The Netherlands Press

Journal of Emerging Strategies in New Economics

Article

GDP PREDICTION FOR COUNTRIES USING MACHINE LEARNING MODELS

[1]Gurunadh Velidi

Program Leader and Associate Professor,
School of Engineering,
University of Petroleum and Energy Studies
Energy Acres, Bidholi, via, Prem Nagar, Uttarakhand, India

Orchid Id: https://orcid.org/0000-0003-4024-247X

E-mail: gvssvelidi@ddn.upes.ac.in

Abstract.

The significance of GDP is demonstrated by the fact that it offers data on the size and functioning of an economy. Real GDP growth is frequently used as a measure of the state of the economy as a whole. Real GDP growth is typically regarded as a positive indicator of the state of the economy. The goal of this research is to forecast global GDP and determine its annual growth rate using machine learning. Since GDP varies from year to year, it is crucial to understand the status and importance of the variables that influence a country's GDP now. If you depict this in a diagram, it will be simpler to grasp. Additionally, the significance of the impacted parameters in determining GDP is calculated in this research-based study. Viewers may easily compare the GDP growth rates of other nations as well as their best and worst performance predictions. In this study, by using a variety of machine learning algorithms to analyse the global GDP.

Keywords: Machine Learning, Linear Regression, Parameters, GDP.

Received Date: 14 July 2022
Accepted Date: 12 August 2022
Published Date: 07 September 2022

1. Introduction

The total value of all completed products and services produced in a nation during a specific time period is known as the gross domestic product (GDP). Higher GDP nations often sustain higher living standards because they place greater value on domestically generated labour and commodities [1]. Although it may also be computed on a quarterly basis, gross domestic product is often determined on an annual basis. GDP, or gross domestic product or services, is often quite significant since it offers details on the character and functioning of an economy. The "World Gross Domestic Product Forecast" initiative is mostly focused on research [2]. To determine the best and worst prediction performance, four different learning regressions (linear regression, SVM, random forest, and gradient boosting) were examined in this study. Our study's main objective is to forecast GDP growth using Python and machine learning. [3]

In this study, by utilising Pandas for data analysis and manipulation, the NumPy library for array manipulation, and the Matplotlib library for point plotting [4]. Employed four different learning regressions gradient boosting, random forest, support vector machine, and linear regression.

In the distinctive columns that influence each country's GDP, the data come from all of the world's nations [5]. Using machine learning methods such as gradient boosting, random forest, and linear regression, our objective is to anticipate and estimate the nation's GDP per capita [6]. In order to predict how the economy will develop in the future, applied mathematics and mathematical models are used in GDP forecasting [7]. This enables us to track and forecast prior economic patterns but shifts in the state of the current economy may alter the pattern of earlier trends. Therefore, it will be simpler to create economic development targets the more accurate the government estimates are [8].

The most crucial lesson in comprehending future economic patterns is consequently the accurate forecasting of gross domestic product. Concerns about GDP are frequently cited as economic indicator factors [9]. An essential responsibility in examining a nation's economy and growth is GDP forecasting. In contrast to traditional econometric methodologies [10], the approach presented in this study shows how machine learning techniques may increase the accuracy of the computation of gross domestic product. Numerous social, economic, and cultural factors affect a nation's GDP [11]. Finally, the evaluated performance of our model using the three methods, and that discovered as gradient boosting, linear regression, and random forest had the greatest prediction performance [12]. The concept is also used to a web application that anticipates and calculates a nation's GDP by using that nation's qualities as inputs.

2. IMPLEMENTATION OF GDP PREDICTION USING MACHINE LEARNING:

By employing the approach, predictions-related laborious, time-consuming, and inaccurate chores are reduced. Economic calculations may be performed using the programme, simplifying structural alterations.

2.1. DATA CLEANING:

In order to complete this work, a data collection must gather for comprising all of the GDP factor values. The study also computed the significant variables and visually displayed those that had the greatest influence. In this case, first extract dataset from the CSV file that it was saved as. The element's value must next be examined to see if it includes a null value or another data type. The most recent machine learning techniques are then used to plot GDP predictions on a graph. All the columns might have an impact on GDP. By working on PySpark, a Python interface to Apache Spark.

