Assignment 4

Name:Pratiksha Gaikwad
Gr.No: 21920026
RollNo: 321066
TY-Comp Batch: A3

Aim :Build a Data model in Python using any classification model (Decision Tree or Naïve Bayes) and infer the result using accuracy score .Compare different classification models (not limited to NB and DT only) with respect to feature selection and accuracy. Infer the result :which model best suit for the dataset chosen .

Outcome:

Dataset Name: World Happiness Report 2019

Overview of Dataset attribute: This dataset was created by PromptCloud and Datastock.
Each column of data has the next description.

1. **Country (region)** : Name of the country.
2. **Ladder:** is a measure of life satisfaction.
3. **SD of Ladder:** Standard deviation of the ladder.
4. **Positive affect:** Measure of positive emotion.
5. **Negative affect:** Measure of negative emotion.
6. **Social support** :The extent to which Social support contributed to the calculation of the Happiness Score.
7. **Freedom:** The extent to which Freedom contributed to the calculation of the Happiness Score.
8. **Corruption:** The extent to which Perception of Corruption contributes to Happiness Score.
9. **Generosity:** The extent to which Generosity contributed to the calculation of the Happiness Score.
10. **Log of GDP per capita:** The extent to which GDP contributes to the calculation of the Happiness Score.
11. **Healthy life expectancy:** The extent to which Life expectancy contributed to the calculation of the Happiness Score.

Output

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Assignment 4 Gr.no:21920026

Build a Data model in Python using any classification model (Decision Tree or Naïve Bayes)and infer the result using accuracy score.

In [62]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import sklearn
import warnings
from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
from sklearn.svm import SVC, LinearSVC
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
warnings.filterwarnings('ignore')
```

In [63]:
```
hpy_df = pd.read_csv('/home/pratiksha/Documents/data science/assignment 4/world-happiness-report-2019.csv')
```

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [63]:
```
hpy_df = pd.read_csv('/home/pratiksha/Documents/data science/assignment 4/world-happiness-report-2019.csv')
hpy_df.head()
```

Out[63]:

	Country (region)	Ladder	SD of Ladder	Positive affect	Negative affect	Social support	Freedom	Corruption	Generosity	Log of GDP\nper capita	Healthy life\nexpectancy
0	Finland	1	4	41.0	10.0	2.0	5.0	4.0	47.0	22.0	27.0
1	Denmark	2	13	24.0	26.0	4.0	6.0	3.0	22.0	14.0	23.0
2	Norway	3	8	16.0	29.0	3.0	3.0	8.0	11.0	7.0	12.0
3	Iceland	4	9	3.0	3.0	1.0	7.0	45.0	3.0	15.0	13.0
4	Netherlands	5	1	12.0	25.0	15.0	19.0	12.0	7.0	12.0	18.0

In [64]:
```
# cleaning dataset
hpy_df = hpy_df.drop(['Ladder', 'SD of Ladder'], axis=1)

hpy_df = hpy_df.rename(columns={
    'Country (region)':'Country',
    'Positive affect':'Pos',
    'Negative affect':'Neg',
    'Log of GDP\nper capita':'GDP',
    'Healthy life\nexpectancy':'Life expectancy'
})

hpy_df = hpy_df[~hpy_df.isnull().any(axis=1)]
hpy_df.shape

size = int(hpy_df.shape[0]/3)
size
hpy_df['Class'] = 0
hpy_df.iloc[size:2*size]['Class'] = 1
hpy_df.iloc[2*size:]['Class'] = 2
```

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [64]:
```
# cleaning dataset
hpy_df = hpy_df.drop(['Ladder', 'SD of Ladder'], axis=1)

hpy_df = hpy_df.rename(columns={
    'Country (region)':'Country',
    'Positive affect':'Pos',
    'Negative affect':'Neg',
    'Log of GDP\nper capita':'GDP',
    'Healthy life\nexpectancy':'Life expectancy'
})

