

India's GDP Prediction

AUTHORS:

SATYAJIT GHANA, SHIKHAR SINGH

Table of Contents

Project Aim.....	2
Project Data	2
About the Data:.....	2
Graphical Visualization.....	4
Model Structure	7
Model 1 (MLP Regressor):.....	7
Model 2 (Time Series):	8
Outcomes and Results:	8
Project Members:	10

India's State GDP Prediction

Project Aim

The main aim of this project was to create a ML model that would help in predicting the expected GDP of each state of India with minimum margin of error for any given year in the future such as 2020 and beyond. This is very beneficial since by this data we could predict the growth of different states of the country and it will be able to predict how the GDP must be used in an effective way for the benefit of the state.

Project Data

The Data used in the project was taken from <https://niti.gov.in/>. The RAW Data was taken from the financial section that contained the following documents: Aggregate Expenditure, Aggregate Receipts, Capital Expenditure, Capital Receipts, Fiscal Deficits, Interest Payments, Outstanding Liabilities, Own Tax Revenues, Revenue Deficits, Revenue Receipts, Social Sector Expenditure and Nominal GSDP Series. The RAW data was in XLSX format which was converted to a well-known CSV format that could be later easily read by python pandas library. The unknown values that were originally represented by a '-' was replaced by NA's by pandas.

The Data was given for 29 states and 2 Union Territories, from the year 1981 to 2016, some of the data was missing for some of the states, due to the fact that the data was started to be collected at a later period of time, or maybe the state wasn't even formed in that year, such years were removed from the data by using remove NA's in python pandas, and the clean and preprocessed data was dumped to a CSV file. The preprocessing done to create the final CSV file that contained the list of features and the corresponding GSDP was created by fetching the features of every state and every year and matching them to their corresponding GSDP of that state in that year. By using this and removing those series which had NA's a total of 660 observations were made in the final CSV file.

About the Data:

```
[3]: niti = pd.read_csv('NETDATA.csv')
[4]: niti.head()
```

	Capital Receipts	Aggregate Receipts	Social Sector Expenditure	Interest Payments	Own Tax Revenues	Fiscal Deficits	Outstanding Liabilities	Aggregate Expenditure	Revenue Receipts	Revenue Expenditure	Revenue Deficits	Capital Expenditure	Nominal_GSDP_Series
0	325.0	1590.0	532.0	81.7	582.0	222.1	81.50	1610.0	1265.0	1161.0	-104.0	449.0	8191.0
1	187.0	709.0	171.0	30.0	66.0	-27.4	43.41	758.0	522.0	357.0	-165.0	401.0	2516.0
2	702.0	1690.0	452.0	107.1	277.0	335.8	106.33	1791.0	988.0	929.0	-59.0	862.0	7353.0
3	345.0	1370.0	406.0	68.6	531.0	246.5	80.76	1442.0	1025.0	903.0	-122.0	539.0	7427.0
4	116.0	576.0	137.0	37.0	234.0	112.0	30.76	607.0	460.0	401.0	-59.0	207.0	3386.0

```
[6]: niti.describe().T
```

