In [1]:
```
import numpy as np #linear algebra
import pandas as pd #data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import xgboost as xgb
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_squared_log_error
from sklearn.neural_network import MLPRegressor
from math import sqrt
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
```

In [2]:
```
data = pd.read_csv('C:/Users/HP-PC/Desktop/GDP data.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	Year	Exports (Million Pounds)	Short term interest rate (% per annum)	Long term interest rate (% per annum)	Exchange rate (Pounds)	Government Expenditure(Million pounds)	Unemployment rate (% of labour force)	BCI	CCI	Private consumption (%)
0	1971	141149.388	6.632173	7.868333	0.410920	19294	3.5	NaN	NaN	3.453198
1	1972	142667.420	4.632531	8.375000	0.400390	22028	3.8	NaN	NaN	6.537888
2	1973	160223.232	4.225272	10.558330	0.408171	25496	2.7	NaN	NaN	5.837593
3	1974	171940.600	13.873965	14.206670	0.427756	32665	2.6	NaN	97.59261	-1.153243
4	1975	166845.532	17.298897	13.181670	0.452041	43234	4.2	NaN	95.55269	-0.238090

In [348…
```
## let us show the UK_GDP Columns
```

In [349…
```
data.columns
```

Out[349…
```
Index(['Year', 'Exports (Million Pounds)',
       'Short term interest rate (% per annum)',
       'Long term interest rate (% per annum)', 'Exchange rate (Pounds)',
       'Government Expenditure( Million pounds)',
       'Unemployment rate (% of labour force)', 'BCI', 'CCI',
       'Private consumption (%)', 'GDP (Million Pounds)'],
      dtype='object')
```

In [350…
```
## Inspecting Missing Values
```

In [351…
```
print('Number of Missing Values:')
print(data.isnull().sum())
```

```
Number of Missing Values:
Year                                       0
Exports (Million Pounds)                   0
Short term interest rate (% per annum)     0
Long term interest rate (% per annum)      0
Exchange rate (Pounds)                     0
Government Expenditure( Million pounds)    0
Unemployment rate (% of labour force)      0
BCI                                        6
CCI                                        3
Private consumption (%)                    0
GDP (Million Pounds)                       0
dtype: int64
```

In [352…

```
data.dtypes
```

Out[352…

```
Year                                          int64
Exports (Million Pounds)                    float64
Short term interest rate (% per annum)      float64
Long term interest rate (% per annum)       float64
Exchange rate (Pounds)                      float64
Government Expenditure( Million pounds)       int64
Unemployment rate (% of labour force)       float64
BCI                                         float64
CCI                                         float64
Private consumption (%)                     float64
GDP (Million Pounds)                        float64
dtype: object
```

In [353…

```
data[['Exports (Million Pounds)']]= data[['Exports (Million Pounds)']].astype("int")
```

In [354…

```
data.dtypes
```

Out[354…

```
Year                                          int64
Exports (Million Pounds)                      int32
Short term interest rate (% per annum)      float64
Long term interest rate (% per annum)       float64
Exchange rate (Pounds)                      float64
Government Expenditure( Million pounds)       int64
Unemployment rate (% of labour force)       float64
BCI                                         float64
CCI                                         float64
Private consumption (%)                     float64
GDP (Million Pounds)                        float64
dtype: object
```

In [355…