hpy_df = hpy_df[~hpy_df.isnull().any(axis=1)]
hpy_df.shape

size = int(hpy_df.shape[0]/3)
size
hpy_df['Class'] = 0
hpy_df.iloc[size:2*size]['Class'] = 1
hpy_df.iloc[2*size:]['Class'] = 2
```

In [65]:
```
# Reseting index since some samples were dropped before that a few numbers skip
hpy_df.index = np.arange(hpy_df.shape[0])
```

In [66]:
```
# Randomly choosing testing samples
happy_idx = np.random.choice(np.arange(size), size=5, replace=False)
neutral_idx = np.random.choice(np.arange(size, 2*size), size=5, replace=False)
sad_idx = np.random.choice(np.arange(2*size, hpy_df.shape[0]), size=5, replace=False)

test_idx = list(happy_idx) + list(neutral_idx) + list(sad_idx)
```

In [67]:
```
test = hpy_df.iloc[test_idx]
test
```

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [66]:
```
# Randomly choosing testing samples
happy_idx = np.random.choice(np.arange(size), size=5, replace=False)
neutral_idx = np.random.choice(np.arange(size, 2*size), size=5, replace=False)
sad_idx = np.random.choice(np.arange(2*size, hpy_df.shape[0]), size=5, replace=False)

test_idx = list(happy_idx) + list(neutral_idx) + list(sad_idx)
```

In [67]:
```
test = hpy_df.iloc[test_idx]
test
```

Out[67]:

	Country	Pos	Neg	Social support	Freedom	Corruption	Generosity	GDP	Life expectancy	Class
34	Poland	76.0	33.0	44.0	52.0	108.0	77.0	41.0	36.0	0
2	Norway	16.0	29.0	3.0	3.0	8.0	11.0	7.0	12.0	0
19	Czech Republic	74.0	22.0	24.0	58.0	121.0	117.0	32.0	31.0	0
9	Austria	64.0	24.0	31.0	26.0	19.0	25.0	16.0	15.0	0
8	Canada	18.0	49.0	20.0	9.0	11.0	14.0	19.0	8.0	0
84	Cameroon	106.0	129.0	129.0	90.0	120.0	91.0	121.0	141.0	1
50	Japan	73.0	14.0	50.0	64.0	39.0	92.0	24.0	2.0	1
64	Montenegro	143.0	118.0	60.0	139.0	77.0	76.0	61.0	44.0	1
54	Hungary	86.0	31.0	51.0	138.0	140.0	100.0	42.0	56.0	1
89	Benin	118.0	148.0	153.0	103.0	75.0	116.0	128.0	133.0	1
132	Botswana	87.0	65.0	105.0	60.0	54.0	150.0	66.0	113.0	2
137	Afghanistan	152.0	133.0	151.0	155.0	136.0	137.0	134.0	139.0	2
115	Sri Lanka	32.0	81.0	80.0	55.0	111.0	35.0	79.0	54.0	2
127	Madagascar	46.0	96.0	128.0	146.0	116.0	136.0	144.0	111.0	2
109	Tunisia	147.0	132.0	121.0	143.0	101.0	144.0	84.0	67.0	2

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

127	Madagascar	46.0	96.0	128.0	146.0	116.0	136.0	144.0	111.0	2
109	Tunisia	147.0	132.0	121.0	143.0	101.0	144.0	84.0	67.0	2

In [68]:
```
train = hpy_df[~hpy_df.index.isin(test_idx)]
train.shape, test.shape
```

Out[68]:
```
((125, 10), (15, 10))
```

In [69]:
```
def split_data(dat):
    X = dat.loc[:, ['Social support', 'GDP', 'Life expectancy']]
    y = dat.loc[:, 'Class']
    return X, y
```

In [70]:
```
# useing three features
X_train, y_train = split_data(train)
X_test, y_test = split_data(test)
```

In [71]:
```
# Set random_state for reproducibility
clf = DecisionTreeClassifier(random_state=123)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[71]:
```
0.6
```

