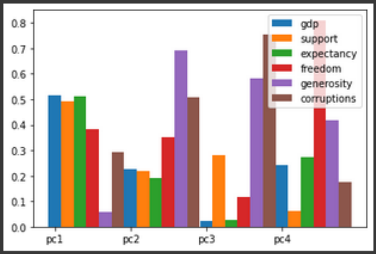
the loading scores, the principal components are scaled to 1. this also shows which feature has the largest effect on the corresponding principal component

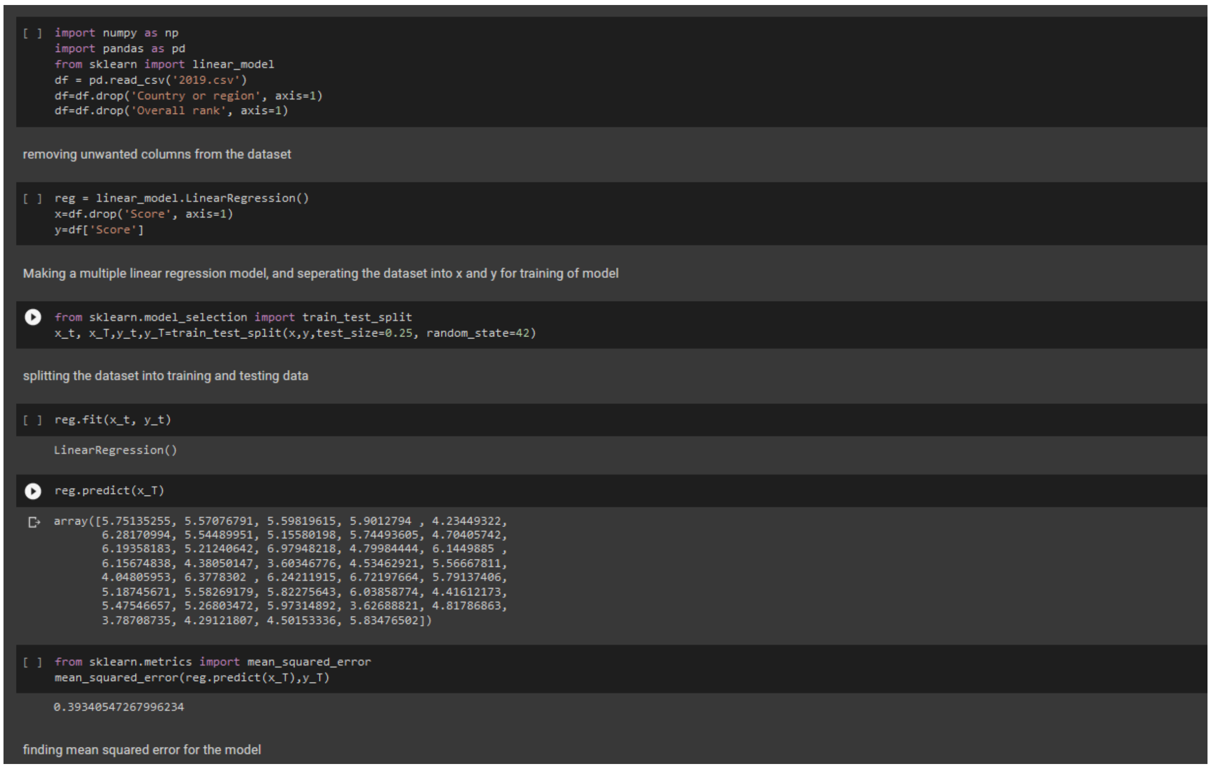
```
[ ] df2=pd.DataFrame(pca.components_)
    df2
```

	0	1	2	3	4	5
0	-0.514595	-0.490649	-0.510567	-0.380958	-0.059484	-0.291737
1	-0.227818	-0.220284	-0.192272	0.352122	0.693507	0.507606
2	-0.023809	0.281420	0.028086	0.118550	0.580817	-0.753687
3	0.240402	-0.063313	0.274806	-0.810425	0.418915	0.174361



```
for i in range(df2.shape[0]):
    for j in range(df2.shape[1]):
      df2.at[i,j] = abs(df2.at[i, j])
X=['pc1', 'pc2', 'pc3', 'pc4']
X_axis = np.arange(4)
plt.bar(X_axis , np.array(df2[0]),width=0.166,label = 'gdp')
plt.bar(X_axis +0.166, df2[1],width=0.166,label = 'support')
plt.bar(X_axis +0.333, df2[2],width=0.166,label = 'expectancy')
plt.bar(X_axis +0.499, df2[3],width=0.166,label = 'freedom')
plt.bar(X_axis +0.666, df2[4],width=0.166,label = 'generosity')
plt.bar(X_axis +0.833, df2[5],width=0.166,label = 'corruptions')
plt.xticks(X_axis, X)
plt.legend()
plt.show()
```





```
import numpy as np
import pandas as pd
from sklearn import linear_model
df = pd.read_csv('2019.csv')
df=df.drop('Country or region', axis=1)
df=df.drop('Overall rank', axis=1)
```

removing unwanted columns from the dataset

```
reg = linear_model.LinearRegression()
x=df.drop('Score', axis=1)
y=df['Score']
```