```
data.describe(include='all').transpose()
```

Out[355…

	count	mean	std	min	25%	50%	75%
Year	46.0	1.993500e+03	13.422618	1971.000000	1982.250000	1.993500e+03	2.004750e+03
Exports (Million Pounds)	46.0	4.101662e+05	214220.636958	141149.000000	207304.000000	3.324475e+05	6.153792e+05
Short term interest rate (% per annum)	46.0	6.649844e+00	4.073912	0.498992	4.633853	6.270524e+00	8.992842e+00
Long term interest rate (% per annum)	46.0	7.902362e+00	3.897396	1.305208	4.663610	7.995216e+00	1.108812e+01
Exchange rate (Pounds)	46.0	5.873132e-01	0.090920	0.400390	0.544519	6.092830e-01	6.458640e-01
Government Expenditure(Million pounds)	46.0	3.137662e+05	230898.397886	19294.000000	119913.500000	2.865855e+05	4.966230e+05
Unemployment rate (% of labour force)	46.0	7.078629e+00	2.497860	2.600000	5.342839	6.544225e+00	8.562500e+00
BCI	40.0	9.993644e+01	2.176294	94.083970	98.808880	1.004413e+02	1.011878e+02
CCI	43.0	1.000122e+02	2.273243	95.552690	98.363430	1.008833e+02	1.015478e+02
Private consumption (%)	46.0	2.763605e+00	2.405090	-2.733007	1.400176	3.097470e+00	4.365271e+00
GDP (Million Pounds)	46.0	1.280067e+06	804887.149833	218729.687000	581117.138500	1.103986e+06	1.954913e+06

Loading [MathJax]/extensions/Safe.js

In [356…
```
# Filling missing values
```

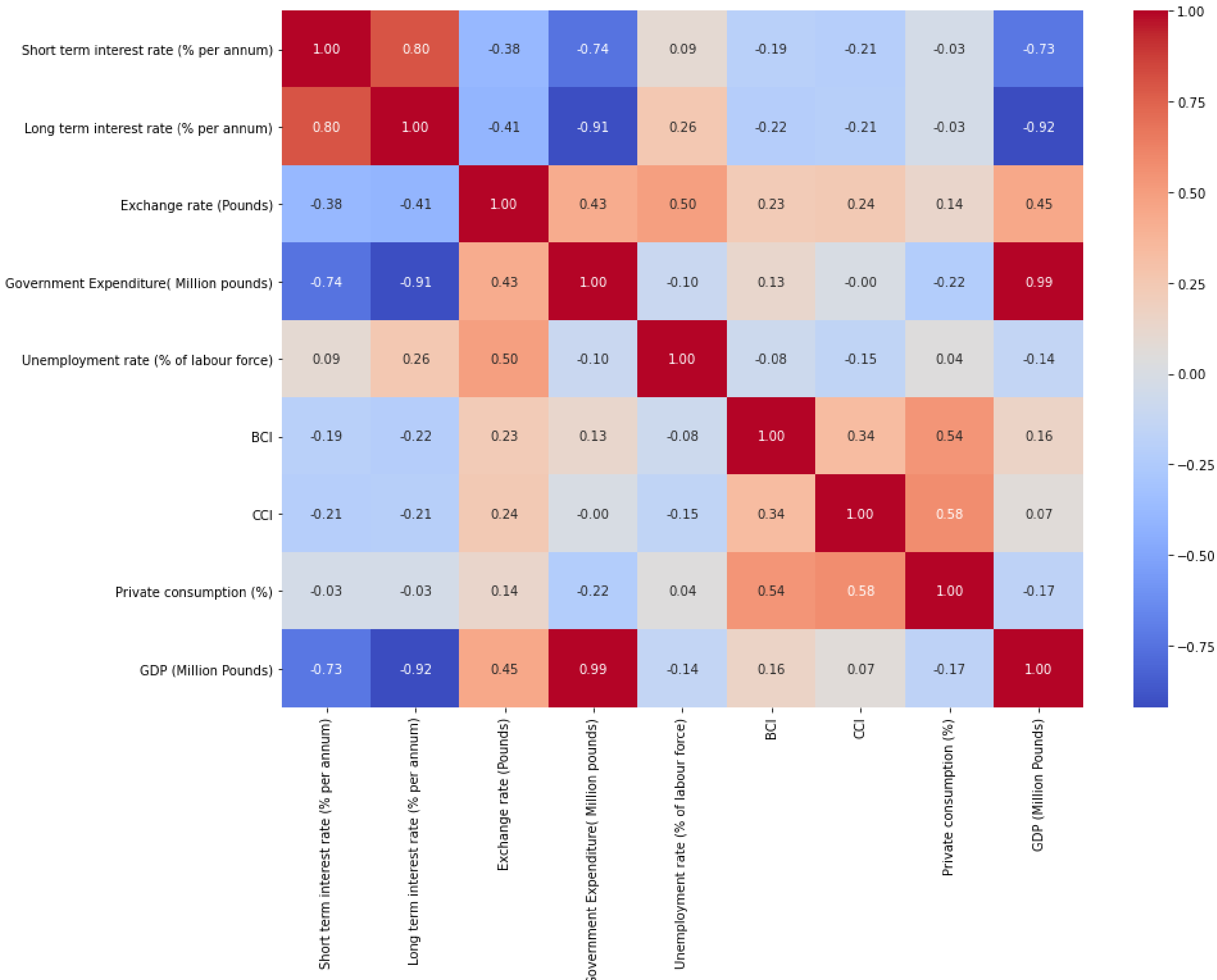
In [357…
```
data['BCI']=data['BCI'].fillna(data['BCI'].mean())
data['CCI']=data['CCI'].fillna(data['CCI'].mean())
```

In [358…
```
print(data.isnull().sum())
```

