GDP /Capita Prediction Using IBMWatson Machine Learning

1. INTRODUCTION

1.1 Overview

Gross Domestic Product (GDP) is the final value of all the economic goods and services produced within the country's geographic boundaries during a specified period of time. GDP growth rate is the major indicator of a country's economic performance. Broadly speaking, the primary sector (agricultural), the secondary sector (industry), and the tertiary sector (services) all contribute to GDP by producing goods and services (services). Gross Domestic Product (GDP) is a key tool that guides investors, policymakers, and businesses in strategic decision-making. Per capita GDP is a global indicator of a country's economy that economists use in combination with GDP to assess a country's wealth based on its economic growth.

The project aims at building a web app that predicts the Gross Domestic Product (GDP) value by taking the input values. In this project we created a model from a data set that includes region, population, area, population density, coastline, net migration, phone, arable, crops, climate, birthrate, death rate, agriculture, industry, service. Then we created IBM Watson Studio Service, IBM Watson Machine Learning and IBM Cloud Object Storage Service on IBM Cloud.

IBM Watson Studio Service is used for training the model. IBM Watson Machine Learning service is used for deploying the model and IBM Cloud Object Storage Service for storing.

1.2 Purpose

GDP not only helps in diagnosing the problems related to the economy, but also helps in correcting it. If GDP is rising, the economy is in solid shape, and the nation is moving forward. On the other hand, if gross domestic product is falling, the economy might be in trouble, and the nation is losing ground.

The primary goal of this project is to investigate the dataset "Countries of the World" and to focus on the elements that are influencing a Country's GDP per capita. By using this application we can easily predict the value of GDP/capita. So that we can analyze whether the country is moving forward or is it in a trouble.

2. LITERATURE SURVEY

2.1 Existing problem

- Existing system is highly manual and involves a lot of paper work and calculation. Therefore may be erroneous. This led to inconsistency and inaccuracy.
- Data may be lost, stolen or destroyed because it is stored on paper.
- Consume lot of time causing inconvenience to user
- Failure to represent degree of all inputs.
- Failure to account for cost imposed in human health

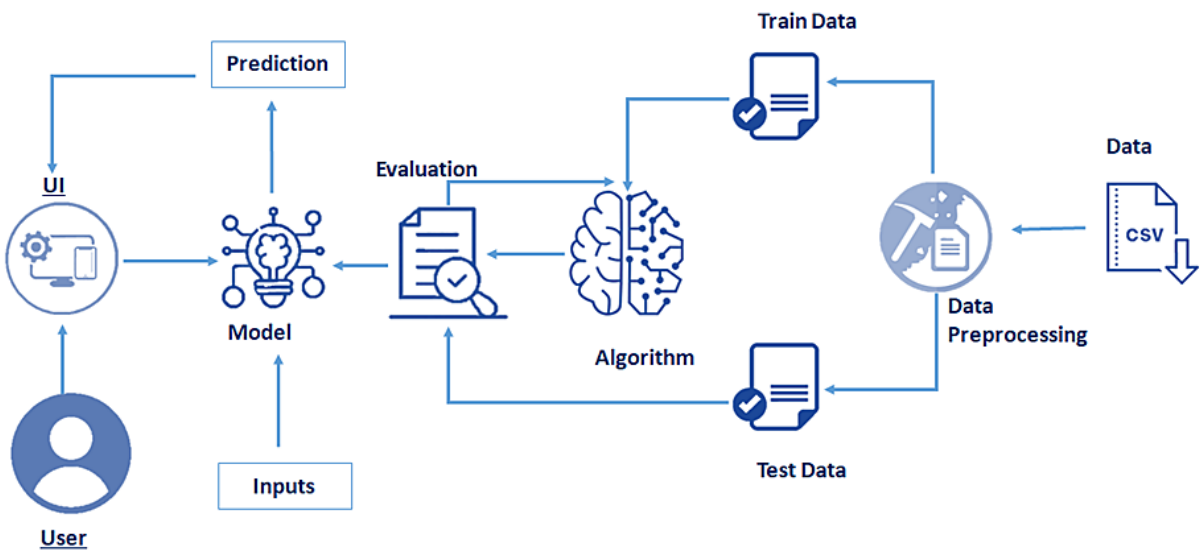
2.2 Proposed solution

- User friendly
- Easy to access

- Input data are valid
- Reduce lot of paper work
- Machine performs all calculation. Hence chance of error are nearer to nil.

3. THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

- IBM Watson Studio - IBM Watson Studio helps data scientists andanalysts prepare data and build models at scale across any cloud.
- IBM Watson Machine Learning - IBM Watson Machine Learninghelps data scientists and developers in deploying the model

- IBM Cloud Object Storage - IBM Cloud Object Storage makes itpossible to store practically limitless amounts of data, simply and costeffectively.

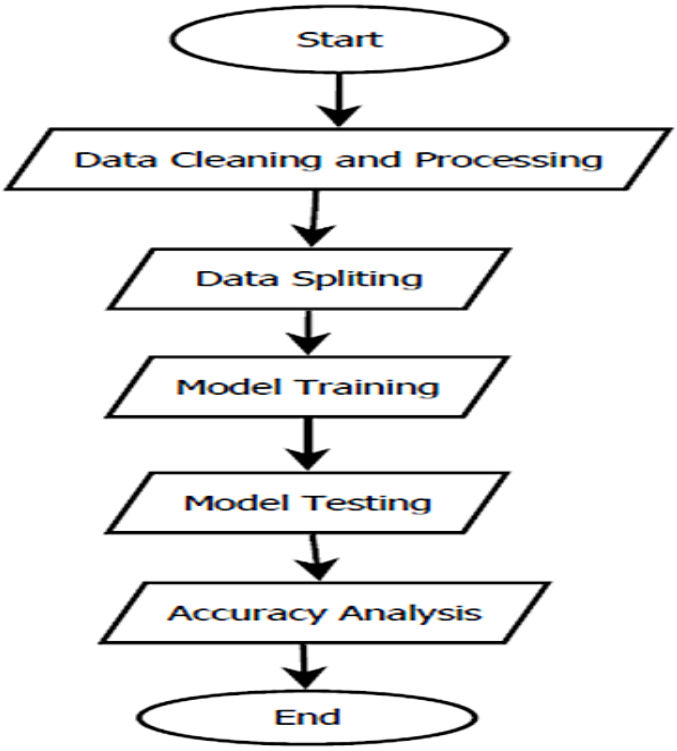
4. EXPERIMENTAL INVESTIGATIONS

A machine learning model is built that predicts GDP/capita value based on following parameters:

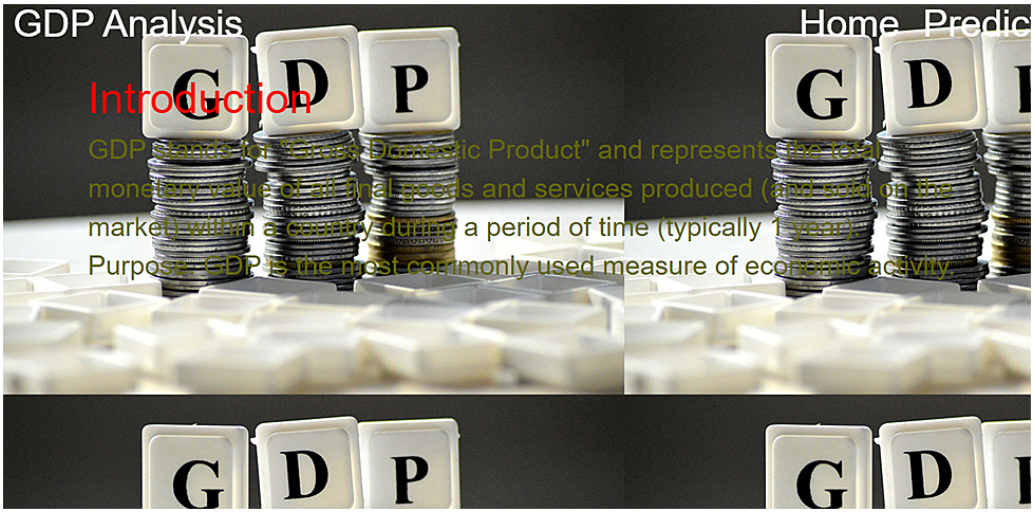
- Region
- Population
- Area
- Population density
- Coastline
- net migration
- Phone
- Arable
- Crops
- Climate
- Birthrate
- death rate
- Agriculture
- Industry
- Service

