```
import pandas as pd
import numpy as np
xls = pd.read_excel('Investment_dataset\
IMFInvestmentandCapitalStockDataset2021
(2).xlsx',sheet_name=['Definitions','Dataset'])
df = xls['Dataset']
df = df.dropna()
# df.head()
```

```
<>:3: SyntaxWarning: invalid escape sequence '\I'
<>:3: SyntaxWarning: invalid escape sequence '\I'
C:\Users\ASUS\AppData\Local\Temp\ipykernel_26316\218476968.py:3:
SyntaxWarning: invalid escape sequence '\I'
  xls = pd.read_excel('Investment_dataset\
IMFInvestmentandCapitalStockDataset2021
(2).xlsx',sheet_name=['Definitions','Dataset'])
```

```
df = df[df['country'] == 'India']
df.head()
```

```
     isocode  ifscode country  year   igov_rppp    kgov_rppp
ipriv_rppp  \
4585     IND      534   India  1985  104.837128  1150.348022
63.565014
4586     IND      534   India  1986  123.229591  1219.615479
61.220966
4587     IND      534   India  1987  107.421791  1304.720703
97.403656
4588     IND      534   India  1988  110.030991  1371.472412
101.886238
4589     IND      534   India  1989  107.043198  1438.534790
123.371620

       kpriv_rppp  ippp_rppp  kppp_rppp     GDP_rppp      igov_n
kgov_n  \
4585  1032.477661        0.0        0.0  1184.217896  386.210419
4237.776855
4586  1045.347778        0.0        0.0  1240.782837  487.095917
4820.836914
4587  1055.095703        0.0        0.0  1289.984375  463.895630
5634.371094
4588  1099.467773        0.0        0.0  1414.181274  525.495972
6550.002930
4589  1145.846680        0.0        0.0  1498.287354  563.203918
7568.798828

         ipriv_n      kpriv_n  kppp_n        GDP_n
income
4585  234.167694  3803.553223     0.0  2845.340820  Emerging Market
Economies
```

```
4586  241.991257  4132.000000     0.0  3183.659424  Emerging Market
Economies
4587  420.632812  4556.377930     0.0  3618.646973  Emerging Market
Economies
4588  486.597565  5250.937988     0.0  4293.629883  Emerging Market
Economies
4589  649.115295  6028.831055     0.0  4932.776367  Emerging Market
Economies
```

##With Single factor

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Define features and target variable
X = pd.DataFrame(df,columns=['kgov_rppp'])
y = df['GDP_rppp']



# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Print the shapes of the resulting datasets
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

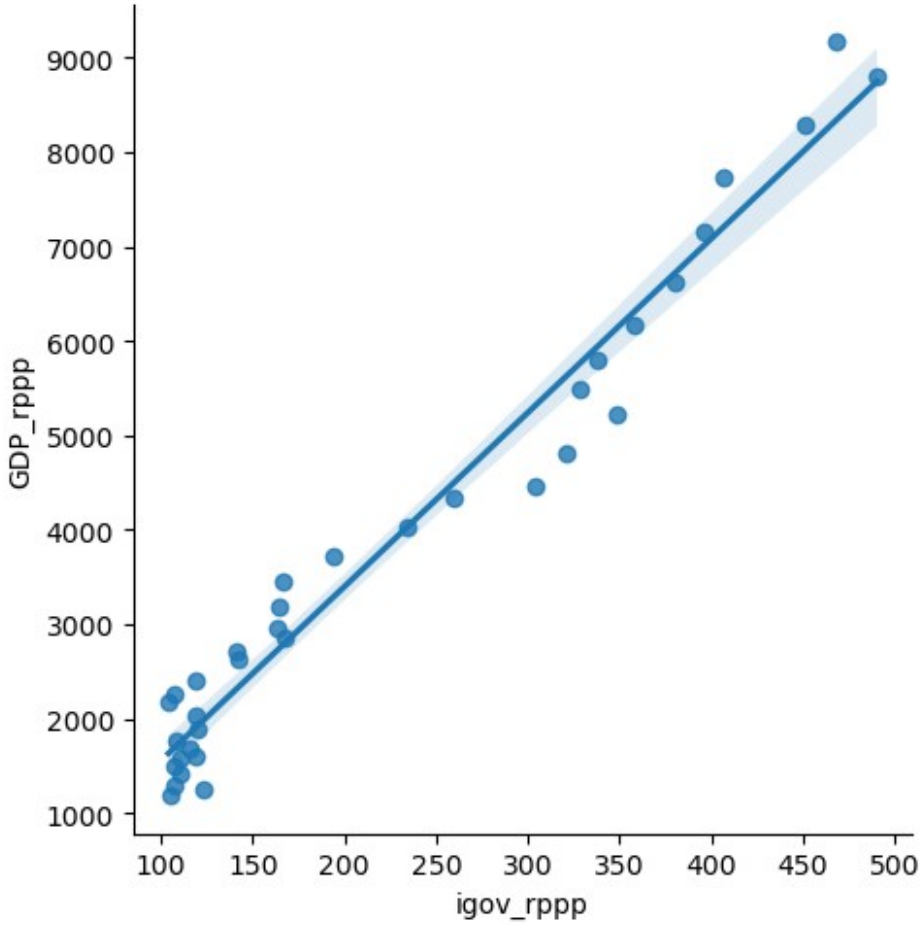
```
X_train shape: (28, 1)
X_test shape: (7, 1)
y_train shape: (28,)
y_test shape: (7,)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
le = LinearRegression()
le.fit(X_train, y_train)
y_pred = le.predict(X_test)
print(f"Mean squared error: {mean_squared_error(y_test, y_pred)}")
print(f"R2 score: {le.score(X_test, y_test)}")
```

```
Mean squared error: 271024.0368077816
R2 score: 0.8635594312987258
```