```
Year                                      0
Exports (Million Pounds)                  0
Short term interest rate (% per annum)    0
Long term interest rate (% per annum)     0
Exchange rate (Pounds)                    0
Government Expenditure( Million pounds)   0
Unemployment rate (% of labour force)     0
BCI                                       0
CCI                                       0
Private consumption (%)                   0
GDP (Million Pounds)                      0
dtype: int64
```

In [359…
```
##Correlation Matrix
```

In [360…
```
plt.figure(figsize=(14,10))
sns.heatmap(data=data.iloc[:,2:].corr(),annot=True,fmt='.2f',cmap='coolwarm')
plt.show()
```



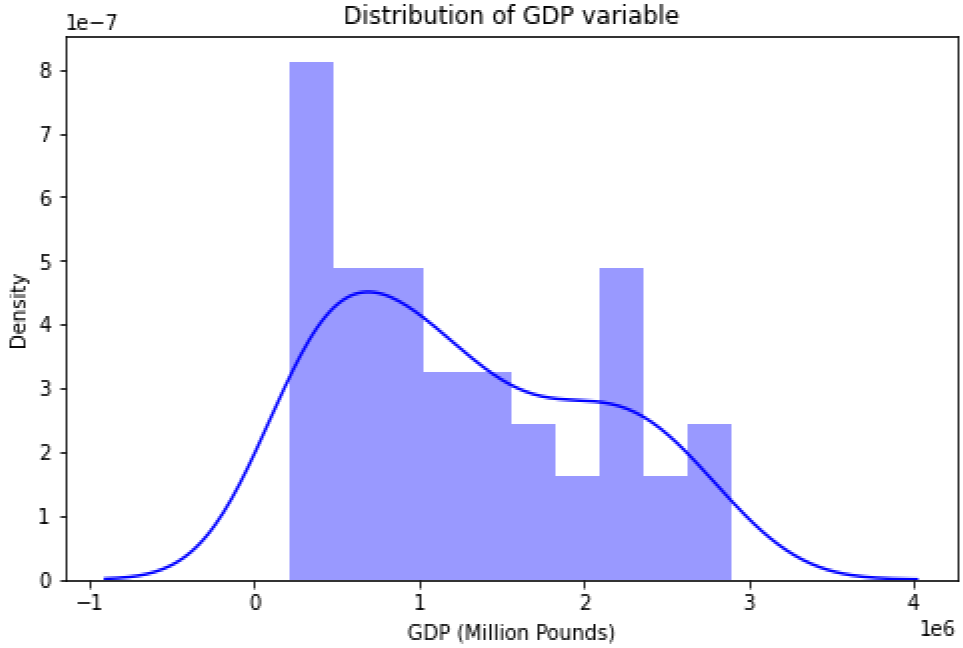
Loading [MathJax]/extensions/Safe.js

DISTRIBUTION PLOT

In [362…

```
hist=sns.distplot(data['GDP (Million Pounds)'], bins=10, color='blue')
hist.set_title("Distribution of GDP variable")
plt.show()
```

```
C:\Users\HP-PC\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [363…

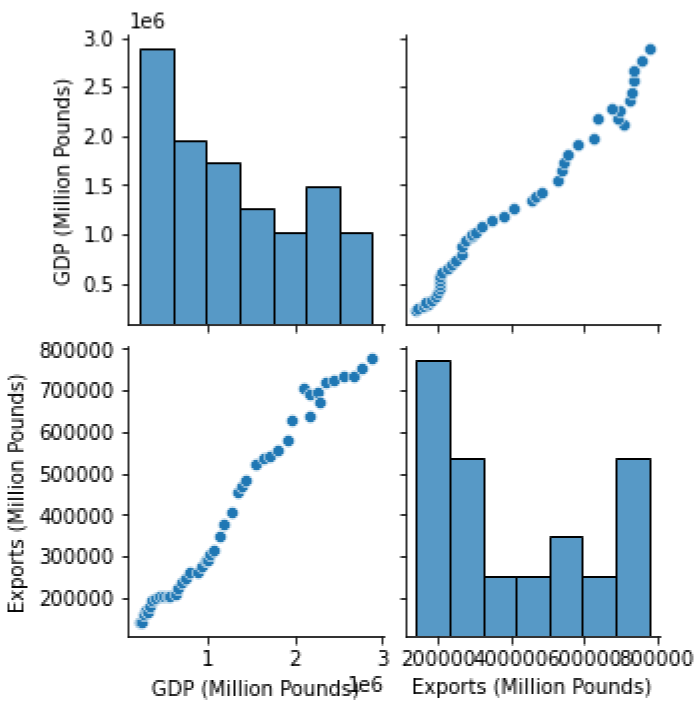
```
## LET US SEE A RELATIONSHIP BETWEEN THE GDP AND TO EXPORT if it a linear
```