[6]:		count	mean	std	min	25%	50%	75%	max
	Capital_Receipts	660.0	2428.893939	4110.573309	-5116.00	238.7500	811.50	2583.750	37639.0
	Aggregate_Receipts	660.0	7791.672727	10883.932609	38.00	1143.5000	3191.50	9882.000	72074.0
	Social_Sector_Expenditure	660.0	2623.565152	3470.775608	9.00	381.2500	1114.00	3512.000	24268.0
	Interest_Payments	660.0	1060.228030	1757.027349	0.00	81.7000	290.00	1240.000	11870.0
	Own_Tax_Revenues	660.0	2604.877273	4224.076558	0.00	90.0000	879.50	3174.750	33540.0
	Fiscal_Deficits	660.0	1630.949848	2688.436374	-360.00	159.9500	517.00	1736.150	18620.0
	Outstanding_Liabilities	660.0	417.522303	596.079415	1.42	49.6725	177.21	505.725	4067.4
	Aggregate_Expenditure	660.0	7663.422727	10658.752422	43.00	1175.5000	3078.50	9737.250	72668.0
	Revenue_Receipts	660.0	5362.778788	7044.667651	37.00	892.5000	2347.50	7210.750	48438.0
	Revenue_Expenditure	660.0	6022.030303	8317.578141	30.00	864.0000	2275.50	7755.000	52280.0
	Revenue_Deficits	660.0	659.255455	1824.575250	-4328.00	-62.3750	41.25	594.675	18583.0
	Capital_Expenditure	660.0	1641.445455	2554.712652	13.00	289.5000	797.00	1851.750	21622.0
	Nominal_GSDP_Series	660.0	37596.312121	57944.592770	52.00	2963.0000	14311.00	44635.000	486766.0

