Happiness Score 3

October 2, 2019

In [36]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error
data = pd.read_csv('2015.csv')
data2 = pd.read_csv('2016.csv')
data3 = pd.read_csv('2017.csv')
```

In [37]:
```
# data.append(data2, ignore_index = True)
# data.append(data3, ignore_index = True)
data.drop(['Happiness Rank','Country','Region','Standard Error'], axis = 1, inplace =
data.iloc[0]
```

Out[37]:
```
Happiness Score                  7.58700
Economy (GDP per Capita)         1.39651
Family                           1.34951
Health (Life Expectancy)         0.94143
Freedom                          0.66557
Trust (Government Corruption)    0.41978
Generosity                       0.29678
Dystopia Residual                2.51738
Name: 0, dtype: float64
```

In [38]:
```
data2.drop(['Happiness Rank','Country','Region','Lower Confidence Interval', 'Upper Co
data2.iloc[0]
```

Out[38]:
```
Happiness Score                  7.52600
Economy (GDP per Capita)         1.44178
Family                           1.16374
Health (Life Expectancy)         0.79504
Freedom                          0.57941
Trust (Government Corruption)    0.44453
Generosity                       0.36171
Dystopia Residual                2.73939
Name: 0, dtype: float64
```

In [39]:
```
data3.drop(['Happiness.Rank','Country','Whisker.high', 'Whisker.low'], axis = 1, inpla
data3.iloc[0]
```

```
Out[39]: Happiness.Score                  7.537000
         Economy..GDP.per.Capita.         1.616463
         Family                           1.533524
         Health..Life.Expectancy.         0.796667
         Freedom                          0.635423
         Generosity                       0.362012
         Trust..Government.Corruption.    0.315964
         Dystopia.Residual                2.277027
         Name: 0, dtype: float64
```

```
In [40]: len(data3)
```

```
Out[40]: 155
```

```
In [41]: data.columns = ['Happiness Score','Economy','Family','Health','Freedom','Trust','Gene
         data2.columns = ['Happiness Score','Economy','Family','Health','Freedom','Trust','Gen
         data3.columns = ['Happiness Score','Economy','Family','Health','Freedom','Trust','Gen

         data = data.append(data2)
         print(len(data))
         data = data.append(data3)
         print(len(data))
```

```
315
470
```

```
In [42]: data.head()
```

```
Out[42]:    Happiness Score  Economy   Family   Health  Freedom    Trust  Generosity  \
         0            7.587  1.39651  1.34951  0.94143  0.66557  0.41978     0.29678
         1            7.561  1.30232  1.40223  0.94784  0.62877  0.14145     0.43630
         2            7.527  1.32548  1.36058  0.87464  0.64938  0.48357     0.34139
         3            7.522  1.45900  1.33095  0.88521  0.66973  0.36503     0.34699
         4            7.427  1.32629  1.32261  0.90563  0.63297  0.32957     0.45811

            Dystopia
         0   2.51738
         1   2.70201
         2   2.49204
         3   2.46531
         4   2.45176
```

```
In [43]: data_13 = data.iloc[:,0:4]
         data_13.head()
         data_22 = data.iloc[:,[0,1,2]]
         data_22.head()
         data_32 = data.iloc[:,[0,6,7]]
         data_32.head()
         data_47 = data.iloc[:,0:7]
         data_47.head()
```

Out[43]:

	Happiness Score	Economy	Family	Health	Freedom	Trust	Generosity
0	7.587	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	7.522	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811

In [44]:
```
features_13 = data_13.drop(['Happiness Score'], axis = 1)
features_22 = data_22.drop(['Happiness Score'], axis = 1)
features_32 = data_32.drop(['Happiness Score'], axis = 1)
features_47 = data_47.drop(['Happiness Score'], axis = 1)
target = data['Happiness Score']
```

In [45]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
reg_13 = LinearRegression().fit(features_13, target)

y_pred_13 = reg_13.predict(features_13)
mean_squared_error(target, y_pred_13)
```