In [364…

```
sns.pairplot(data,vars= ['GDP (Million Pounds)','Exports (Million Pounds)'])
```

Out[364…

```
<seaborn.axisgrid.PairGrid at 0x28fa432a6a0>
```



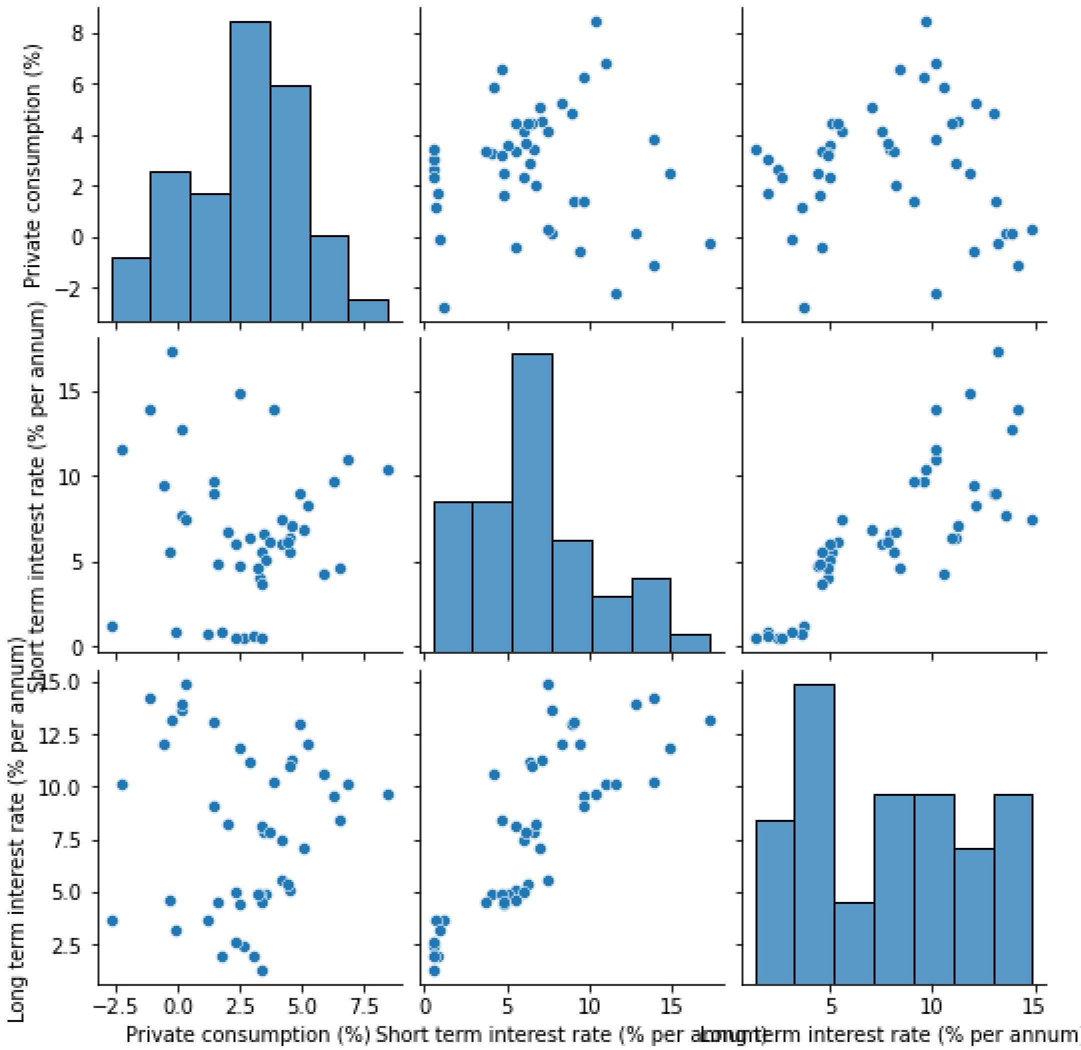
In [365…

```
sns.pairplot(data,vars= [
                       'Private consumption (%)',
                       'Short term interest rate (% per annum)',
```

```
                    'Long term interest rate (% per annum)'
                ])
```