It offers a PySpark shell for interactive data analysis in a distributed environment in addition to allowing you to create Spark apps using the Python API. Clustering important economic factors that affect the GDP of a country and based on these indicators predict GDP of that country.

```
from pyspark.sql.types import StructField,StructType,DateType,DoubleType,IntegerType,StringType
from pyspark.sql.functions import round

gdpschema = StructType([
    StructField("country", StringType()),
    StructField("year", IntegerType()),
    StructField("gdp", DoubleType()),
    StructField("gdpNote", IntegerType()),
    StructField("oilPrice", DoubleType()),
    StructField("gasPrice", DoubleType())
])

schema = StructType([
    StructField("country", StringType()),
    StructField("year", IntegerType()),
    StructField("workingPop", DoubleType())
])

gdp = spark.read.csv("gdpCleaned.csv",header=True,mode="DROPMALFORMED",schema=gdpschema)
work = spark.read.csv("workPop.csv", header=True, mode="DROPMALFORMED",schema=schema)
gdp = gdp.join(work,(gdp.year==work.year)&(gdp.country==work.country),"left").drop(work.year).drop(work.country)

gdp = gdp.select("country","year","gdp","gdpNote","oilPrice","gasPrice",round("workingPop",4).alias("workingPop")).orderBy("country","year")

gdp = gdp.coalesce(1)

gdp.write.csv("gdpCleanedD.csv", header=True)
```

A Spark Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator[country, cleaned name, country code, year, gdp, gdp note, debt, inflation, unemployment, lending interest, tech export, total reserves, fdi, crop production, oil price, gas price, working pop, exchange rate, oil prod, oil import price, ppp, researcherPer1k, ppi, total IndProd, share price, pollution, narrowMoneyM1, broadMoney, longTermInterest, shortTermInterest, importexport ratio, new business Diff, naturalGasImport, naturalGasExport, external Debt Stock, oilImport, oilExport,trade Import]. These stages are run in order, and the input DataFrame is transformed as it passes through each stage.

```
from pyspark.sql.types import StructField,StructType,DateType,DoubleType,IntegerType,StringType
from pyspark.sql.functions import year,round

gdpschema = StructType([
    StructField("country", StringType()),
    StructField("year", IntegerType()),
    StructField("gdp", DoubleType()),
    StructField("gdpNote", IntegerType()),
])

schema = StructType([
    StructField("date", DateType()),
    StructField("Price", DoubleType()),
])

gdp = spark.read.csv("gdpCleaned.csv",header=True,mode="DROPMALFORMED",schema=gdpschema)
oil = spark.read.csv("oil.csv", header=True, mode="DROPMALFORMED",schema=schema)
gas = spark.read.csv("natgas.csv",header=True,mode="DROPMALFORMED", schema=schema)

oil = oil.withColumnRenamed("Price","oilPrice")
gas = gas.withColumnRenamed("Price","gasPrice")

joinedData = oil.join(gas,oil.date==gas.date,'outer').drop(gas.date)

joinedData = joinedData.select(year(joinedData.date).alias("year"),joinedData.oilPrice,joinedData.gasPrice)

joinedData = (joinedData.groupBy("year").avg("oilPrice","gasPrice")
                        .withColumnRenamed("avg(oilPrice)","oilPrice")
                        .withColumnRenamed("avg(gasPrice)","gasPrice"))

oilngasData = (joinedData.select(
                                "year",
                                round("oilPrice",3).alias("oilPrice"),
                                round("gasPrice",3).alias("gasPrice")).orderBy("year"))

gdp = gdp.join(oilngasData,gdp.year==oilngasData.year,'left').drop(oilngasData.year).orderBy("country","year")

gdp = gdp.coalesce(1)

gdp.write.csv("gdpCleanedD.csv", header=True)
```