Out[45]: 0.383132414874312\93

In [46]:
```
reg_22 = LinearRegression().fit(features_22, target)

y_pred_22 = reg_22.predict(features_22)
mean_squared_error(target, y_pred_22)
```

Out[46]: 0.4328876930949258

In [47]:
```
reg_32 = LinearRegression().fit(features_32, target)

y_pred_32 = reg_32.predict(features_32)
mean_squared_error(target, y_pred_32)
```

Out[47]: 0.9297629730704532

In [48]:
```
reg_47 = LinearRegression().fit(features_47, target)

y_pred_47 = reg_47.predict(features_47)
mean_squared_error(target, y_pred_47)
```

Out[48]: 0.30197336216035153

In [49]:
```
# MLP model
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor(hidden_layer_sizes=(100),max_iter=200000)
```

In [50]:
```
#Multi-layer Perceptron is sensitive to feature scaling, so it is highly recommended t
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
scaler.fit(features_13)
# Now apply the transformations to the data:
mlp_features_13 = scaler.transform(features_13)
mlp.fit(mlp_features_13, target)
```

Out[50]:
```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=100, learning_rate='constant',
       learning_rate_init=0.001, max_iter=200000, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

In [51]:
```
mlp_pred_13 = mlp.predict(mlp_features_13)
mean_squared_error(target, mlp_pred_13)
```

Out[51]: 0.3155719175515372

In [52]:
```
scaler.fit(features_22)
# Now apply the transformations to the data:
mlp_features_22 = scaler.transform(features_22)
mlp.fit(mlp_features_22, target)
mlp_pred_22 = mlp.predict(mlp_features_22)
mean_squared_error(target, mlp_pred_22)