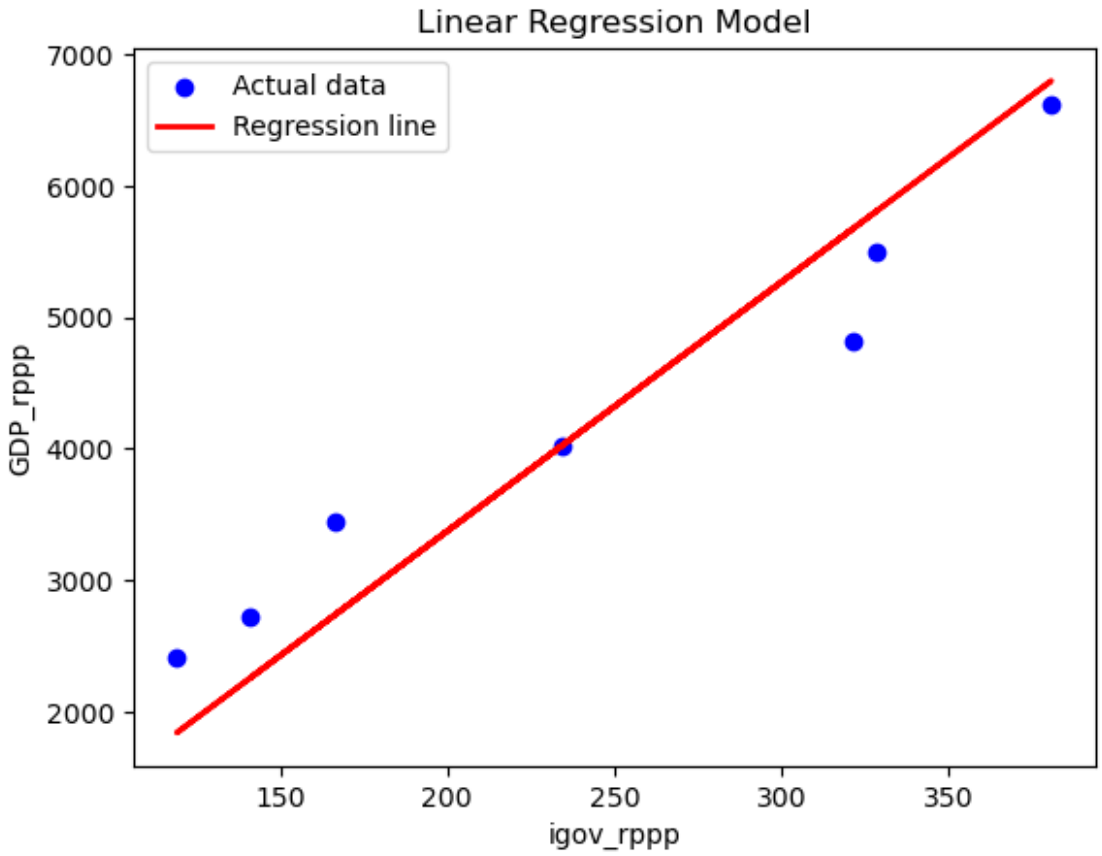
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lmplot(x='kgov_rppp', y='GDP_rppp' , data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x281d365cbf0>
```

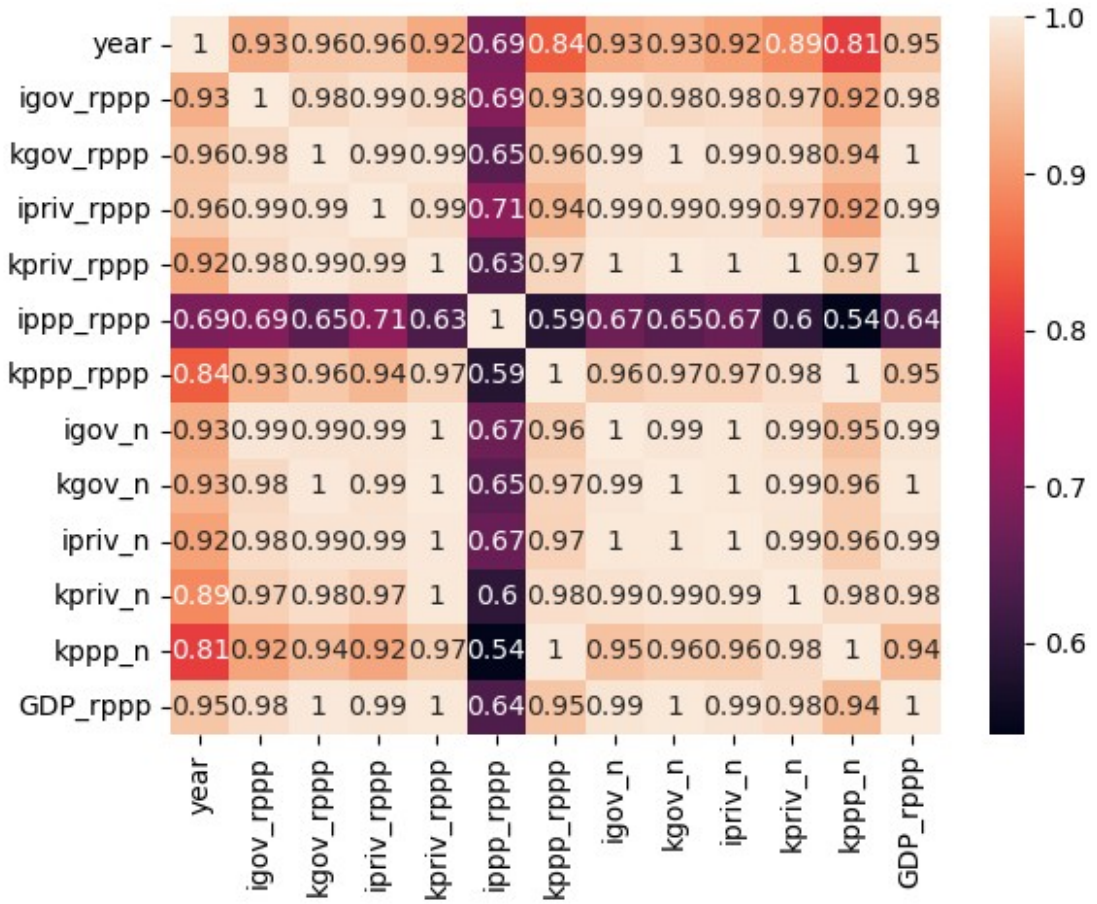


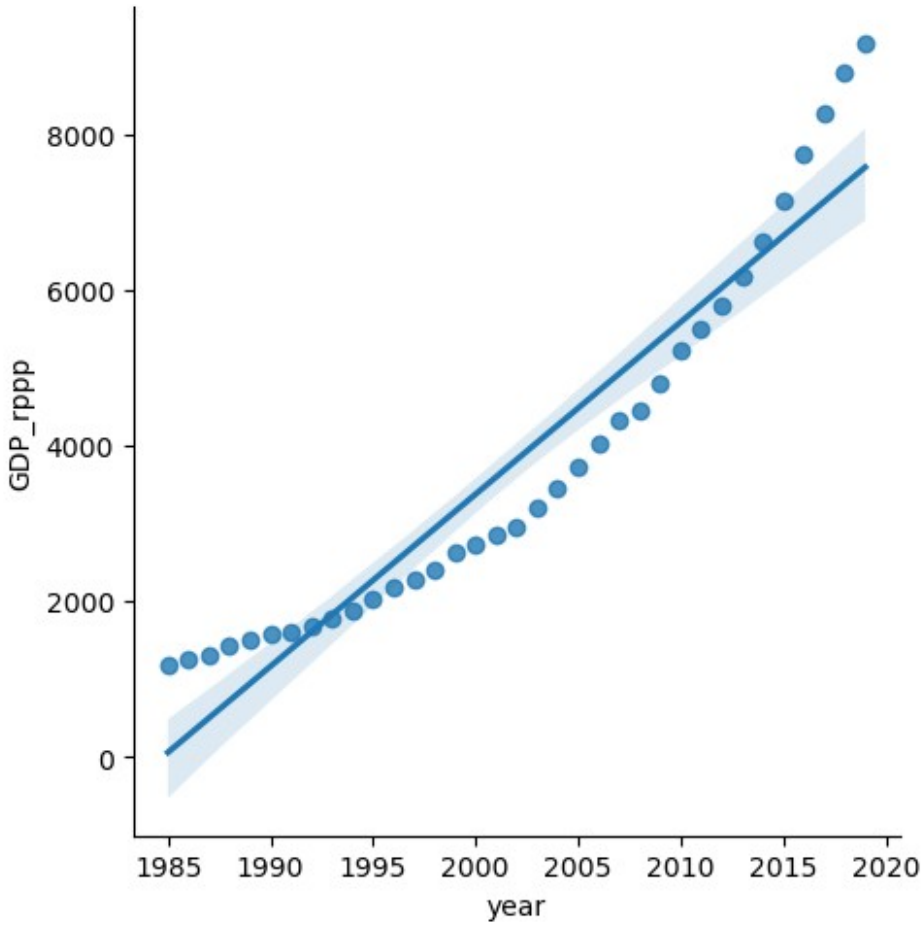