2.2. BUILDING PIPELINE IN THE SCHEMA:

An ML pipeline is created using Pyspark. Spark machine learning refers to this MLlib DataFrame-based API.A machine learning (ML) pipeline is a complete workflow combining multiple machine learning algorithms together. There can be many steps required to process and learn from data, requiring a sequence of algorithms.

```
from pyspark.sql.types import StructField,StructType,DateType,DoubleType,IntegerType,StringType

schema = StructType([
    StructField("country", StringType()),
    StructField("year", IntegerType()),
    StructField("gdp", DoubleType()),
    StructField("gdpNote", IntegerType()),
    StructField("debt", DoubleType()),
    StructField("inflation", DoubleType()),
    StructField("unemployment", DoubleType()),
    StructField("lendingInterest", DoubleType()),
    StructField("techExport", DoubleType()),
    StructField("totalReserves", DoubleType()),
    StructField("fdi", DoubleType()),
    StructField("cropProduction", DoubleType()),
    StructField("oilPrice", DoubleType()),
    StructField("gasPrice", DoubleType()),
    StructField("workingPop", DoubleType()),
])

countrySch = StructType([
    StructField("country", StringType()),
    StructField("cleanedName", StringType()),
    StructField("countryCode", StringType())
])

gdp = spark.read.csv("final.csv",header=True,mode="DROPMALFORMED",schema=schema)

countryCode = spark.read.csv("countryCode.csv",header=True,mode="DROPMALFORMED",schema=countrySch)

gdp = gdp.join(countryCode,gdp["country"]==countryCode["country"],"inner").drop(countryCode["country"])

gdp = gdp.select("country","cleanedName","countryCode","year","gdp","gdpNote","debt","inflation","unemployment",\
                 "lendingInterest","techExport","totalReserves","fdi","cropProduction","oilPrice","gasPrice",\
                "workingPop")

gdp.write.csv("gdpCleanedD.csv", header=True)
```

2.3. SAVING THE FINAL CLEANED DATA INTO CSV:

By giving the data types a SQL table is generated and passed through pipeline. By using the SQL table, the prediction is done. For cleaning the data, null values should be checked in the data and if there are any special characters is there or not and the special characters are removed.

```
schema = StructType([
    StructField("country", StringType()),
    StructField("cleanedName", StringType()),
    StructField("countryCode", StringType()),
    StructField("year", IntegerType()),
    StructField("gdp", DoubleType()),
    StructField("gdpNote", IntegerType()),
    StructField("debt", DoubleType()),
    StructField("inflation", DoubleType()),
    StructField("unemployment", DoubleType()),
    StructField("lendingInterest", DoubleType()),
    StructField("techExport", DoubleType()),
    StructField("totalReserves", DoubleType()),
    StructField("fdi", DoubleType()),
    StructField("cropProduction", DoubleType()),
    StructField("oilPrice", DoubleType()),
    StructField("gasPrice", DoubleType()),
    StructField("workingPop", DoubleType()),
    StructField("exchangeRate", DoubleType()),
    StructField("oilProd", DoubleType()),
    StructField("oilImportPrice", DoubleType()),
    StructField("ppp",DoubleType()),
    StructField("researcherPer1k",DoubleType()),
    StructField("ppi", DoubleType()),
    StructField("totalIndProd", DoubleType()),
    StructField("sharePrice",DoubleType()),
    StructField("pollution", DoubleType()),
    StructField("narrowMoneyM1",DoubleType()),
    StructField("broadMoney",DoubleType()),
    StructField("longTermInterest",DoubleType()),
    StructField("shortTermInterest",DoubleType()),
    StructField("importExportRatio",DoubleType()),
    StructField("newBusinessDiff",DoubleType()),
    StructField("naturalGasImport",DoubleType()),
    StructField("naturalGasExport",DoubleType()),
    StructField("externalDebtStock",DoubleType()),
    StructField("oilImport",DoubleType()),
    StructField("oilExport",DoubleType()),
    StructField("tradeImport",DoubleType())
])

gdp = spark.read.csv("gdpCleaned.csv",mode="DROPMALFORMED",header=True,schema=schema)
```

```
ppi = spark.read.csv("ppi.csv",header=True,mode="DROPMALFORMED",schema=ppi)

ppi = ppi.filter(ppi.Type=="AGRWTH").drop(ppi.Type)
ppi = ppi.groupBy("country","year").avg("ppi").alias("ppi")

gdp = gdp.join(ppi, (gdp.countryCode==ppi.country) & (gdp.year==ppi.year), "left").drop(ppi.year).drop(ppi.country)

gdp.write.csv("gdpCleanedD.csv", header=True)
```