```

Out[52]: 0.4121454745983672

In [53]:
```
scaler.fit(features_32)
# Now apply the transformations to the data:
mlp_features_32 = scaler.transform(features_32)
mlp.fit(mlp_features_32, target)
mlp_pred_32 = mlp.predict(mlp_features_32)
mean_squared_error(target, mlp_pred_32)
```

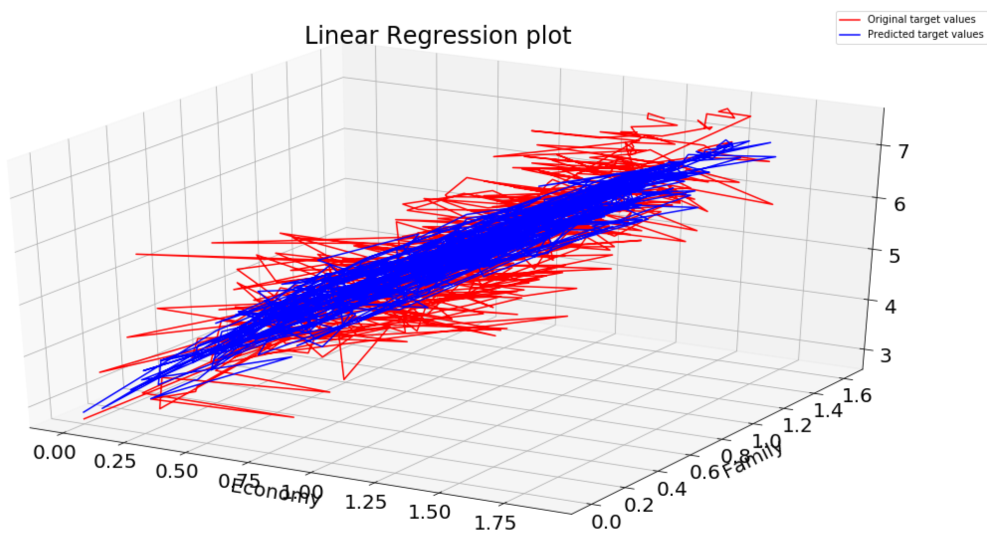
Out[53]: 0.8125211948250888

In [54]:
```
scaler.fit(features_47)
# Now apply the transformations to the data:
mlp_features_47 = scaler.transform(features_47)
mlp.fit(mlp_features_47, target)
mlp_pred_47 = mlp.predict(mlp_features_47)
mean_squared_error(target, mlp_pred_47)
```

Out[54]: 0.16937233634659382

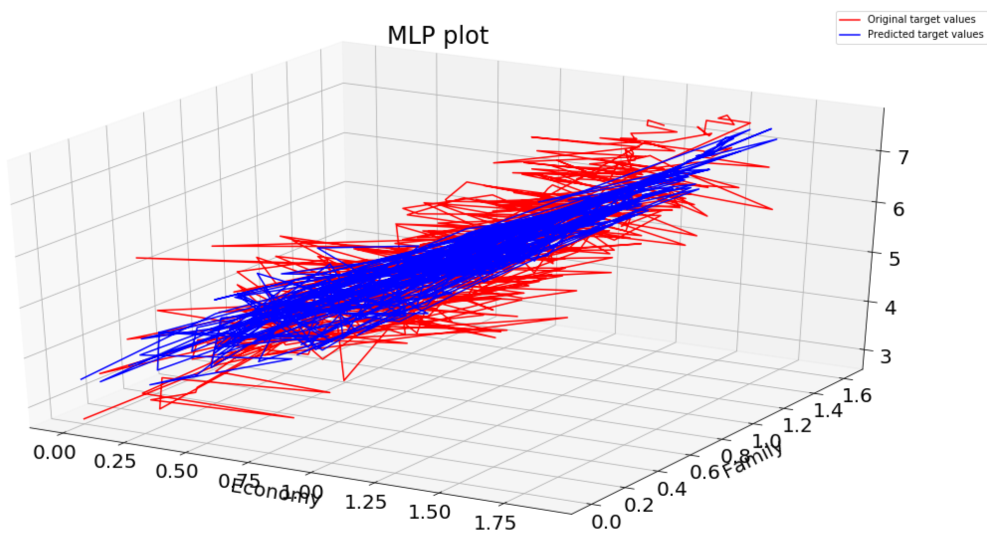
In [55]:
```
#Plotting training and predicted target values for 2 features selected using Linear R
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

```
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 20})

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure(figsize=(20,10))
ax = fig.gca(projection='3d')

z = target
z1 = y_pred_22
x = data.iloc[:,1]
y = data.iloc[:,2]

ax.plot(x, y, z,'r', label='Original target values')
ax.plot(x, y, z1,'b', label='Predicted target values')
ax.legend()
plt.xlabel('Economy')
plt.ylabel('Family')
plt.title('Linear Regression plot')

plt.show()
```




```
In [56]: #Plotting training and predicted target values for 2 features selected using MLP mode
         import matplotlib as mpl
         from mpl_toolkits.mplot3d import Axes3D
         import numpy as np
         import matplotlib.pyplot as plt
         plt.rcParams.update({'font.size': 20})
```