These 15 parameters affect the value of GDP. And these parameters help in predicting the GDP value

5. FLOWCHART



6. RESULT



This is the home page of the application

This is the second page of the application. This page will be redirected while clicking on predict in home page. In this user can input values in the field provided.

This is the third page of the application. Here predicted value GDP will be displayed

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

Gross Domestic Product (GDP) is an economic measure of a nation's total income and

output for a given time period (usually a year). Economists use GDP to measure the relative wealth and prosperity of different nations, as well as to measure the overall growth or decline of a nation's economy.

It helps to make cross-country comparisons of average living standards and economic wellbeing.

GDP helps in diagnosing the problems related to the economy, and also helps in correcting it.

GDP is the sum of the following elements:

- Total domestic consumption
- Total domestic investment expenditures
- Government expenditures
- Net exports

DISADVANTAGES:

It has some important limitations that includes:

- GDP doesn't count unpaid volunteer work
- The failure to represent the degree of income inequality in society
- The failure to indicate whether the nation’s rate of growth is sustainable or not
- GDP doesn't account for quality of goods
- Loss of privacy

8. APPLICATIONS

- GDP is an important measurement for economists and investors because it is a representation of economic production and growth. Both economic production and growth have a large impact on nearly everyone within a given economy. When the economy is healthy, there is usually a lower level of unemployment, and wages tend to increase as businesses hire more labor to meet the growing demand of the economy.

- Economists look at positive GDP growth between different time periods (usually year-to-year) to make an assessment of how much an economy is flourishing. Conversely, if there is negative GDP growth, it may be an indicator that an economy is in or approaching a recession or an economic downturn.
- Investors pay attention to the GDP because a significant percentage change in the GDP–either up or down–can has a significant impact on the stock market. In general, a bad economy usually means lower earnings for companies. And this can translate into lower stock prices.
- Investors may pay attention to positive and negative GDP growth when they are devising an investment strategy. However, it's important to note that because GDP is a measurement of the economy in the previous quarter or year, it is better used to help explain how economic growth and production have impacted your stocks and your investments in the past. It is not considered a helpful predictor of how the market will move in the future.

9. CONCLUSION

As technology is used in every aspect of our lives, the country's economy is no exception. Data science deals with massive amounts of data using modern tools and techniques and enables better decision making, predictive analysis, and pattern discovery. Using data science in GDP analysis enables us to know the factors that are affecting the GDP per capita of various countries. This helps to focus on the areas that help to foster economic development.

10. FUTURE SCOPE

As we know that day-to-day all the requirements are changing and getting more flexible so In the future we can add more parameters if needed to predict and analyzethe GDP value.

11. BIBILOGRAPHY

- https://www.investopedia.com/terms/g/gdp.asp
- https://www.cliffsnotes.com/cliffsnotes/subjects/economics/what-are-the-advantages-and-disadvantages-of-gross-domestic-product
- https://towardsdatascience.com/a-data-science-workflow-26c3f05a010e
- https://www.investopedia.com/terms/p/per-capita-gdp.asp
- https://thecleverprogrammer.com/2020/05/26/gdp-analysis-with-data-science/
- https://www.guru99.com/data-science-tutorial.html