3. FEATURE ENGINEERING & TRAINING DATA:

When working on any feature engineering task, the data should be examined to determine whether there is any variance or if any non-negative columns have negative values. These are all examined, after the completion of cleaning task. Additionally, for later usage, the cleaned data is stored as a separate CSV file. By using order by, sorting of all

columns by year and nation. With the same structure, all of the files have now been re-registered and added to the pipeline. The data has now been divided into features and labels, and this split data needs to be used with PySpark to train and test models. Additionally, training and test data are separated from the data. Binomial logistic regression may be used to predict binary outcomes by putting training and test data via a transform of vectors before transferring the data to a linear regression model in spark.ml increase. The multinomial logistic regression findings are applied. A transformer called a "Vector Assembler" creates a single vector column from a supplied list of columns. Combining raw data and features produced by several feature converters into a single feature vector is helpful for training ML models like logistic regression and decision trees.

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import Normalizer
from pyspark.ml.linalg import Vectors
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit

text = sc.textFile('filled.csv')
text = text.map(lambda text:text.split(";"))


countries = ['Canada','India','United States','China','France']
header = text.first()
features = text.filter(lambda line:line!=header)

for country in countries:
    data = features.filter(lambda text:text[-1]==country)
    data = data.filter(lambda text:text[-3]>1984)
    test = data.filter(lambda text:text[-3] in ['2014','2015'])
    train = data.filter(lambda text:text[-3] not in ['2014','2015'])
    train = train.map(lambda text:(float(text[-2]),Vectors.dense(text[0:3]))).toDF(['label','features'])
    test = test.map(lambda text:(float(text[-2]),Vectors.dense(text[0:3]))).toDF(['label','features'])
    lr = LinearRegression(maxIter=3)
    pipeline = Pipeline(stages=[lr])
    paramGrid = (ParamGridBuilder().addGrid(lr.regParam, [1, 0.75 ,0.5, 0.1, 0.01])
                                   .addGrid(lr.elasticNetParam, [0.0,0.5, 1.0])
                                   .build())
    crossval = CrossValidator(estimator=pipeline,
                              estimatorParamMaps=paramGrid,
                              evaluator=RegressionEvaluator(),
                              numFolds=3)
    cvModel = crossval.fit(train)
    prediction = cvModel.transform(test)
    prediction.show()
```