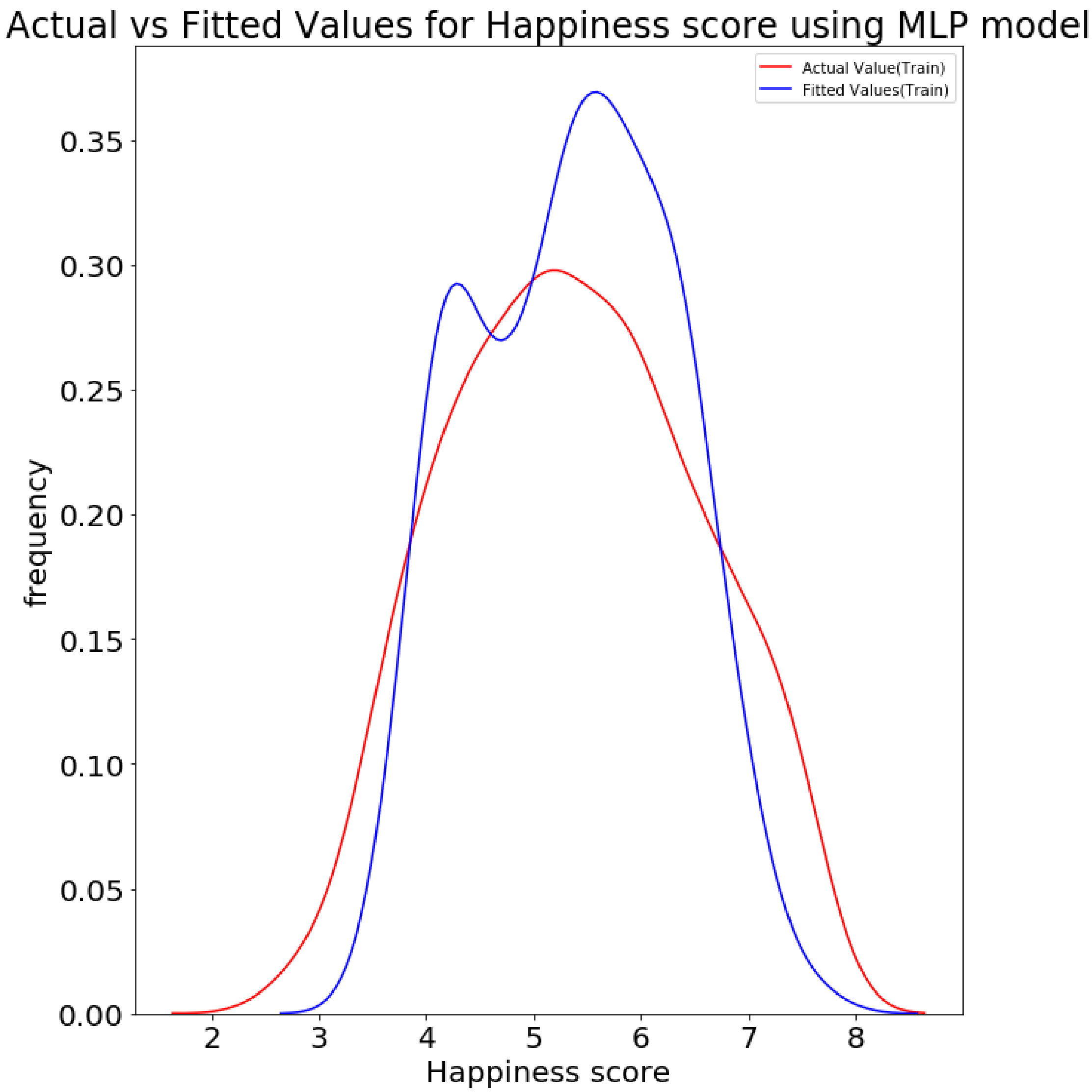
```
mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure(figsize=(20,10))
ax = fig.gca(projection='3d')

z = target
z1 = mlp_pred_22
x = data.iloc[:,1]
y = data.iloc[:,2]

ax.plot(x, y, z,'r', label='Original target values')
ax.plot(x, y, z1,'b', label='Predicted target values')
ax.legend()
plt.xlabel('Economy')
plt.ylabel('Family')
plt.title('MLP plot')

plt.show()
```