In [72]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_predict = decision_tree.predict(X_test)
value_dt= round(decision_tree.score(X_train, y_train) * 100, 2)
value_dt
```

Out[72]:
```
100.0
```

localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [73]:
```
X_train, y_train = train.drop(['Class', 'Country'], axis=1), train.loc[:, 'Class']
X_test, y_test = test.drop(['Class', 'Country'], axis=1), test.loc[:, 'Class']
```

In [74]:
```
# Decision tree classifier
clf2 = DecisionTreeClassifier(random_state=123)
clf2.fit(X_train, y_train)
clf2.score(X_test, y_test)
```

Out[74]:
```
0.6
```

In [75]:
```
# by sort the features by descending order based on its importance to Class will have the following result
feature_importances = np.stack((clf2.feature_importances_, list(X_train)), axis=1)
feature_importances = feature_importances[feature_importances.argsort(axis=0)[:, 0]][::-1]
feature_importances
```

Out[75]:
```
array([['0.3311319955973244', 'Social support'],
       ['0.23196727282304822', 'Life expectancy'],
       ['0.1330651757961149', 'GDP'],
       ['0.1274948103221641', 'Pos'],
       ['0.08868898339710374', 'Freedom'],
       ['0.04481412896350422', 'Corruption'],
       ['0.04283763310074046', 'Generosity'],
       ['0.0', 'Neg']], dtype='<U32')
```

In [76]:
```
# decision tree in loop for all feature to calculate important feature score
scores = []
for i in range(1, len(feature_importances)+1):
    features = feature_importances[:i, 1]
    clf = DecisionTreeClassifier(random_state=123)
    clf.fit(X_train.loc[:, features], y_train)
    scores.append(clf.score(X_test.loc[:, features], y_test))
```

In [77]:
```
plt.figure(figsize=(8, 6))
```

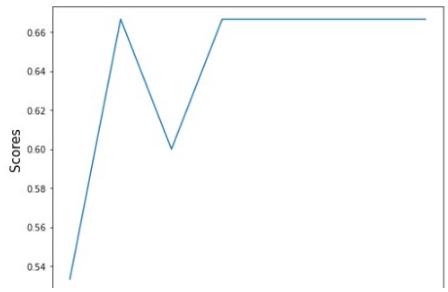
localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [76]:
```
# decision tree in loop for all feature to calculate important feature score
scores = []
for i in range(1, len(feature_importances)+1):
    features = feature_importances[:i, 1]
    clf = DecisionTreeClassifier(random_state=123)
    clf.fit(X_train.loc[:, features], y_train)
    scores.append(clf.score(X_test.loc[:, features], y_test))
```

In [77]:
```
plt.figure(figsize=(8, 6))
plt.plot(scores)
plt.xlabel('Numer of Features', size=15)
plt.ylabel('Scores', size=15)
plt.show();
```



localhost:8888/notebooks/Downloads/ds4%20(1).ipynb

jupyter ds4 (1) Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [78]:
```
feature_importances
```

Out[78]:
```
array([['0.3311319955973244', 'Social support'],
       ['0.23196727282304822', 'Life expectancy'],
       ['0.1330651757961149', 'GDP'],
       ['0.1274948103221641', 'Pos'],
       ['0.08868898339710374', 'Freedom'],