```
# Plot the data points
plt.scatter(X_test, y_test, color='blue', label='Actual data')

# Plot the regression line
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression
line')

# Add labels and title
plt.xlabel('igov_rppp')
plt.ylabel('GDP_rppp')
plt.title('Linear Regression Model')
plt.legend()

# Display the plot
plt.show()
```

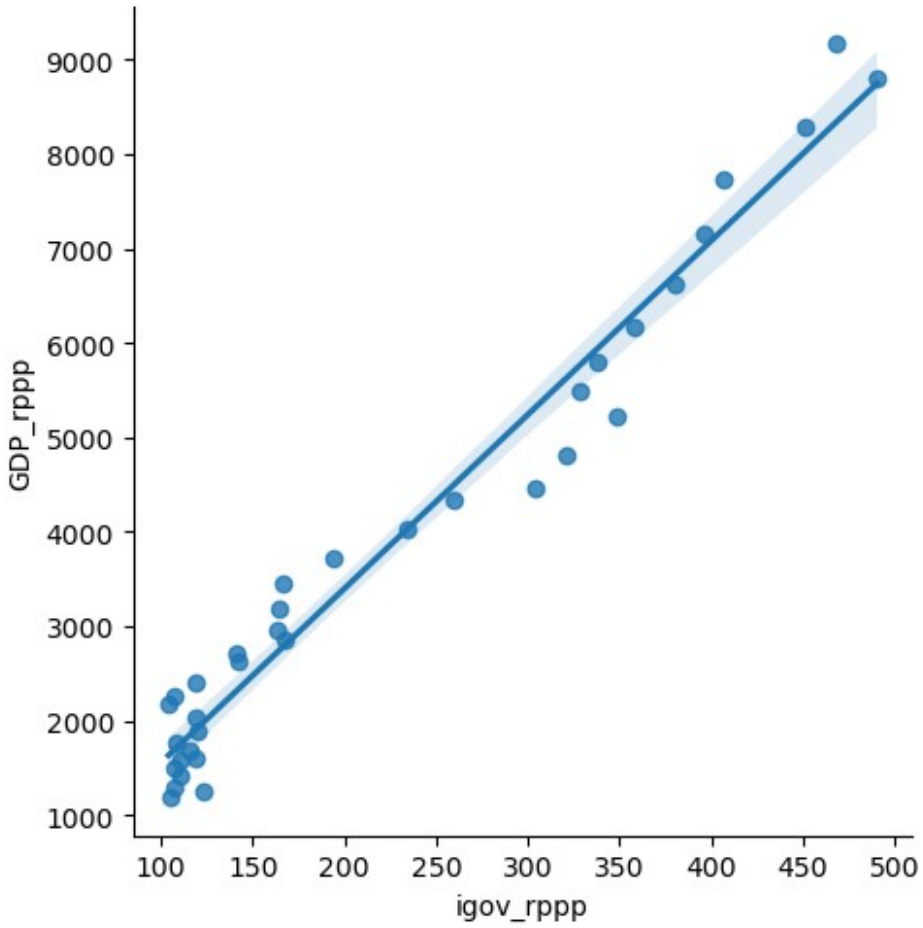
```
# Print the equation of the line
slope = le.coef_[0]
intercept = le.intercept_
print(f"Equation of the line: GDP_rppp = {slope:.2f} * igov_rppp +
{intercept:.2f}")
```

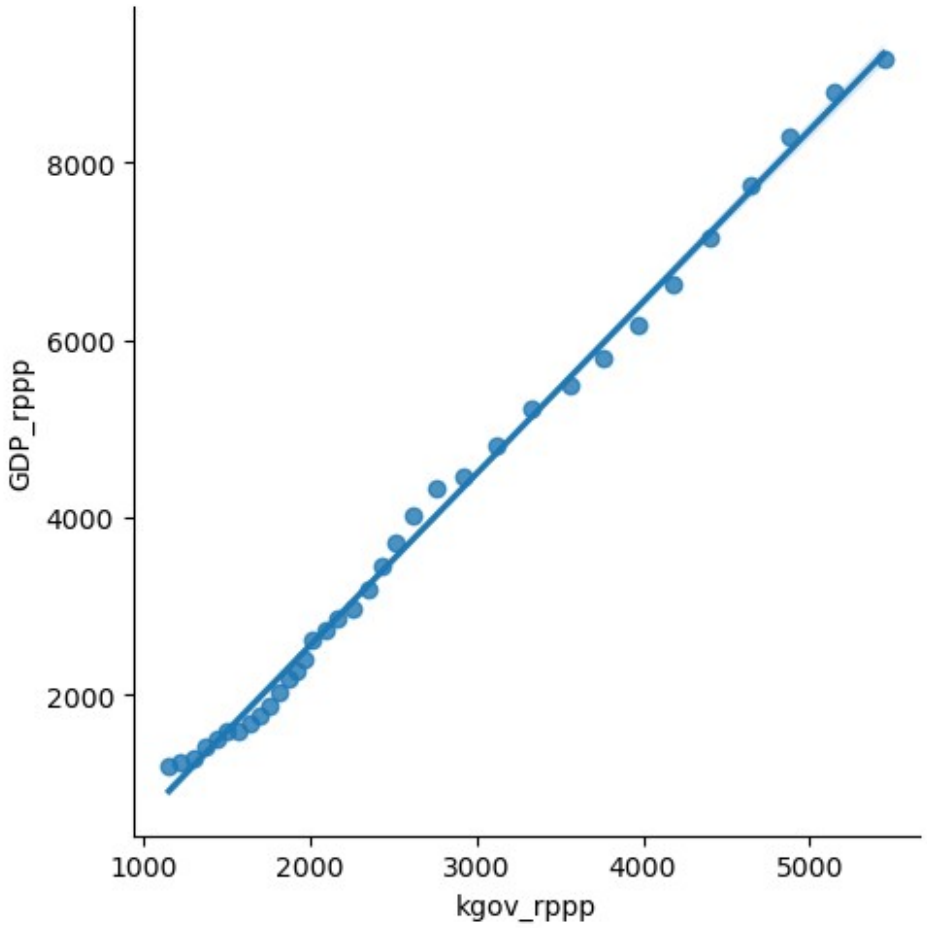


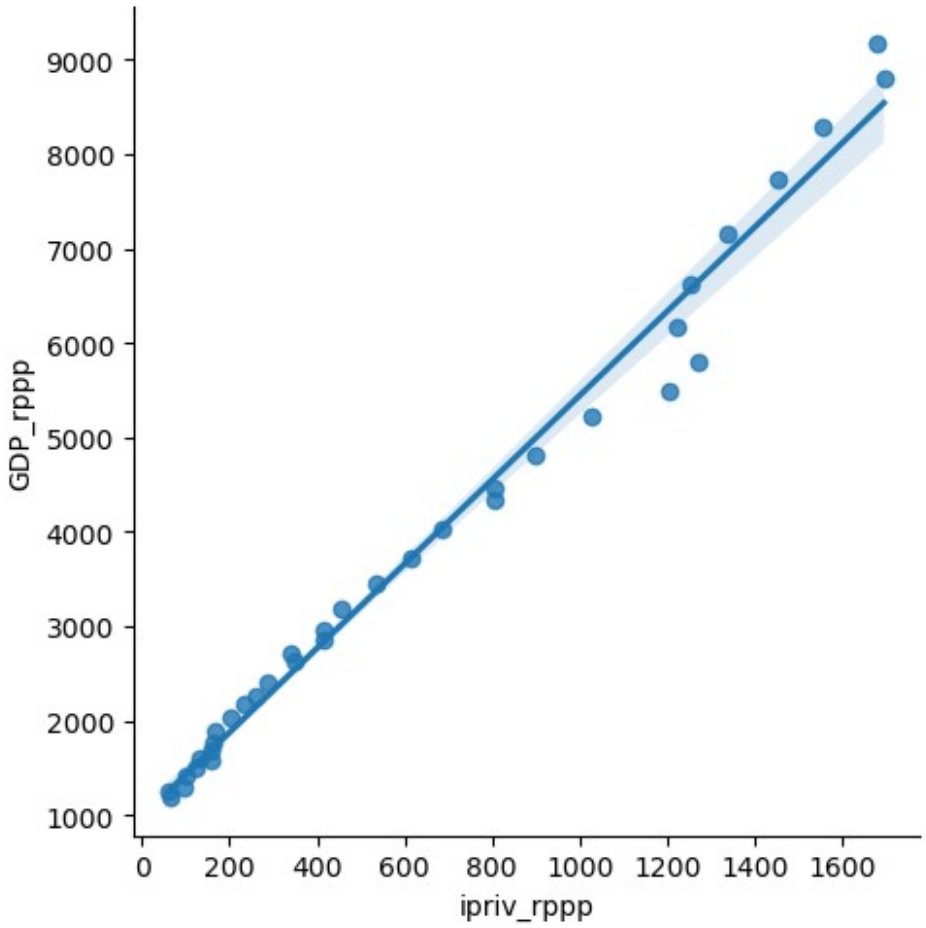