Figure 0-1 Data Description

Graphical Visualization

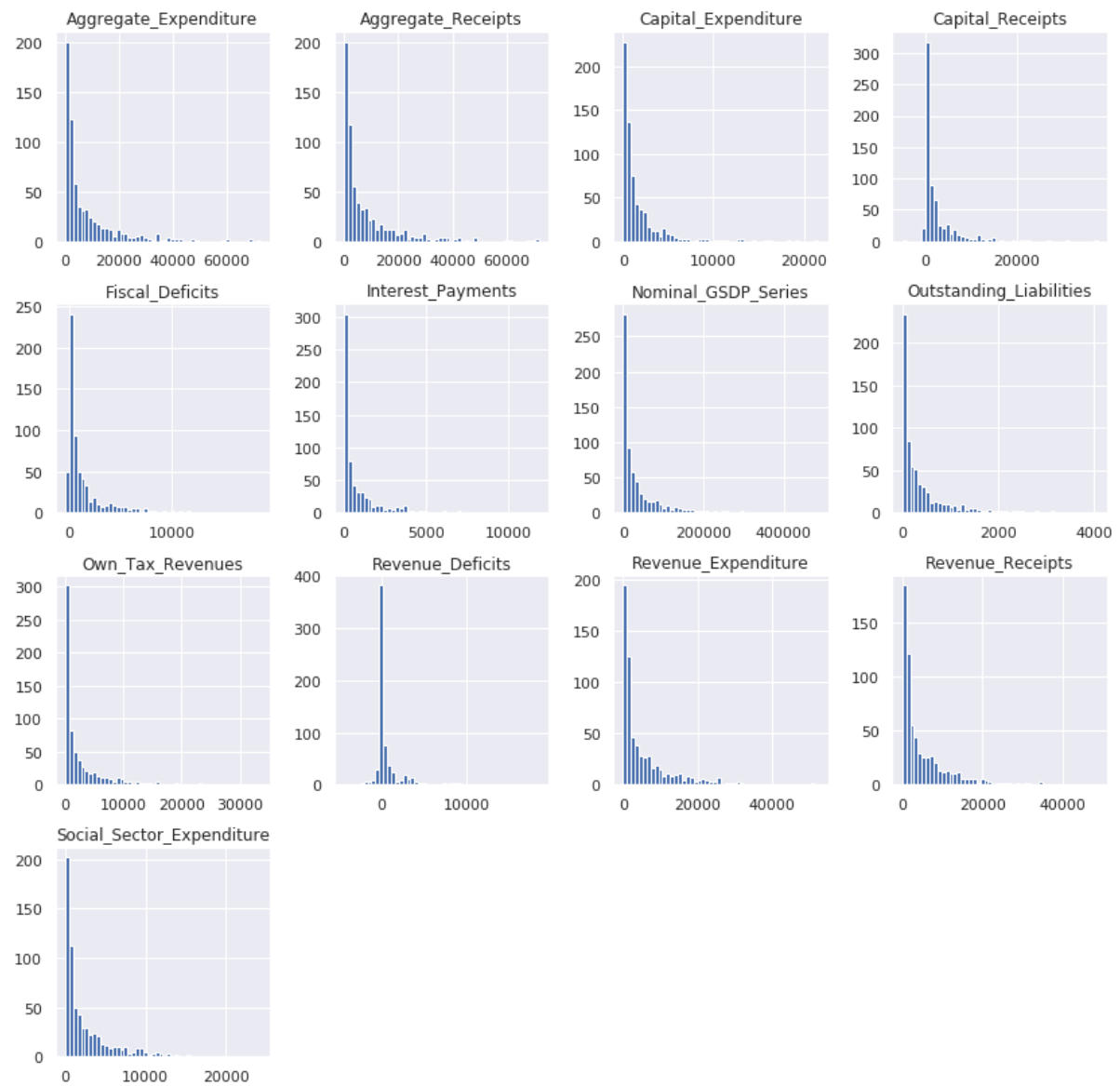


Figure 0-1 Histogram Plot



Figure 0-2 Pair Plot

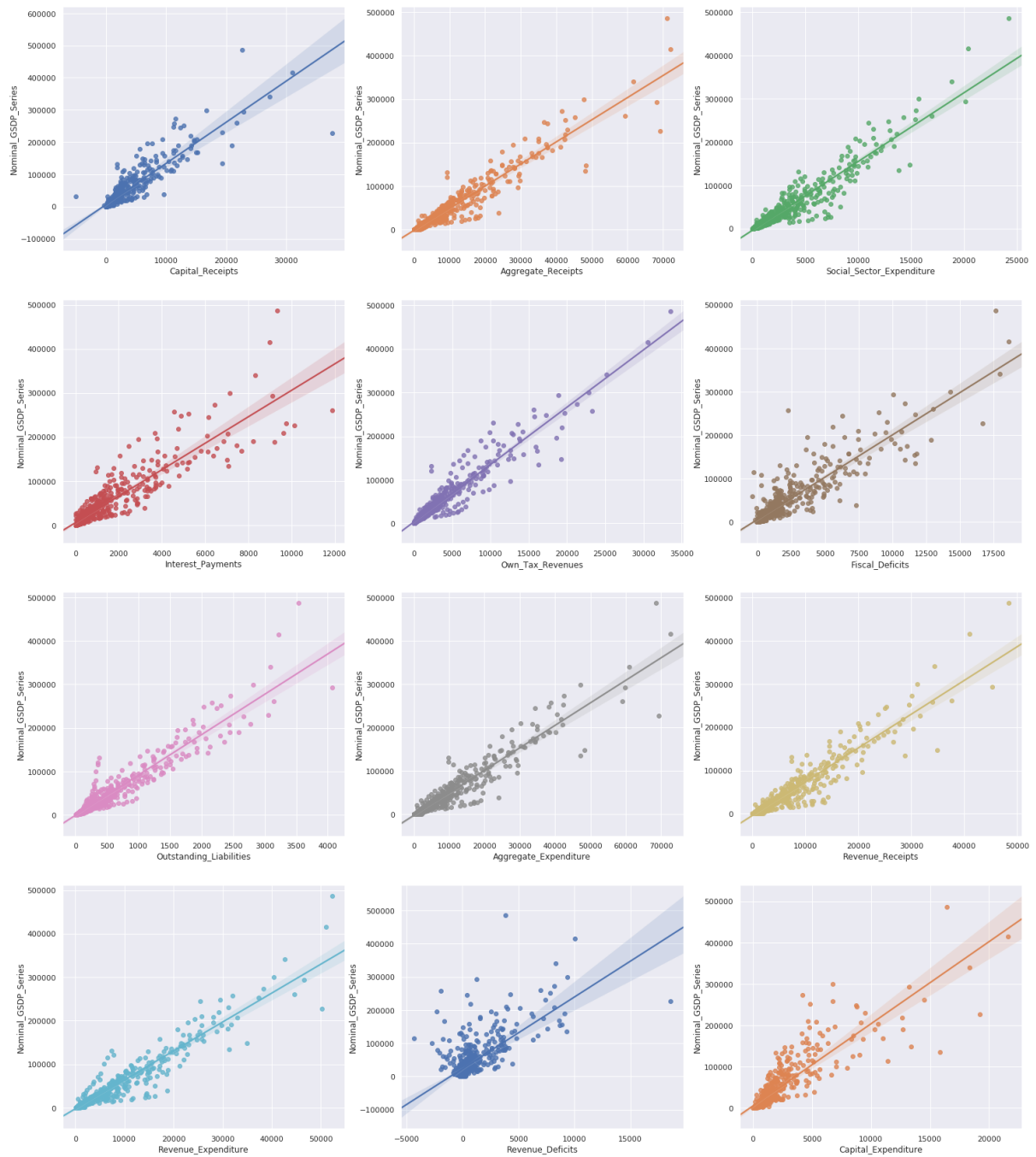


Figure 0-3 Regression Plot

Model Structure

The Model is split into two parts, one model predicts the Nominal GSDP from the features mentioned in the Project Data such as the Aggregate Expenditure, Aggregate Receipts, etc. and the other model is a time series Model that predicts these features for any given