       ['0.04481412896350422', 'Corruption'],
       ['0.04283763310074046', 'Generosity'],
       ['0.0', 'Neg']], dtype='<U32')
```

With the above graph, we could use 3, 4, 5, or 6 features and still have the same score

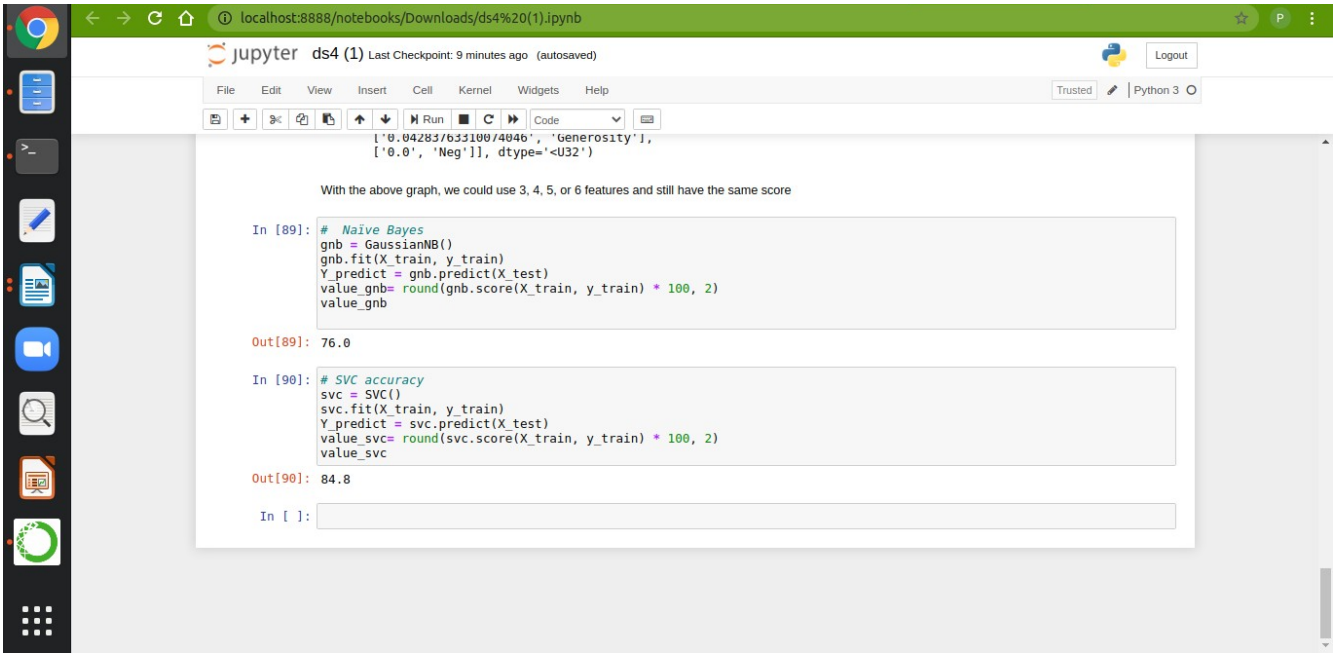
In [89]:
```
#  Naive Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
Y_predict = gnb.predict(X_test)
value_gnb= round(gnb.score(X_train, y_train) * 100, 2)
value_gnb
```

Out[89]: 76.0

In [90]:
```
# SVC accuracy
svc = SVC()
svc.fit(X_train, y_train)
Y_predict = svc.predict(X_test)
value_svc= round(svc.score(X_train, y_train) * 100, 2)
value_svc
```

Out[90]: 84.8

In [ ]:



Description:

- **Classification:** Classification is the process of predicting the class of given data points.
- **Classification model:** A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- Classification **Accuracy:** It is the ratio of number of correct predictions to the total number of input samples.
- **Performance Metrics :** It aims to model the relationship between a certain number of features and a continuous target variable.
- In my dataset I have created new ‘class’ (0-happy, 1-neutral and 2 is sad) feature by using that building a model which predicts based on other features.

Following classification models are used

1) SVC (Support Vector Classifier)

- The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data.
- As classes are created (0-happy, 1-neutral and 2 is sad), random samples of class and country attributes are used for testing and traning.
- Comparison of actual test set values and predicted values or accuracy score is **84** of given dataset.

2) decision trees classification

- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- By using this finding important feature according to score and also plot graph for same and identifying what are the important feature are there.
- ‘Class’ and ‘country’ these 2 and 'Social support', 'GDP', 'Life expectancy' with ‘class’ attribute used in traing and testing randomly.
- By using this classification model 0.6 accuracy score we got.

3) Naive Bayes classification

- Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features.
- Random samples of Class and other attribute of dataset is taken for traning and testing.
- By using this model 76.0 accuracy score we got.

Interpretation:

- According to decision trees classification accuracy score ‘Social support', 'Life expectancy' , 'GDP’, 'Positivity’ these feature plays important role in happiness of country.
- SVC (Support Vector Classifier) model gives good accuracy i.e. 84.4% for given dataset and it could be best fit for dataset.
- Naive Bayes gives 76 % accuracy which is also good for dataset.