12. APPENDIX

Jupyter code

```
_
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.preprocessing import LabelEncoder, StandardScaler
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
import scipy.stats as stat
import pylab
import pickle
```

```
df = pd.read_csv('world.csv',decimal=',')
df.head()

df.columns



df['Region'].replace(['ASIA (EX. NEAR EAST)         ','EASTERN EUROPE
',
                      'NORTHERN AFRICA                ','OCEANIA                    ',
                         'WESTERN EUROPE              ','SUB-SAHARAN AFRICA
',
                         'LATIN AMER. & CARIB    ','C.W. OF IND. STATES ','NEAR
EAST                      ',
                            'NORTHERN AMERICA                    ','BALTICS
'],
                       ['ASIA (EX. NEAR EAST)','EASTERN EUROPE','NORTHERN
AFRICA','OCEANIA','WESTERN EUROPE',
                        'SUB-SAHARAN AFRICA', 'LATIN AMER. & CARIB','C.W. OF
IND. STATES','NEAR EAST',
                       'NORTHERN AMERICA','BALTICS'],inplace=True)

plt.figure(figsize=(10,6))
sns.countplot(df['Region'])
plt.xticks(rotation=90)
plt.show()

df_n = df.select_dtypes(include=['int','float'])
features = list(df_n.columns)

"""Out of 18 features 15 features have outliers"""

for i,j in enumerate(features):
plt.figure(figsize=(20,50))
plt.subplot(9,2,i+1)
sns.boxplot(df_n[j])
plt.show()
```

```
# Countries belongs to respective regions

df['Country'].groupby(by=df['Region']).sum()

"""Literacy is defined as being able to read and write, or to having knowledge about a specific subject"""

literacy = ['0-20%' if x<=20.0 else '20-40%' if x>20.0 and x<=40.0 else '40-60%' if x>40.0 and x<=60.0 else '60-80%'
        if x>60.0 and x<=80.0 else '80-100%' for x in df['Literacy (%)']]

phone = ['0-200' if x<=200.0 else '200-400' if x<=400.0 and x>200.0 else '400-600' if x>400.0 and x<=600.0
        else '600-800' if x>600.0 and x<=800.0 else '800-1000' for x in df['Phones (per 1000)']]



"""Arable farming is growing crops in fields, which have usually been ploughed before planting"""

arable = ['0-20%' if x<=20.0 else '20-40%' if x>20.0 and x<=40.0 else '40-60%' if x>40.0 and x<=60.0 else '60-80%'
        if x>60.0 and x<=80.0 else '80-100%' for x in df['Arable (%)']]

crops = ['0-20%' if x<=20.0 else '20-40%' if x>20.0 and x<=40.0 else '40-60%' if x>40.0 and x<=60.0 else '60-80%'
        if x>60.0 and x<=80.0 else '80-100%' for x in df['Crops (%)']]

others = ['0-20%' if x<=20.0 else '20-40%' if x>20.0 and x<=40.0 else '40-60%' if x>40.0 and x<=60.0 else '60-80%'
        if x>60.0 and x<=80.0 else '80-100%' for x in df['Other (%)']]

gdp = ['0-10000$' if x<=10000.0 else '10000-20000$' if x>10000.0 and x<=20000.0
        else '20000-30000$' if x>20000.0 and x<=30000.0 else '30000-40000$' if
```

```
x>30000.0 and x<=40000.0
        else '40000-50000$' if x>40000.0 and x<=50000.0 else 'more than 50000$'
for x in df['GDP ($ per capita)']]

population = ['Below 1 million' if x<=1000000 else '1-20 million' if x>1000000
and x<=20000000 else '20-60 million'
            if x>20000000 and x<=60000000 else '60-100 million' if x>60000000
and x<=100000000 else '100-500 million'
             if x>100000000 and x<=500000000 else 'above 500 million' for x in
df['Population']]

plt.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=literacy)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=phone)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=arable)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()


.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=crops)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(18,8))
```

```
sns.countplot(df['Region'],hue=others)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=gdp)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(18,8))
sns.countplot(df['Region'],hue=population)
plt.xticks(rotation=30)
plt.legend(loc='upper right')
plt.show()

plt.figure(figsize=(15,15))
sns.heatmap(df.corr().abs(),annot=True)

df.describe(include='all')

df.shape

df.info()

"""From the descriptive statistics, we found country column completely have unique values. So, drop country column.
From the heat map we found ['Other (%)' & 'Arable (%)'] & ['Infant mortality (per 1000 births)' & 'Birthrate'] are highly