```
Equation of the line: GDP_rppp = 18.93 * igov_rppp + -412.54


# with multiple data

df_corr= pd.DataFrame(df,columns=['year', 'igov_rppp', 'kgov_rppp',
       'ipriv_rppp', 'kpriv_rppp', 'ippp_rppp', 'kppp_rppp','igov_n',
'kgov_n', 'ipriv_n', 'kpriv_n', 'kppp_n','GDP_rppp'])
hm = sns.heatmap(df_corr.corr(), annot=True)
for col in df_corr.columns:
    if col != 'GDP_rppp':
        sns.lmplot(x=col, y='GDP_rppp', data=df)
```

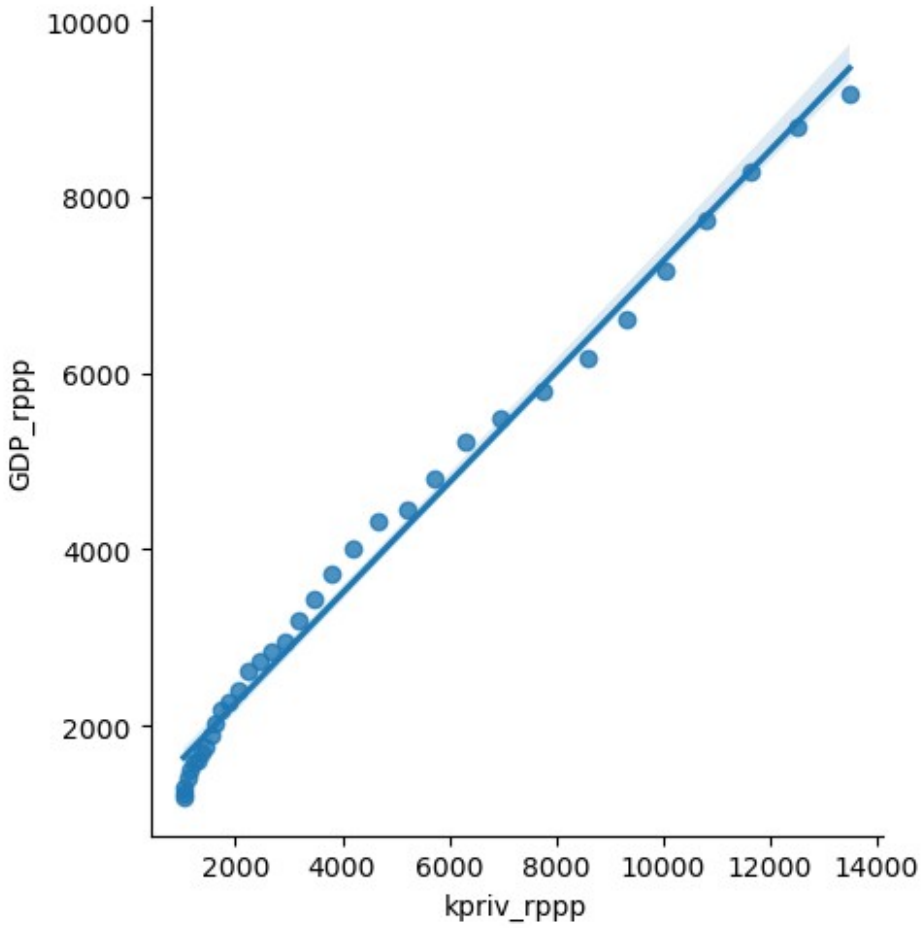

year 1 0.93 0.96 0.96 0.92 0.69 0.84 0.93 0.93 0.92 0.89 0.81 0.95
igov_rppp 0.93 1 0.98 0.99 0.98 0.69 0.93 0.99 0.98 0.98 0.97 0.92 0.98
kgov_rppp 0.96 0.98 1 0.99 0.99 0.65 0.96 0.99 1 0.99 0.98 0.94 1