year, now the idea is to use the already given data from NITI Aayog and train the model to predict the GSDP series, then feed the predicted data for the year 2020 to get the Nominal GSDP values for the year 2020 and beyond.

Model 1 (MLP Regressor):

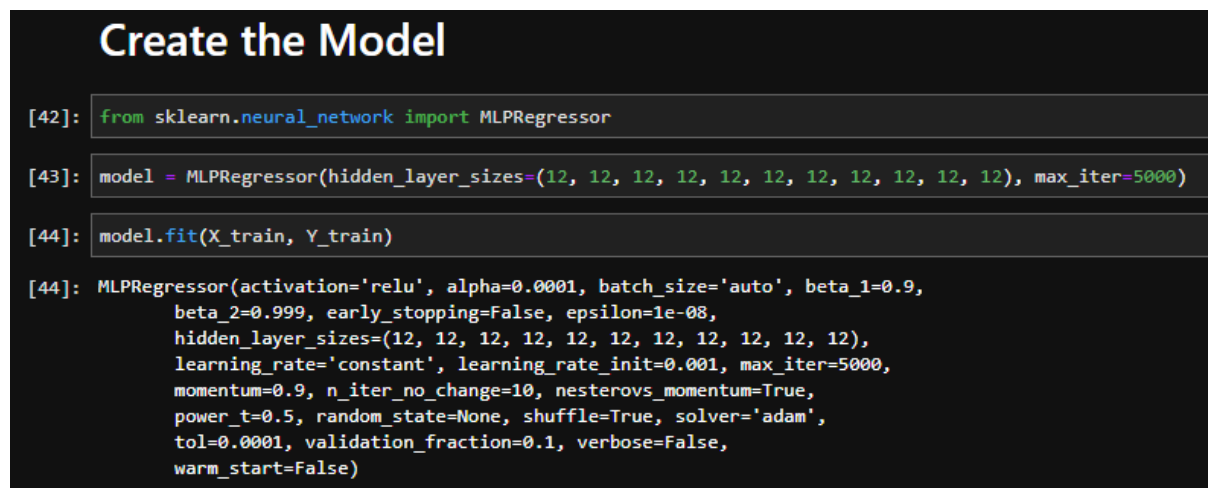
The first model is a Multi-Layer Perceptron Regressor, a type of neural network, which is used here to predict the Nominal GSDP for the given data. The features were scaled before being used in the regressor model.