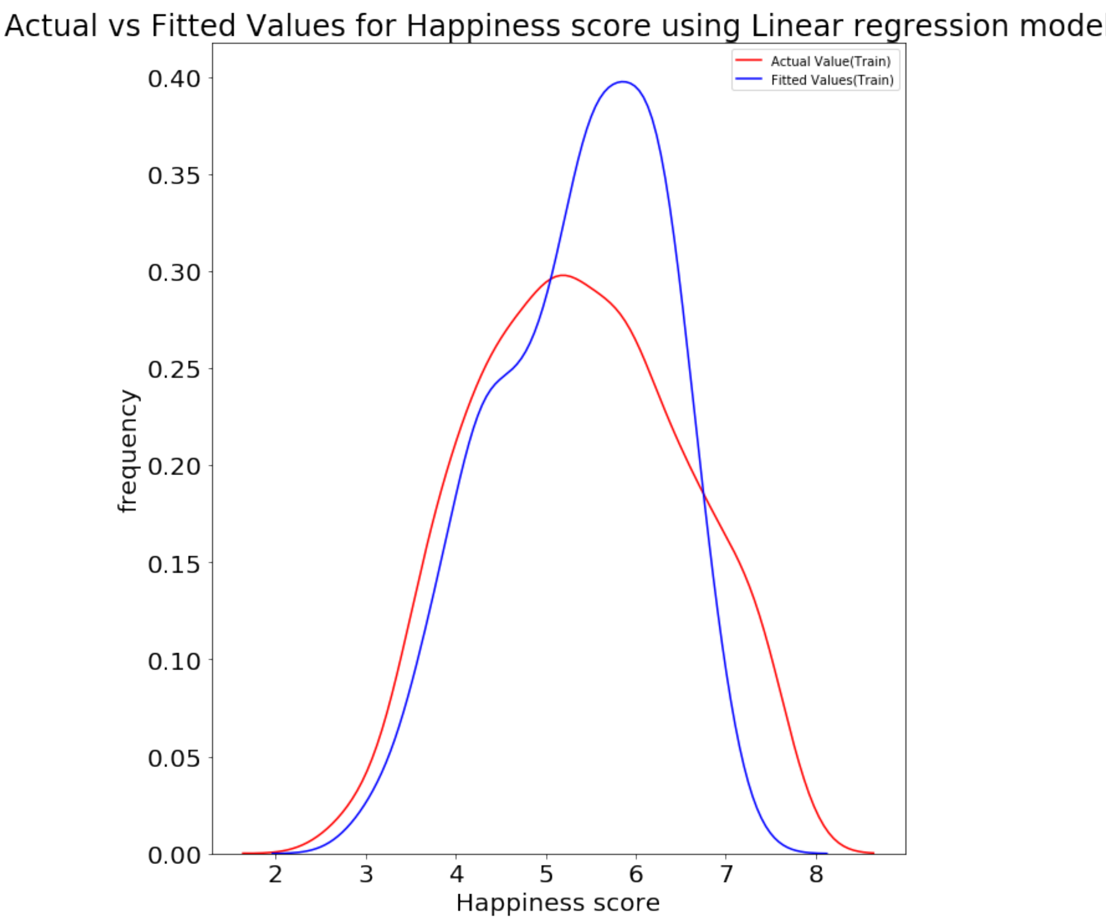
0.1 Distribution plot between train data and predicted data

```
In [58]: import seaborn as sns
         plt.figure(figsize=(10, 12))
         ax1 = sns.distplot(z, hist=False, color="r", label="Actual Value(Train)")
         sns.distplot(z1, hist=False, color="b", label="Fitted Values(Train)" , ax=ax1)
         plt.title('Actual vs Fitted Values for Happiness score using MLP model')
         plt.xlabel('Happiness score')
         plt.ylabel('frequency')
```

```
plt.show()
plt.close()
```




```
In [59]: import seaborn as sns
         plt.figure(figsize=(10, 12))
         ax1 = sns.distplot(target, hist=False, color="r", label="Actual Value(Train)")
         sns.distplot(y_pred_22, hist=False, color="b", label="Fitted Values(Train)" , ax=ax1)
         plt.title('Actual vs Fitted Values for Happiness score using Linear regression model')
         plt.xlabel('Happiness score')
         plt.ylabel('frequency')
         plt.show()
         plt.close()
```



In [63]:
```
target = data.iloc[:,0]
features = data.iloc[:,1:]
print(target.head())
features.head()
```

```
0    7.587
1    7.561
2    7.527
3    7.522
4    7.427
Name: Happiness Score, dtype: float64
```

Out[63]:

	Economy	Family	Health	Freedom	Trust	Generosity	Dystopia
0	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176

```
In [64]: reg = LinearRegression().fit(features, target)
         y_pred = reg.predict(features)
         mean_squared_error(target, y_pred)
```