ipriv_rppp 0.96 0.99 0.99 1 0.99 0.71 0.94 0.99 0.99 0.99 0.97 0.92 0.99
kpriv_rppp 0.92 0.98 0.99 0.99 1 0.63 0.97 1 1 1 1 0.97 1
ippp_rppp 0.69 0.69 0.65 0.71 0.63 1 0.59 0.67 0.65 0.67 0.6 0.54 0.64
kppp_rppp 0.84 0.93 0.96 0.94 0.97 0.59 1 0.96 0.97 0.97 0.98 1 0.95
igov_n 0.93 0.99 0.99 0.99 1 0.67 0.96 1 0.99 1 0.99 0.95 0.99
kgov_n 0.93 0.98 1 0.99 1 0.65 0.97 0.99 1 1 0.99 0.96 1
ipriv_n 0.92 0.98 0.99 0.99 1 0.67 0.97 1 1 1 0.99 0.96 0.99
kpriv_n 0.89 0.97 0.98 0.97 1 0.6 0.98 0.99 0.99 0.99 1 0.98 0.98
kppp_n 0.81 0.92 0.94 0.92 0.97 0.54 1 0.95 0.96 0.96 0.98 1 0.94
GDP_rppp 0.95 0.98 1 0.99 1 0.64 0.95 0.99 1 0.99 0.98 0.94 1
year
igov_rppp
kgov_rppp