Out[365…

```
<seaborn.axisgrid.PairGrid at 0x28fa4b892b0>
```



In [366…

```
## selection of Predictors and Target

X = data.drop(['GDP (Million Pounds)','Year'], axis=1)
Y = data['GDP (Million Pounds)']
```

In [367…

```
## Splitting the dataset using Train_Test_Split method
```

In [368…

```
train_X, test_X, train_Y, test_Y = train_test_split(X, Y,test_size=0.25,random_state=21)
```

In [369…

```
len(train_X),len( test_X), len(train_Y),len( test_Y)
```

Out[369…

```
(34, 12, 34, 12)
```

In [370…

```
test_X
```

Out[370…

	Exports (Million Pounds)	Short term interest rate (% per annum)	Long term interest rate (% per annum)	Exchange rate (Pounds)	Government Expenditure(Million pounds)	Unemployment rate (% of labour force)	BCI	CCI	Private consumption (%)
19	290297	14.808970	11.802500	0.563177	218137	6.825000	98.215630	96.929240	2.520864
39	673864	0.699774	3.624425	0.647179	670970	7.867382	100.620600	99.075150	1.153952

Loading [MathJax]/extensions/Safe.js

	Exports (Million Pounds)	Short term interest rate (% per annum)	Long term interest rate (% per annum)	Exchange rate (Pounds)	Government Expenditure(Million pounds)	Unemployment rate (% of labour force)	BCI	CCI	Private consumption (%)
28	482135	5.545774	5.093525	0.618057	330911	5.979465	98.741050	102.632000	4.480913
14	237749	6.438648	10.970000	0.779246	145364	11.175000	101.313000	99.166190	4.463877
17	263721	10.352680	9.675834	0.562170	168318	8.450000	104.453300	101.306800	8.481428
7	198157	8.299936	12.065000	0.521505	63630	6.100000	100.801800	104.719800	5.214587
26	456887	6.911723	7.052592	0.610836	312860	6.981590	101.097100	102.920000	5.076231
23	347112	5.564841	8.122100	0.653427	292082	9.512666	101.706300	98.177170	3.371168
13	224603	6.353592	11.127500	0.751807	137337	10.850000	101.381500	101.260900	2.885895
25	406863	6.110596	7.810184	0.640958	308867	8.113807	100.539000	101.403900	3.695914
11	206266	9.009463	13.085000	0.572447	117455	13.000000	96.912100	99.655860	1.415843
0	141149	6.632173	7.868333	0.410920	19294	3.500000	99.936437	100.012168	3.453198

RandomForest

In [371…
```
from sklearn.ensemble import RandomForestRegressor
```

In [372…
```
forest_model = RandomForestRegressor(random_state=21)
model = forest_model.fit(train_X,train_Y)
```

In [373…
```
pred = model.predict(test_X)
```

In [374…
```
print(pred)
```

```
[ 954699.69249 2318403.00309 1505521.54091  813492.30134  811054.41271
  402880.0899  1236312.36125 1089968.9383   632909.02914 1139457.47891
  533897.37605  301155.95828]
```

In [375…
```
from sklearn.metrics import r2_score
r2_score(test_Y, pred)
```

Out[375…
```
0.9820313390612223
```

In [376…
```
len(pred)
```

Out[376…
```
12
```

In [377…
```
len(test_X)
```

Out[377…
```
12
```

In [378…
```
model.predict([[776420, 0.498992, 1.305208, 0.740634, 716384, 4.892704, 101.217700, 101.58
```

Out[378…
```
array([2725288.59854])
```

In [379…
```
print(pred)
```

```
[ 954699.69249 2318403.00309 1505521.54091  813492.30134  811054.41271
  402880.0899  1236312.36125 1089968.9383   632909.02914 1139457.47891
  533897.37605  301155.95828]
```