Making a multiple linear regression model, and seperating the dataset into x and y for training of model

```
from sklearn.model_selection import train_test_split
x_t, x_T,y_t,y_T=train_test_split(x,y,test_size=0.25, random_state=42)
```

splitting the dataset into training and testing data

```
reg.fit(x_t, y_t)
```

```
LinearRegression()
```

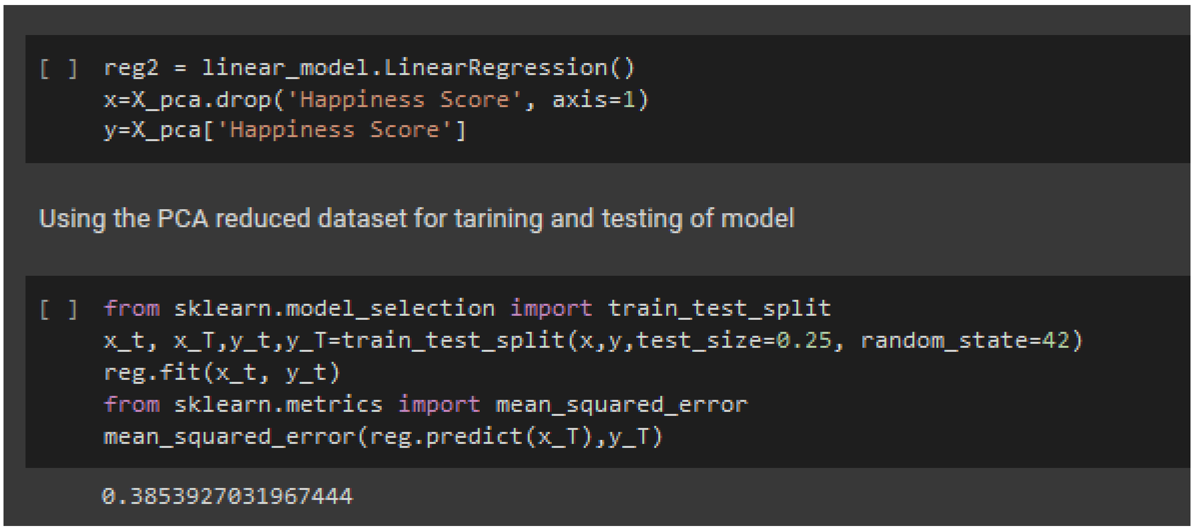
```
reg.predict(x_T)
```

```
array([5.75135255, 5.57076791, 5.59819615, 5.9012794 , 4.23449322,
       6.28170994, 5.54489951, 5.15580198, 5.74493605, 4.70405742,
       6.19358183, 5.21240642, 6.97948218, 4.79984444, 6.1449885 ,
       6.15674838, 4.38050147, 3.60346776, 4.53462921, 5.56667811,
       4.04805953, 6.3778302 , 6.24211915, 6.72197664, 5.79137406,
       5.18745671, 5.58269179, 5.82275643, 6.03858774, 4.41612173,
       5.47546657, 5.26803472, 5.97314892, 3.62688821, 4.81786863,
       3.78708735, 4.29121807, 4.50153336, 5.83476502])
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(reg.predict(x_T),y_T)
```

```
0.39340547267996234
```

finding mean squared error for the model



```
reg2 = linear_model.LinearRegression()
x=X_pca.drop('Happiness Score', axis=1)
y=X_pca['Happiness Score']
```

Using the PCA reduced dataset for tarining and testing of model

```
from sklearn.model_selection import train_test_split
x_t, x_T,y_t,y_T=train_test_split(x,y,test_size=0.25, random_state=42)
reg.fit(x_t, y_t)
from sklearn.metrics import mean_squared_error
mean_squared_error(reg.predict(x_T),y_T)
```

```
0.3853927031967444
```

REPORT:

Conclusions from each question:

question 1:

- We found the correlation between different features using scatter plots, and heatmaps
- We plotted the chloropleth maps for the different features around the globe

- We analysed the correlations of different variables and made conclusions on what variables might affect the Happiness index of a country
- We plotted box plots to see the range of values of the features of the dataset

question 2:

- We scaled the independant variables and applied PCA to see the most prominent features
- We found the loading scores of the principal components.

- We plotted the relationship between Principal components and loading scores, which helps us determine which feature affects that principal component more
- We found out that GDP, social support and Healthy life expectancy affect the happiness index the highest
- Thus the features that affct the happiness index very highly are GDP, life expectancy, and social support since they contribute a lot for the first principal component, which has the highest variation ratio of 50%. This makes sense as Money or income, Healthy life, and support from family and friends contribute to healty life

question 3:

- We can observe that the mean squared error reduces when the dataset reduced with pca is used for training the model. A model with lower Mean squared error is better and thus the model trained with the dataset with reduced dimensions by pca is better

Mean squared error for dataset without PCA analysis:
```
0.39340547267996234
```
Mean squared error for dataset with PCA analysis:
```
0.3853927031967444
```