ipriv_rppp
kpriv_rppp
ippp_rppp
kppp_rppp
igov_n
kgov_n
ipriv_n
kpriv_n
kppp_n
GDP_rppp
1.0
0.9
0.8
0.7
0.6

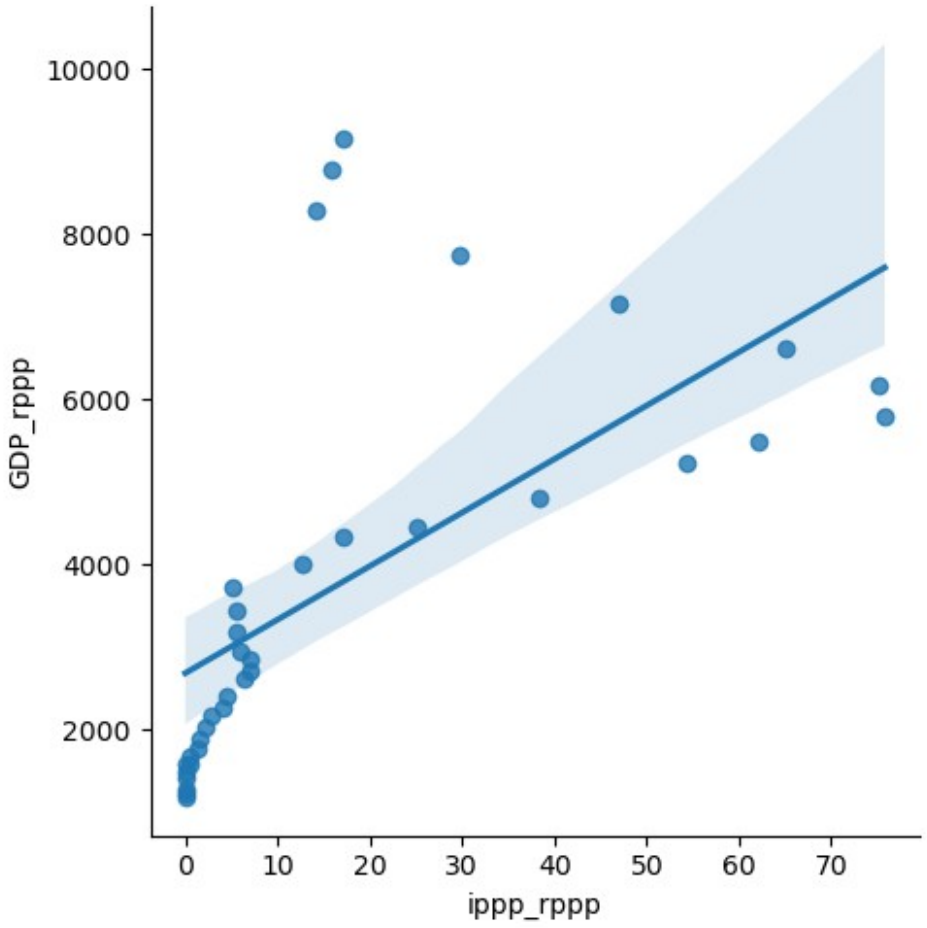

8000
6000
4000
2000
0
GDP_rppp
1985
1990
1995
2000
2005
2010
2015
2020
year

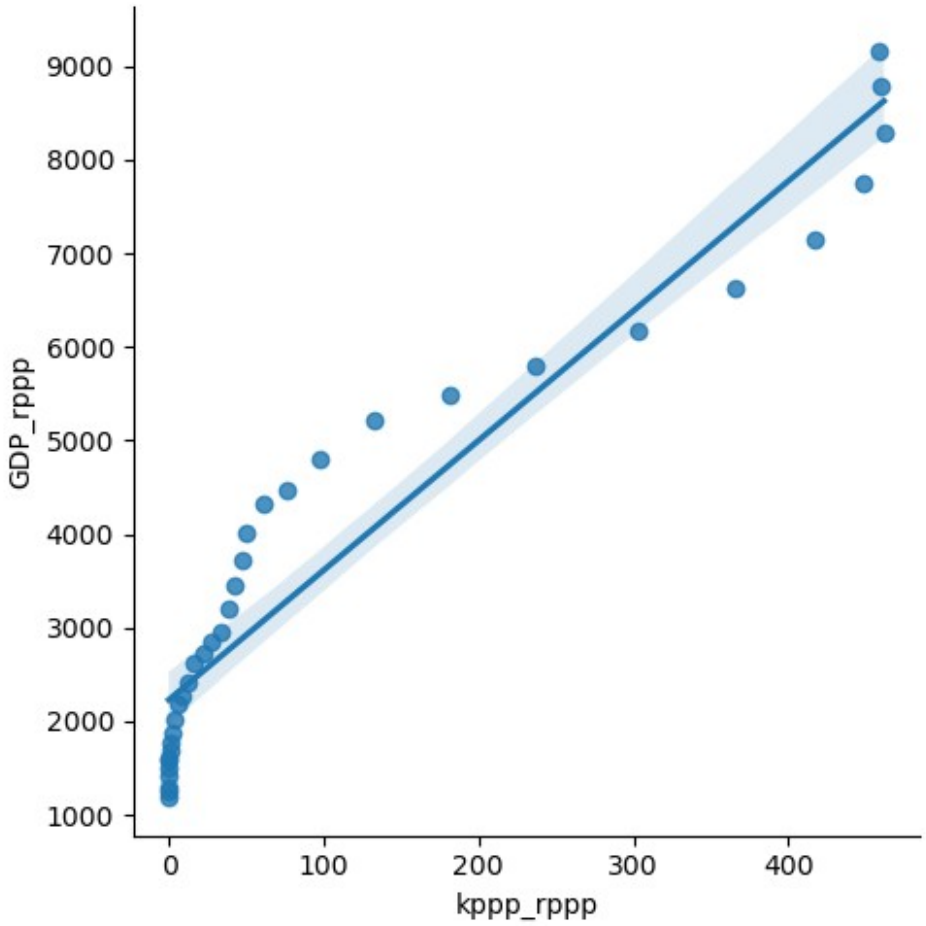

GDP_rppp
igov_rppp
9000
8000
7000
6000
5000
4000
3000
2000
1000
100
150
200
250
300
350
400
450
500

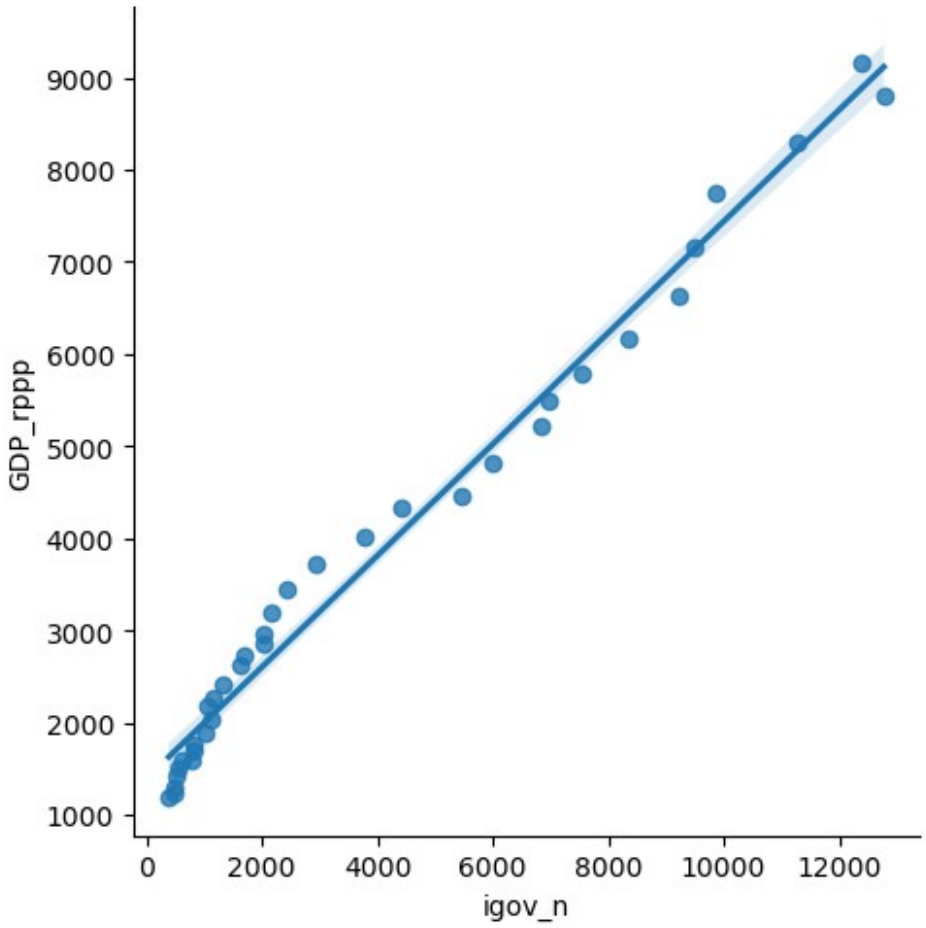

8000
6000
4000
2000
GDP_rppp
1000
2000
3000
4000
5000
kgov_rppp

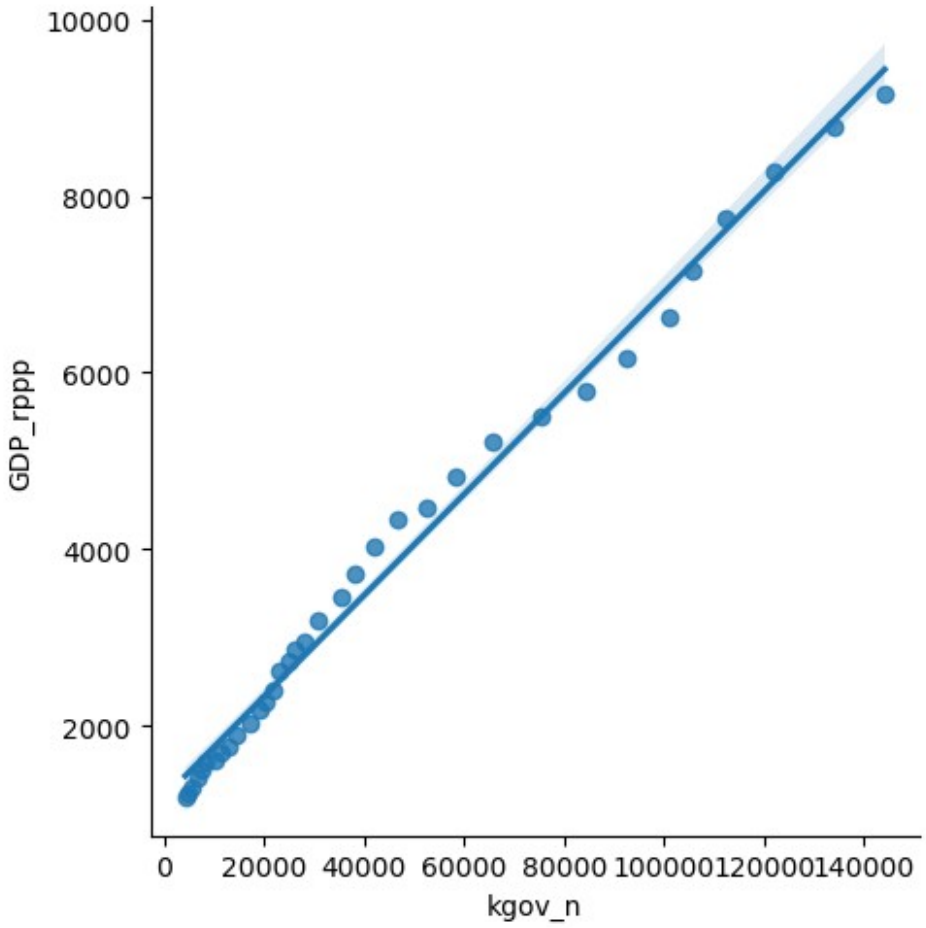

9000
8000
7000
6000
5000
4000
3000
2000
1000
GDP_rppp
0
200
400
600
800
1000
1200
1400
1600
ipriv_rppp

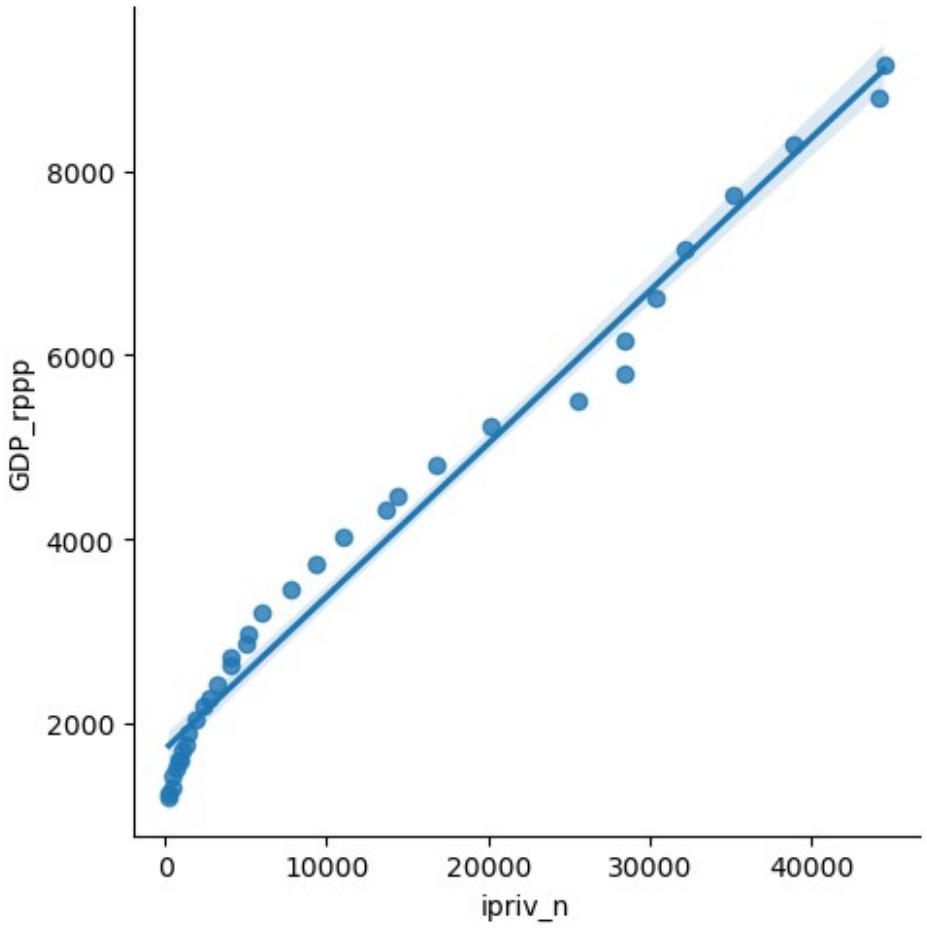

10000