In [380…
```
#Parameter Tuning

RMSE = np.sqrt(np.mean(-cross_val_score(forest_model, train_X, train_Y,cv=5,  scoring='neg
r2_score1= np.mean(cross_val_score(forest_model, train_X, train_Y,cv=5,  scoring='r2'))

print("Root Mean Squere Error(RMSE) : %f" % (RMSE))
print("Coefficient of Determination R2 score: %s" % '{:.2}'.format(r2_score1))
```

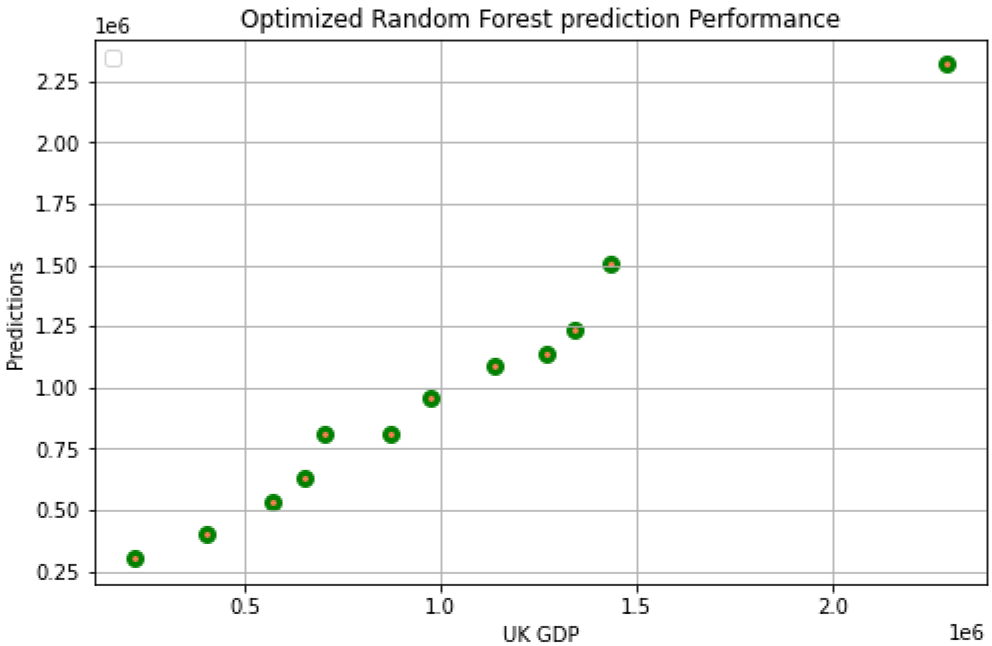
```
Root Mean Squere Error(RMSE) : 200878.271182
Coefficient of Determination R2 score: 0.82
```

In [381…
```
forest_model.score(train_X, train_Y)
```

Out[381… 0.9975255099176958

In [382…
```
fig = plt.figure(figsize=(8, 5))
plt.scatter(test_Y,pred, linewidths=3, edgecolors='g', color='coral')
plt.xlabel('UK GDP')
plt.ylabel('Predictions')
plt.title('Optimized Random Forest prediction Performance')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [384…
```
# Feature importance scores play an important role in a predictive modeling project, inclu
# insight into the model, and the basis for dimensionality reduction and feature selectio
# effectiveness of a predictive model on the problem.
# Machine Learning algorithms rank predictive variables and develop partial dependence plo
# of variables that may cause GDP growth or recessions
```

In [385…
```
importance = model.feature_importances_
```

In [386…
```
# Features:Exports (Million Pounds), Short term interest rate (% per annum)
#          Long term interest rate (% per annum), Exchange rate (Pounds),
#          Government Expenditure( Million pounds),Unemployment rate (% of labour force)
#          BCCI, CCI, Private consumption (%)
```

In [387…
```
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