Features: Aggregate Expenditure, Aggregate Receipts, Capital Expenditure, Capital Receipts, Fiscal Deficits, Interest Payments, Outstanding Liabilities, Own Tax Revenues, Revenue Deficits, Revenue Receipts, Social Sector Expenditure

Output: Nominal GSDP Series

Learning Rate: 0.001

Hidden Layers Sizes: 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12

A screenshot of a Jupyter Notebook with a dark background. The title 'Create the Model' is at the top in a large, bold, white font. Below the title, there are four code cells. The first cell imports MLPRegressor from sklearn.neural_network. The second cell creates a model instance with 11 hidden layers of size 12 and max_iter=5000. The third cell calls model.fit(X_train, Y_train). The fourth cell shows the full definition of the MLPRegressor model with various parameters like 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=(12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12), learning_rate='constant', learning_rate_init=0.001, max_iter=5000, 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, and warm_start=False.

```
[42]: from sklearn.neural_network import MLPRegressor

[43]: model = MLPRegressor(hidden_layer_sizes=(12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12), max_iter=5000)

[44]: model.fit(X_train, Y_train)

[44]: 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=(12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12),
    learning_rate='constant', learning_rate_init=0.001, max_iter=5000,
    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)
```

Figure 0-1 MLP Model

The Model was trained for the training data and the following score was obtained on the test data.


```
[45]: model.score(X_test, Y_test)
[45]: 0.9531020120990584
```

Figure 0-2 Model Score

The Model was analyzed using the standard sklearn metrics

```
[49]: from sklearn.metrics import mean_squared_error, r2_score

[50]: Ya = Y_test
      Yp = model.predict(X_test)
      mse = mean_squared_error(Ya, Yp)
      R2 = r2_score(Ya, Yp)

[52]: 'MSE : {}, RMSE : {}, R2-Score : {}'.format(mse, np.sqrt(mse), R2)
[52]: 'MSE : 160203075.99019524, RMSE : 12657.1353785205, R2-Score : 0.9531020120990584'
```

Figure 0-3 Model Metrics

The R2 score here determines the accuracy of the model, and 95.31% is considered to be a good score.

Model 2 (Time Series):

The second model was to be a time series model that predicts the features for a given year in the future, for this there are options such as ARIMA, which is considered to be a really good model, but since the data points are very few for us, Holt Winters model was used. Holt-Winters forecasting is a way to model and predict the behavior of a sequence of values over time—a time series. Holt-Winters is one of the most popular forecasting techniques for time series.

For each of the states, the features for the year 2020 was predicted, and this was stored in a CSV file.

Outcomes and Results:

The data predicted for the year 2020 from the Model 2 was fed into Model 1 to predict the Nominal GSDP for the individual states and also all the states, to get the following results:

[37]:

	State	GDP_PREDICTED_2020
0	Andhra Pradesh	481240.99330622
1	Arunachal Pradesh	50504.68981682
2	Assam	390211.21918752
3	Bihar	726323.19485558
4	Chhattisgarh	388877.65141521
5	Goa	81462.42403318
6	Gujarat	857814.79435828
7	Haryana	463170.04960620
8	Himachal Pradesh	128457.71566522
9	Jammu & Kashmir	244718.33326239
10	Jharkhand	306534.47590013
11	Karnataka	967747.28644563
12	Kerala	682853.80957752
13	Madhya Pradesh	733880.37806087
14	Maharashtra	1510044.32444540
15	Manipur	32939.69089979
16	Meghalaya	42287.99038998
17	Mizoram	32071.93903682
18	Nagaland	47094.68768639
19	Odisha	461303.82268048
20	Punjab	406445.12533406
21	Rajasthan	833866.56491167
22	Sikkim	51806.25943954
23	Tamil Nadu	1262122.34296933
24	Telangana	1072571.17836985
25	Tripura	111742.44278025
26	Uttar Pradesh	1720322.72289364
27	Uttarakhand	182038.57056880
28	West Bengal	818546.66112111
29	All States	22742357.98458954
30	Delhi	438377.61694767
31	Puducherry	28916.06750310

Figure 0-1 Model Results

Project Members:

Satyajit Ghana
Shikhar Singh