By passing the training and testing data after the conversion of vectors using Vector assembler and the data is then passed into the Linear Regression model in spark.ml logistic regression can be used to predict a binary outcome is obtained by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression. VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees. The only way is to pass the data into the model by fitting the model is trained for the data and can be used for testing. For the model evaluation performance test cross validation is used for the model performance test.

4. RESULTS & DISCUSSIONS:

The findings are shown below in a bar chart after testing and graphing the actual and predicted values. For Canada, the difference between the actual number and the expected value was around 1.5. For India actual and predicted are same. for united states there is a huge difference in actual and predicted one.

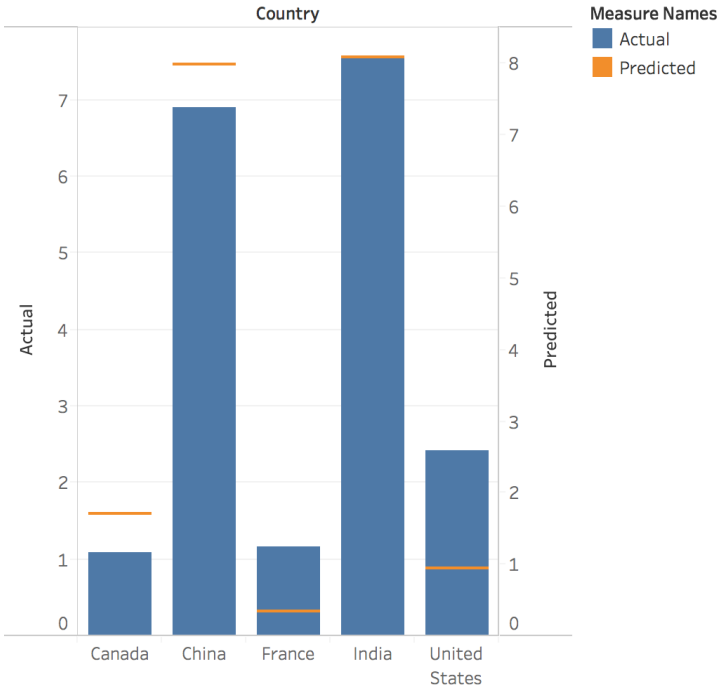

Figure 1: Plot for Actual VS Predicted

The data suggests that the most accurate metric for determining GDP is the number of records obtained. According to the bar chart below, the metric with the most effectiveness was ppp, an the lowest effectiveness was external target stock. From this test the most dependent parameter can be found.

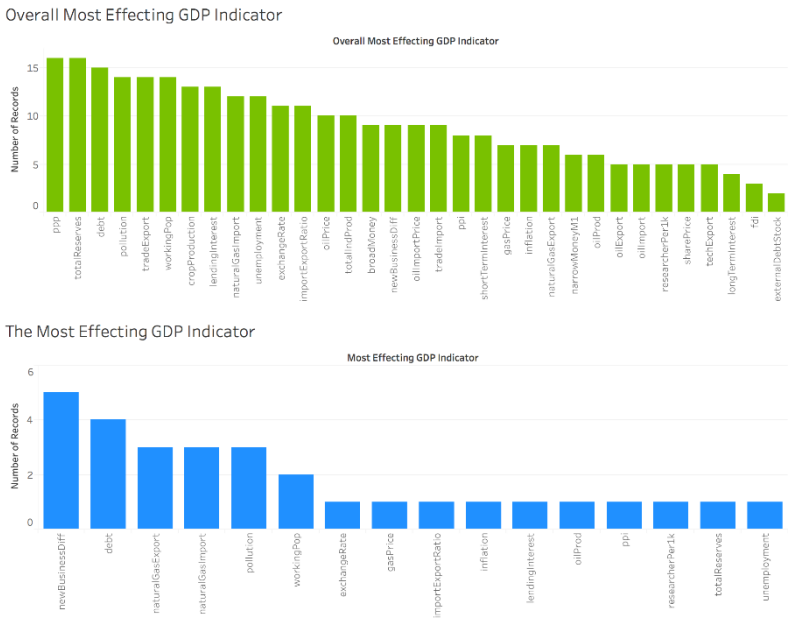

Figure 2: Most Affecting Feature For GDP

4.1. Matrix of Data Correlation:

The correlation coefficients of several variables are displayed in a straightforward table called a correlation matrix. All potential value pairings in a table are represented as relationships in a matrix. It is an effective tool for distilling big data sets and locating and displaying certain data trends. For forecasting the development of connections between

variables, correlation matrices are helpful. A correlation matrix provides a broad overview of the more or less significant correlations between various variables.

For the GDP Analysis ,30 countries from the list are taken for evaluation using confusion matrix, by creating a heatmap of the Pearson correlation between GDP metrics and GDP growth and filter the top 10 metrics for each nation with the greatest correlation values. From this matrix f1-score, recall, precision is obtained.

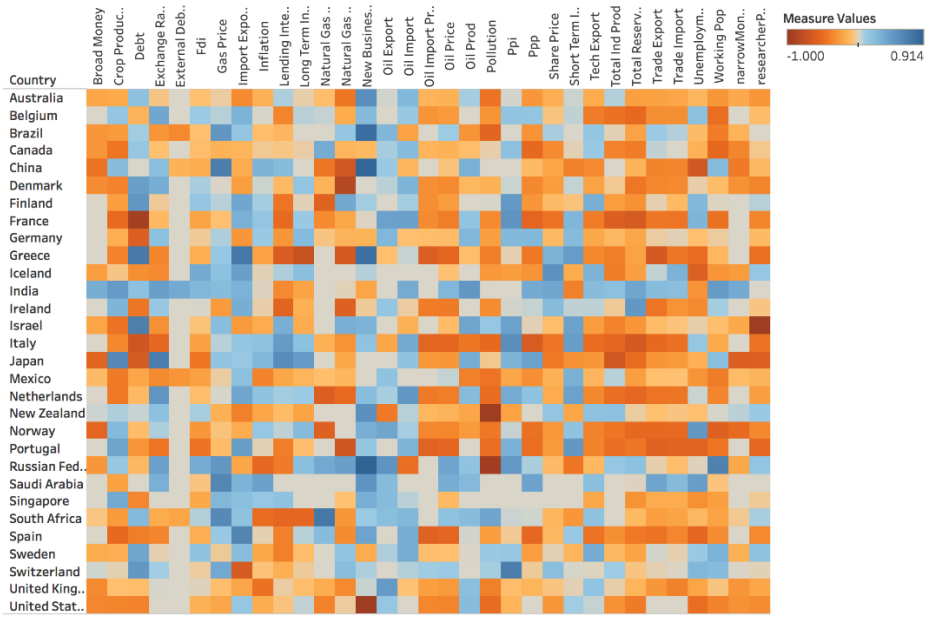

Figure 4: Heat Map

From this actual GDP and the predicted one for linear regression and decision trees, difference between the values obtained are small for both models and hence these models can be used to predict the GDP of a country as it is important because it gives information about the size of the economy and how an economy is performing.

Table: Actual and Predicted Values for Both Models