correlated. So, 'Other (%)' & 'Infant mortality (per 1000 births)' columns are dropped."""

df.drop(['Other (%)','Infant mortality (per 1000 births)','Country'],axis=1,inplace=True)

np.where(df['Phones (per 1000)']>1000)
```

```
df.isnull().sum(axis=1).sort_values(ascending=False)[0:10]

df.drop([223,221,134,78,138],axis=0,inplace=True)
df.tail()

df.isnull().sum()

df['Climate']=df['Climate'].fillna(df['Climate'].mode().max())

nullFeatures=list(df.columns)

for i,j in enumerate(nullFeatures):
    if df[j].isnull().sum()!=0:
df[j]=df[j].fillna(df[j].mean())

df.tail()

df.columns

plt.figure(figsize=(15,20))
plt.subplot(4,3,1)
sns.boxplot(np.log(df['Population']))
plt.subplot(4,3,2)
sns.boxplot(np.log(df['Area (sq. mi.)']))
plt.subplot(4,3,3)
sns.boxplot(np.log1p(df['Pop. Density (per sq. mi.)']))
plt.subplot(4,3,4)
sns.boxplot(np.log1p(df['Coastline (coast/area ratio)']))
plt.subplot(4,3,5)
sns.boxplot(np.sqrt(df['Net migration']))
plt.subplot(4,3,6)
sns.boxplot(np.log(df['GDP ($ per capita)']))
plt.subplot(4,3,7)
sns.boxplot(np.sqrt(df['Phones (per 1000)']))
plt.subplot(4,3,8)
sns.boxplot(np.sqrt(df['Arable (%)']))
```

```
plt.subplot(4,3,9)
sns.boxplot(np.log1p(df['Crops (%)']))
plt.subplot(4,3,10)
sns.boxplot(np.log(df['Deathrate']))
plt.subplot(4,3,11)
sns.boxplot(np.sqrt(df['Agriculture']))
plt.subplot(4,3,12)
sns.boxplot(np.sqrt(df['Industry']))
plt.show()
```

```
# Log transformation and square root transform on outliers

df['Population'] = np.log(df['Population'])
df['Area (sq. mi.)'] = np.log(df['Area (sq. mi.)'])
df['Pop. Density (per sq. mi.)'] = np.log1p(df['Pop. Density (per sq. mi.)'])
df['Coastline (coast/area ratio)'] = np.log1p(df['Coastline (coast/area ratio)'])
#df['Net migration'] = np.sqrt(df['Net migration'])
#df['GDP ($ per capita)'] = np.log(df['GDP ($ per capita)'])
df['Phones (per 1000)'] = np.sqrt(df['Phones (per 1000)'])
df['Arable (%)'] = np.sqrt(df['Arable (%)'])
df['Crops (%)'] = np.log1p(df['Crops (%)'])
df['Deathrate'] = np.log(df['Deathrate'])
df['Agriculture'] = np.sqrt(df['Agriculture'])
df['Industry'] = np.sqrt(df['Industry'])

# 'Literacy (%)' feature contains outliers even after transformation. So, removing this feature

df.drop(['Literacy (%)','Net migration'],axis=1,inplace=True)

# handling categorical feature

le =LabelEncoder()
df['Region']=le.fit_transform(df['Region'])
```

```
# Splitting data

x = df.drop('GDP ($ per capita)',axis=1)
y = df['GDP ($ per capita)']

x.columns

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3,random_state=10)

st=StandardScaler()
xtrain_scaled=st.fit_transform(xtrain)
xtest_scaled=st.transform(xtest)

def linear_reg(xtrain_scaled,xtest_scaled,ytrain,ytest):
lr=LinearRegression()
lr.fit(xtrain_scaled,ytrain)
ypred=lr.predict(xtest_scaled)
   score=r2_score(ytest,ypred)
rmse=np.sqrt(mean_squared_error(ytest,ypred))
print('***Linear Regression model***')
print('Score for Linear Regression model is {}'.format(score))
print('RMSE for Linear Regression model is {}'.format(rmse))
def random_forest_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest):
   rf=RandomForestRegressor()
rf.fit(xtrain_scaled,ytrain)
ypred=(rf.predict(xtest_scaled))
   score=r2_score(ytest,ypred)
rmse=np.sqrt(mean_squared_error(ytest,ypred))
print('***Random Forest Regressor Model***')
print('Score for Random Forest Regressor Model is {}'.format(score))
print('RMSE for Random Forest Regressor Model is {}'.format(rmse))

def svr(xtrain_scaled,xtest_scaled,ytrain,ytest):
svr=SVR()
svr.fit(xtrain_scaled,ytrain)
ypred=(svr.predict(xtest_scaled))
```

```
    score=r2_score(ytest,ypred)
rmse=np.sqrt(mean_squared_error(ytest,ypred))
    print('***SVR***')
print('Score for SVR Model is {}'.format(score))
print('RMSE for SVR Model is {}'.format(rmse))

def model_compare(xtrain_scaled,xtest_scaled,ytrain,ytest):
linear_reg(xtrain_scaled,xtest_scaled,ytrain,ytest)
print('-'*100)
random_forest_regressor(xtrain_scaled,xtest_scaled,ytrain,ytest)
print('-'*100)

model_compare(xtrain_scaled,xtest_scaled,ytrain,ytest)

rf=RandomForestRegressor()
rf.fit(xtrain_scaled,ytrain)
ypred=(rf.predict(xtest_scaled))

pickle.dump(rf,open('model.pkl','wb'))

x_sc = st.transform(x)

cv = cross_val_score(rf,x_sc,y,cv=5)
np.mean(cv)

ytrain.loc[4]

xtrain.loc[4]

rf.predict([xtrain.loc[4]])
```