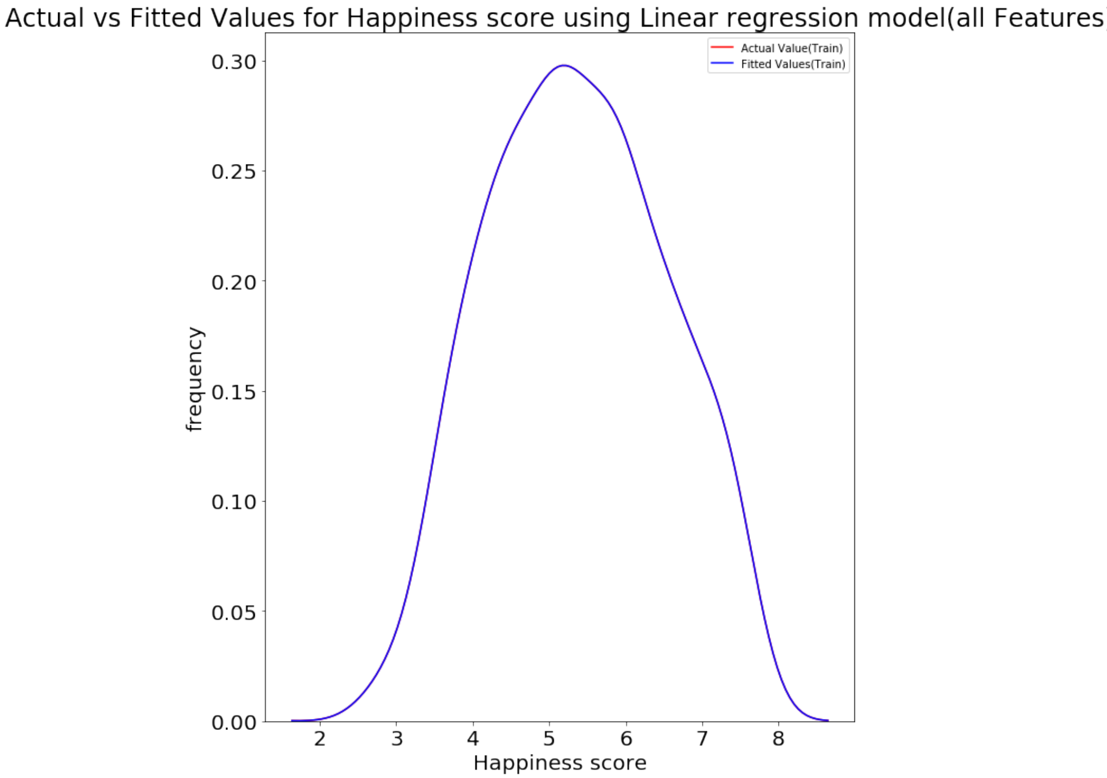
```
Out[64]: 8.1869445091806e-08
```

```
In [66]: mlp = MLPRegressor(hidden_layer_sizes=(100),max_iter=200000)
         scaler = StandardScaler()
         scaler.fit(features)
         # Now apply the transformations to the data:
         mlp_features = scaler.transform(features)
         mlp.fit(mlp_features, target)
         mlp_pred = mlp.predict(mlp_features)
         mean_squared_error(target, mlp_pred)
```

```
Out[66]: 0.017458290440188224
```