COUNTRY	PREDICTED	ACTUAL
Canada	[2.74, 1.06]	[2.47, 1.08]
United States	[1.22, 2.48]	[2.43, 2.43]
Australia	[3.03, 2.88]	[2.5, 2.26]
Netherlands	[0.22, 1.19]	[1.01, 1.99]
South Africa	[2.67, 1.26]	1.55, 1.28]
Italy	[-0.99, 0.89]	[-0.34, 0.76]
India	[6.47, 7.68]	[7.24, 7.57]
New Zealand	[3.27, 3.79]	[3.17, 3.39]
Switzerland	[1.92, 0.52]	[1.89, 0.91]
Israel	[2.52, 2.85]	[2.6, 2.49]
China	[8.05, 5.32]	[7.27, 6.9]
Singapore	[4.99, 2.74]	[3.26, 2.01]
Saudi Arabia	[4.75, 4.98]	[3.64, 3.49]

5. CONCLUSION:

In order to forecast the GDP of all nations, this study examined two machine learning algorithms: a multiple linear regression method and a random forest regressor (decision tree). The analysis' findings demonstrate that the random forest regression method is inferior to the multiple linear regression approach. In emerging nations like India, decision-making in the public and commercial sectors is heavily influenced by the analysis of economic metrics like GDP. The study used data from the World Bank and machine learning algorithms to forecast GDP (current US dollars), one of his primary economic metrics.

References

[1] Fernando J. Gross Domestic Product (GDP). Investopedia. 2021.
[2] Laine M. Introduction to dynamic linear models for time series analysis. In Geodetic Time Series Analysis in Earth Sciences 2020 (pp. 139-156). Springer, Cham.
[3] Lalitha VL, Raju SH, Sonti VK, Mohan VM. Customized Smart Object Detection: Statistics of detected objects using IoT. In2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) 2021 Mar 25 (pp. 1397-1405). IEEE.
[4] Qin T. Machine Learning Basics. In Dual Learning 2020 (pp. 11-23). Springer, Singapore.
[5] Brownlee J. A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. 2016.
[6] Kurihara Y, Fukushima A. AR Model or Machine Learning for Forecasting GDP and Consumer Price for G7 Countries. Applied Economics and Finance. 2019;6(3):1-6.
[7] Richardson A, van FL Orenstein Mulder T, Vehbi T. Nowcasting GDP using machine-learning algorithms: A real-time assessment. International Journal of Forecasting. 2021 Apr 1;37(2):941-8.
[8] Cicceri G, Inserra G, Limosani M. A machine learning approach to forecast economic recessions—an Italian case study. Mathematics. 2020 Feb;8(2):241.
[9] Richardson A, Mulder T. Nowcasting New Zealand GDP using machine learning algorithms.
[10] Daga UR, Das R, Maheshwari B. Estimation, Analysis and Projection of India's GDP.
[11] Chen DH, Dahlman CJ. The knowledge economy, the KAM methodology and World Bank operations. World Bank Institute Working Paper. 2005 Oct 19(37256).
[12] Hawksworth J, Cookson G. The world in 2050. How big will the major emerging market economies get and how can the OECD compete. 2006 Sep.