8000
6000
4000
2000
GDP_rppp
2000
4000
6000
8000
10000
12000
14000
kpriv_rppp

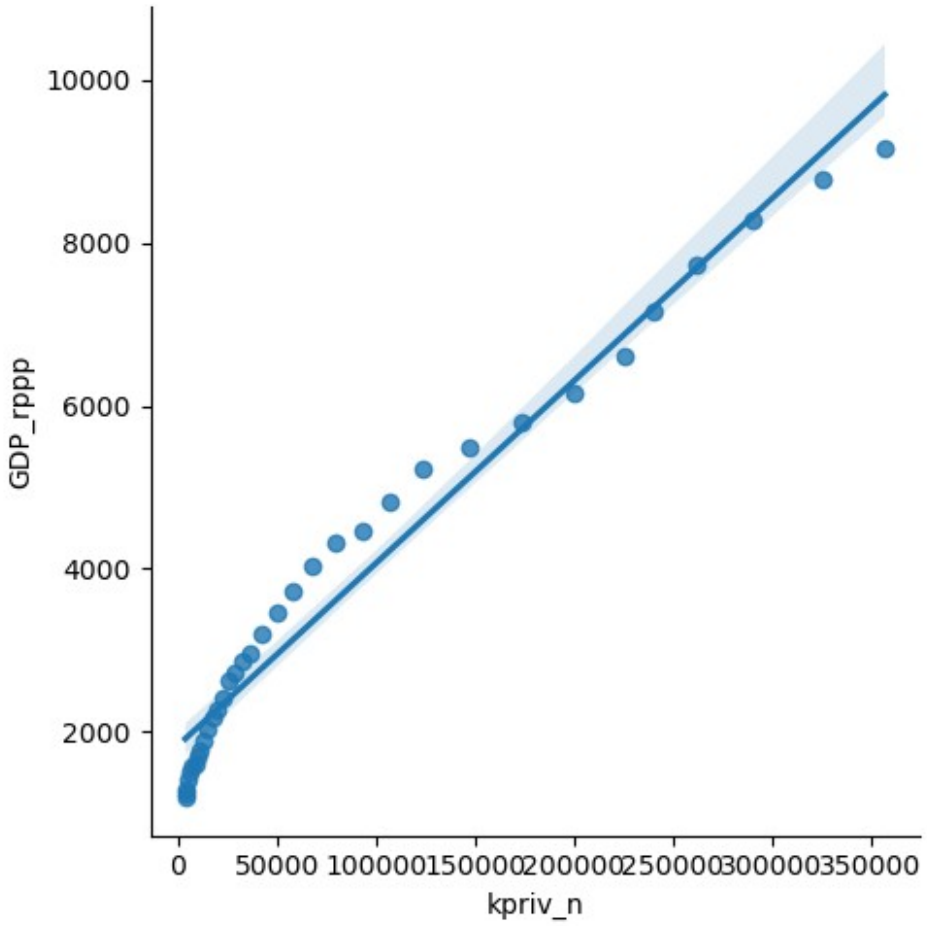

10000
8000
6000
4000
2000
GDP_rppp
0
10
20
30
40
50
60
70
ippp_rppp

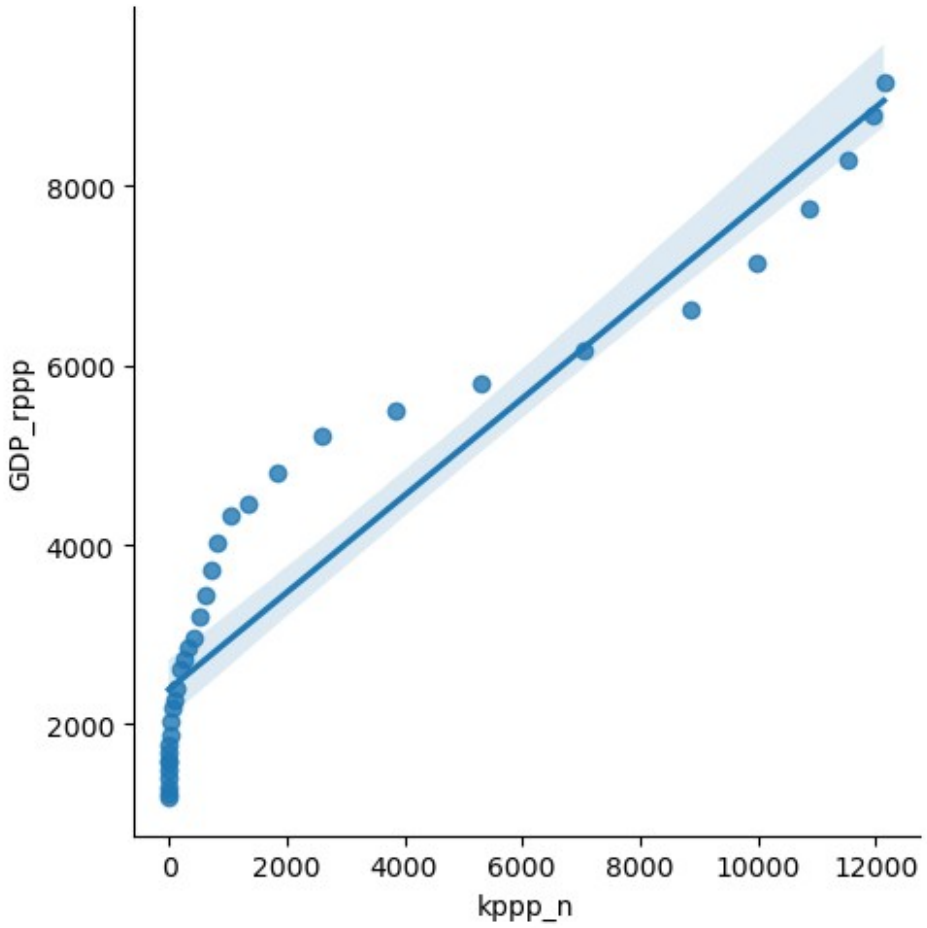

9000
8000
7000
6000
5000
4000
3000
2000
1000
GDP_rppp
0
100
200
300
400
kppp_rppp


GDP_rppp
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
2000
4000
6000
8000
10000
12000
igov_n


10000
8000
6000
4000
2000
GDP_rppp
0
20000
40000
60000
80000
100000
120000
140000
kgov_n


8000
6000
4000
2000
GDP_rppp
0
10000
20000
30000
40000
ipriv_n


10000
8000
6000
4000
2000
GDP_rppp
0
50000
100000
150000
200000
250000
300000
350000
kpriv_n