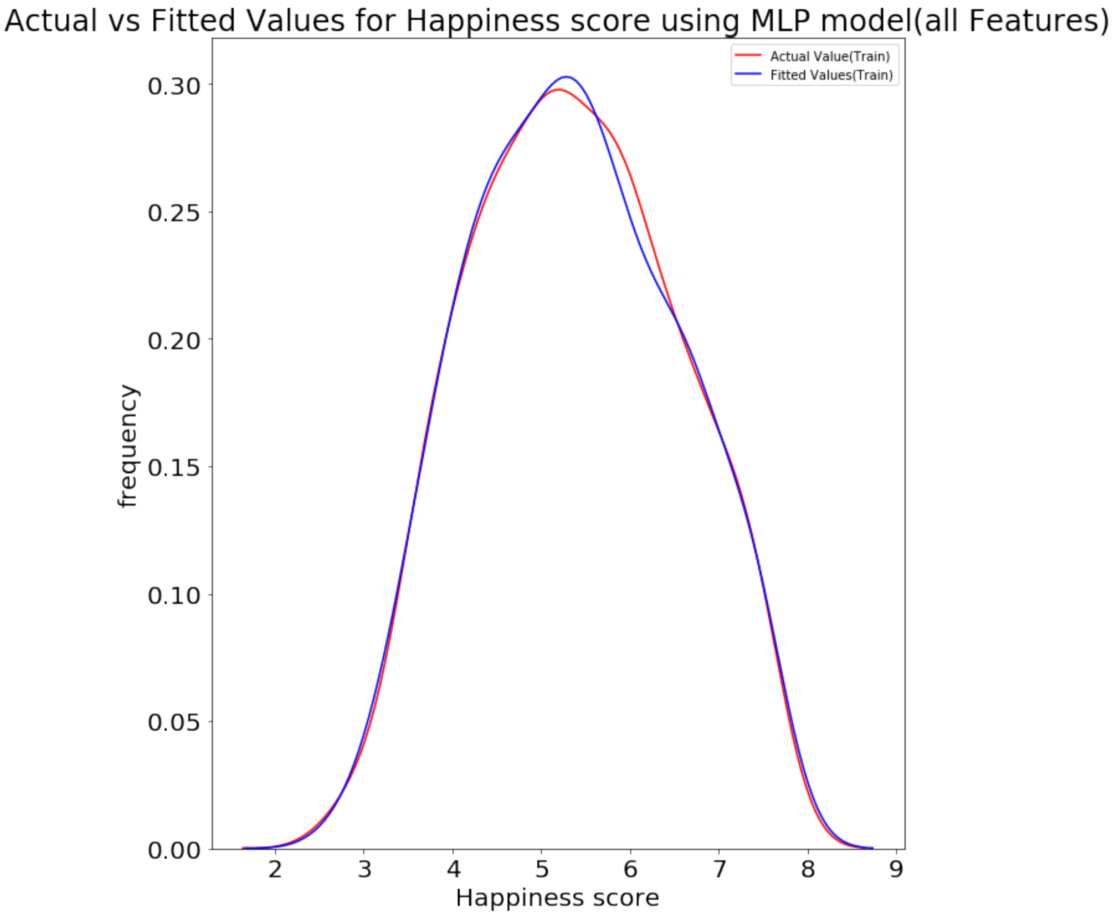
```
In [65]: plt.figure(figsize=(10, 12))
         ax1 = sns.distplot(target, hist=False, color="r", label="Actual Value(Train)")
         sns.distplot(y_pred, hist=False, color="b", label="Fitted Values(Train)" , ax=ax1)
         plt.title('Actual vs Fitted Values for Happiness score using Linear regression model(a
         plt.xlabel('Happiness score')
         plt.ylabel('frequency')
         plt.show()
         plt.close()
```

```
C:\Users\Welcome\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [67]: plt.figure(figsize=(10, 12))
         ax1 = sns.distplot(target, hist=False, color="r", label="Actual Value(Train)")
         sns.distplot(mlp_pred, hist=False, color="b", label="Fitted Values(Train)" , ax=ax1)
         plt.title('Actual vs Fitted Values for Happiness score using MLP model(all Features)')
         plt.xlabel('Happiness score')
         plt.ylabel('frequency')
         plt.show()
         plt.close()
```

```
C:\Users\Welcome\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Actual vs Fitted Values for Happiness score using MLP model(all Features)
Actual Value(Train)
Fitted Values(Train)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
frequency
2
3
4
5
6
7
8
9
Happiness score


```
In [ ]:
```