app_IBM.py

```
from flask import Flask, render_template, request # Flask is a application
import requests
```

```
# used to run/serve our application
# request is used to access the file which is uploaded by the user in out application
# render_template is used for rendering the html pages
import pickle # pickle is used for serializing and de-serializing Python object structures
from sklearn.preprocessing import StandardScaler

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "_27PB6GRNxGmGWPkglVZ9S4KVe510MMdkYbQuA3HUGLI"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]


header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

app = Flask(__name__) # our flask app


@app.route('/')  # rendering the html template
def home():
    return render_template('home.html')


@app.route('/predict')  # rendering the html template
def index():
    return render_template("predict.html")


@app.route('/pred', methods=['GET', 'POST'])  # route for our prediction
def predict():
    Region = request.form['Region']
    Population = request.form['Population']
    Area = request.form['Area']
    Pop = request.form['Pop']
```

```
    Coastline = request.form['Coastline']
    Phones = request.form['Phones']
    Arable = request.form['Arable']
    Crops = request.form['Crops']
    Climate = request.form['Climate']
    Birthrate = request.form['Birthrate']
    Deathrate = request.form['Deathrate']
    Agriculture = request.form['Agriculture']
    Industry = request.form['Industry']
    Service = request.form['Service']
    total = [
        [Region, Population, Area, Pop, Coastline, Phones, Arable, Crops, Climate,
Birthrate, Deathrate, Agriculture,
        Industry, Service]]
    print(total)
payload_scoring = {"input_data": [{"field": [["Region", "Population", "Area",
"Pop", "Coastline", "Phones", "Arable", "Crops", "Climate", "Birthrate",
"Deathrate", "Agriculture", "Industry", 'Service']], "values": total}]}
    # print(total)
response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/72def3c4-6a1a-4e57-9920-
6e10bcf32fd8/predictions?version=2022-03-25', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
    print(response_scoring.json())
    # predictions = response_scoring.json()
    pred = response_scoring.json()
    print(pred)
    prediction = pred['predictions'][0]['values'][0][0]

    return render_template('gdp_pred.html', prediction=prediction)


if __name__ == '__main__':
app.run(debug=False)
```