Loading [MathJax]/extensions/Safe.js ore: 0.25654

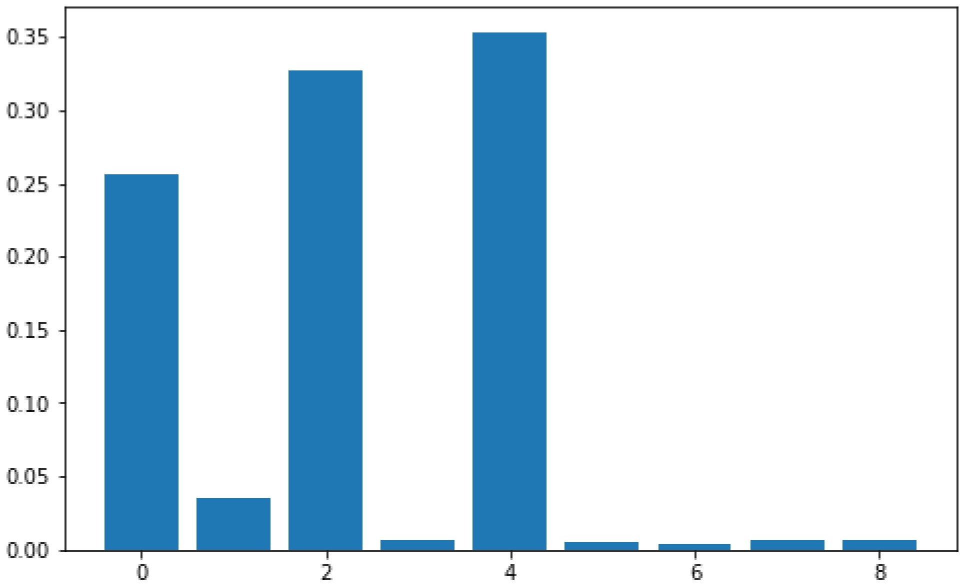
```
Feature: 1, Score: 0.03556
Feature: 2, Score: 0.32663
Feature: 3, Score: 0.00619
Feature: 4, Score: 0.35306
Feature: 5, Score: 0.00571
Feature: 6, Score: 0.00365
Feature: 7, Score: 0.00638
Feature: 8, Score: 0.00629
```

In [388…

```
from matplotlib import pyplot as plt
```

In [389…

```
plt.bar([x for x in range(len(importance))],importance)
plt.show()
```