```
# Define features and target variable
X = pd.DataFrame(df,columns=['year', 'igov_rppp', 'kgov_rppp',
       'ipriv_rppp', 'kpriv_rppp', 'kppp_rppp','igov_n', 'kgov_n',
'ipriv_n', 'kpriv_n', 'kppp_n','income'])
y = df['GDP_rppp']

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object']).columns
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns

# Preprocessing for numerical data
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

# Preprocessing for categorical data
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
```

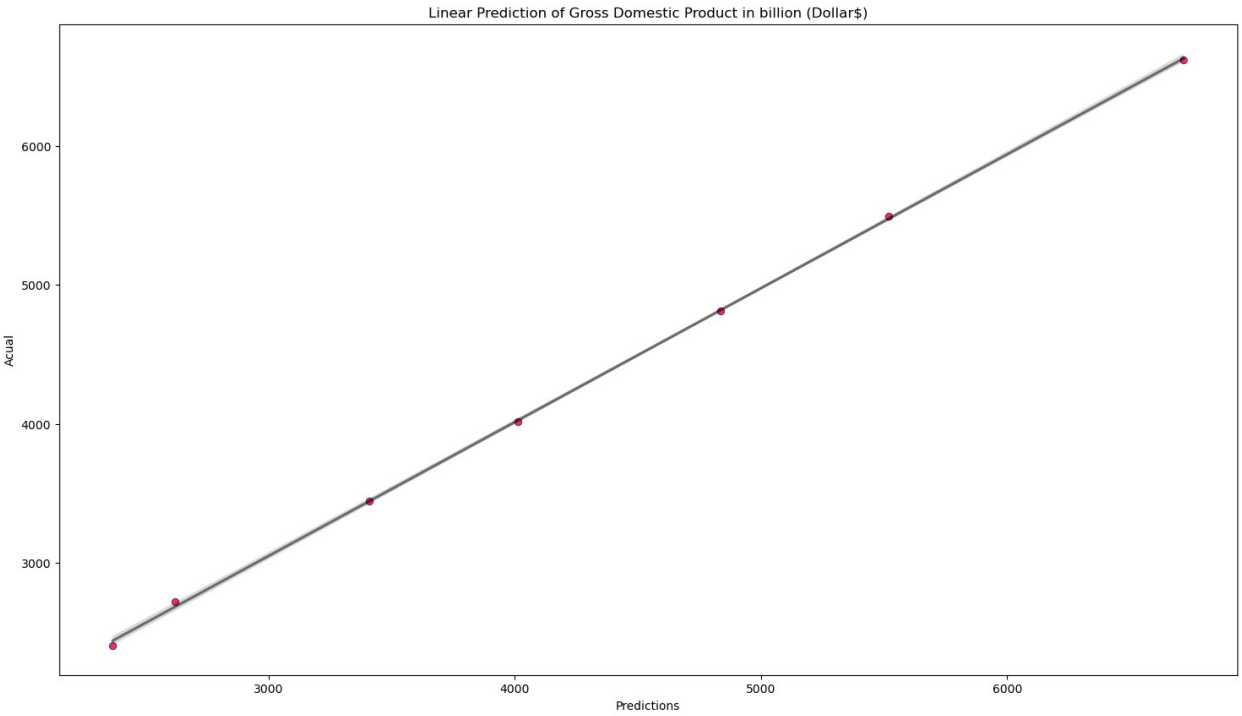
```
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Preprocess the data
X = preprocessor.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Print the shapes of the resulting datasets
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (28, 12)
X_test shape: (7, 12)
y_train shape: (28,)
y_test shape: (7,)
```

```
le = LinearRegression()
le.fit(X_train, y_train)
y_pred = le.predict(X_test)
print(f"Mean squared error: {mean_squared_error(y_test, y_pred)}")
print(f"R2 score: {le.score(X_test, y_test)}")
# Plot the data points
```

```
Mean squared error: 3516.7272923587575
R2 score: 0.9982295877613356
```

```
plt.figure(figsize=(18,10))
sns.regplot(x=y_pred, y=y_test,
            scatter_kws={'color':'red', 'edgecolor':'blue',
'linewidths':0.7},
            line_kws={'color':'black', 'alpha':0.5})

plt.xlabel('Predictions')
plt.ylabel('Acual')
plt.title("Linear Prediction of Gross Domestic Product in billion
(Dollar$)")
plt.show()
```


Linear Prediction of Gross Domestic Product in billion (Dollar$)
Acual
Predictions
6000
5000
4000
3000
3000
4000
5000
6000