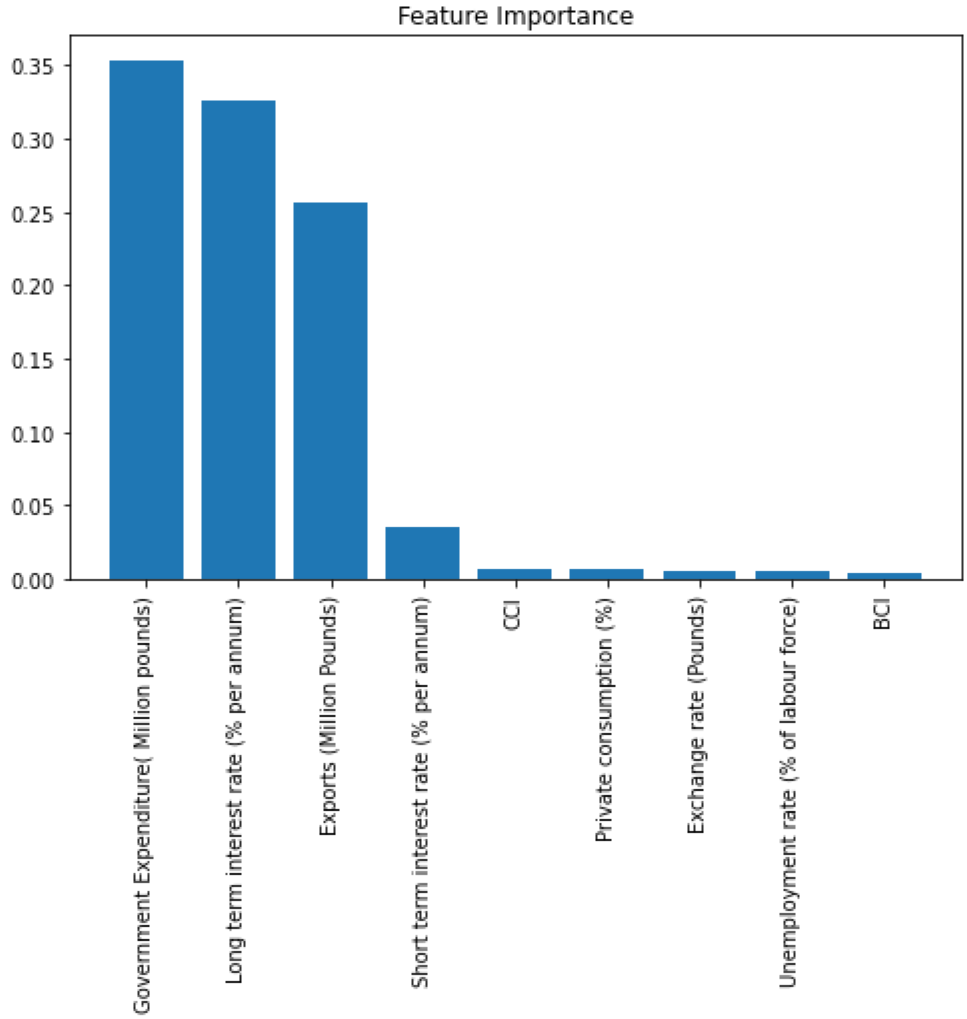


In [390…

```
# Let us order the predictors from stong predictors to the weak predictors
```

In [391…

```
sorted_indices = np.argsort(importance)[::-1]
plt.title('Feature Importance')
plt.bar(range(train_X.shape[1]), importance[sorted_indices], align='center')
plt.xticks(range(train_X.shape[1]), train_X.columns[sorted_indices], rotation=90)
plt.show()
```



In [392…

```
# Let us ivestigate the train_X set
```

In [393…

```
train_X.head()
```

Out[393…

	Exports (Million Pounds)	Short term interest rate (% per annum)	Long term interest rate (% per annum)	Exchange rate (Pounds)	Government Expenditure(Million pounds)	Unemployment rate (% of labour force)	BCI	CCI	Private consumption (%)
45	776420	0.498992	1.305208	0.740634	716384	4.892704	101.21770	101.58660	3.401299
38	636782	1.213653	3.647517	0.641919	645490	7.611054	95.43499	97.74997	-2.733007
40	722546	0.874580	3.135992	0.624141	668199	8.106182	101.17790	96.21677	-0.101321
43	735128	0.542949	2.569083	0.607730	706188	6.178483	102.77780	101.48150	2.338614
32	555129	3.735161	4.526592	0.612472	443447	5.007329	98.55768	101.16910	3.333789

In [394…

```
feature_scores = pd.Series(forest_model.feature_importances_, index = train_X.columns).sor
```

In [395…

```
print (feature_scores)
```

```
Government Expenditure( Million pounds)    0.353056
Long term interest rate (% per annum)      0.326627
Exports (Million Pounds)                   0.256539
Short term interest rate (% per annum)     0.035557
CCI                                        0.006384
Private consumption (%)                    0.006294
Exchange rate (Pounds)                     0.006190
Unemployment rate (% of labour force)      0.005708
```

```
BCI                                        0.003645
dtype: float64
```

Multi-Layer Perceptron Artificial Neural Network

In [396…
```
from sklearn.neural_network import MLPRegressor
```

In [397…
```
clf = MLPRegressor(solver='lbfgs',
                   alpha=1e-5,
                   hidden_layer_sizes=(6,),
                   random_state=1)
clf.fit(train_X,train_Y)
```

Out[397…
```
MLPRegressor(alpha=1e-05, hidden_layer_sizes=(6,), random_state=1,
             solver='lbfgs')
```

In [398…
```
# parameter tuning
```

In [399…
```
RMSE= np.sqrt(np.mean(-cross_val_score(clf, train_X, train_Y,cv=5,  scoring='neg_mean_squa
r2_score1= np.mean(cross_val_score(clf, train_X, train_Y,cv=5,  scoring='r2'))

print("Root Mean Squere Error(RMSE) : %f" % (RMSE))
print("Coefficient of Determination R2 score: %s" % '{:.2}'.format(r2_score1))
```

```
Root Mean Squere Error(RMSE) : 77790.283224
Coefficient of Determination R2 score: 0.97
```

In [400…
```
regr = MLPRegressor(random_state=1, max_iter=500).fit(train_X, train_Y)
regr.predict(test_X)
```

```
C:\Users\HP-PC\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the
optimization hasn't converged yet.
  warnings.warn(
```

Out[400…
```
array([ 902007.04745978, 2402742.13798378, 1436240.44185851,
        671540.23072443,  759575.91049984,  440152.46724614,
       1359609.22658168, 1139695.37153794,  634445.23906683,
       1270263.34502154,  564656.615212  ,  260389.34679817])
```

In [401…
```
regr.score(test_X,test_Y)
```

Out[401…
```
0.9896259567846826
```