Data Science Project

Total Population of All the Countries

Meet our team

Group number- 34

Abhay Panwar

Jatin Kumawat

Chetan Kumar

Gaikwad Chandan Mahadeo

Devansh Mishra

Mallempadi Yashaswini

Agenda

- Data collection
- Data preprocessing and cleaning
- Visualization
- Data statistics
- Hypothetical Statement
- Prediction Task

DATA COLLECTION

- All the data is collected from Kaggle.
- Total Population by sex, annually from 1950 to 2008.
- PopMale: Total male population (thousands)
- PopFemale: Total female population (thousands)
- PopTotal: Total Population, Both sexes (thousands)
- PopDensity: Population per square kilometer (thousands)

- https://population.un.org/wpp/Download/Standard/CSV/

Python Modules used during project

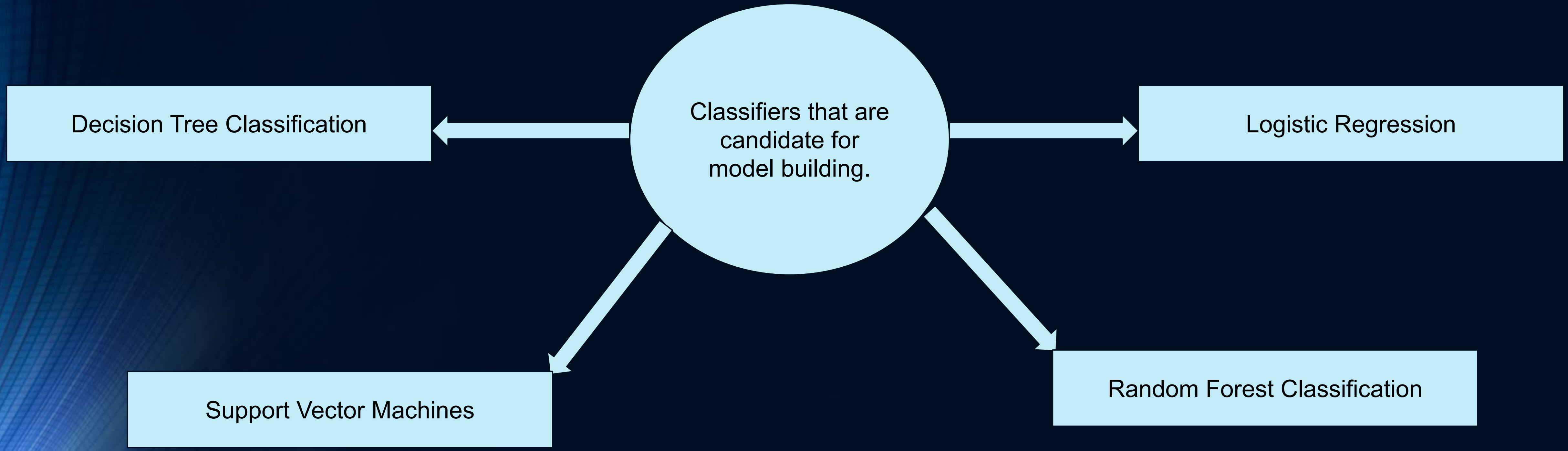
- Numpy
- Pandas
- Plotly
- csv
- Sklearn

Coded in Google Colab

Our Machine Learning Model

- Built using Sklearn and google Collab
- Built by Classifiers.
- Predict Location ID of a country based on given various attributes related to Population and Year.

- Information about working of various classifiers is taken from www.Datacamp.org



Data preprocessing and cleaning

- Taking only medium variant of Population in consideration

- Removing column varID, variant and MidPeriod

- Removed Some names of countries which are not necessary

- Dividing test data and train data for year>=2009 and year<=2008 in 70/30

- While Building Model for prediction, Accuracy was coming significantly low. So, we used built-in train_test_split in 70/30 split as it provide good accuracy.

- Faced One Problem, Many Classifiers needed both numbers as input and output. So using Attribute “Location” possess a problem. Instead attribute “LocID” is used along with other numeric type attributes.

- Columns of csv data file are rearranged manually in excel according to the requirement of Model.

	A	B	C	D	E	F	R
1	Time	PopMale	PopFemale	PopTotal	PopDensit	LocID	Location
2	1961	4730.25	4439.156	9169.406	14.045	4	Afghanistan
3	1962	4816.05	4535.392	9351.442	14.324	4	Afghanistan
4	1963	4907.03	4636.17	9543.2	14.618	4	Afghanistan
5	1964	5003.245	4741.527	9744.772	14.926	4	Afghanistan
6	1965	5104.765	4851.553	9956.318	15.25	4	Afghanistan
7	1966	5210.122	4964.718	10174.84	15.585	4	Afghanistan
8	1967	5319.123	5080.813	10399.94	15.93	4	Afghanistan
9	1968	5434.458	5202.606	10637.06	16.293	4	Afghanistan
10	1969	5559.836	5333.936	10893.77	16.686	4	Afghanistan
11	1970	5697.024	5476.63	11173.65	17.115	4	Afghanistan
12	1971	5845.351	5630.099	11475.45	17.577	4	Afghanistan
13	1972	6000.895	5790.327	11791.22	18.061	4	Afghanistan
14	1973	6157.843	5951.12	12108.96	18.548	4	Afghanistan
15	1974	6308.583	6104.377	12412.96	19.013	4	Afghanistan
16	1975	6446.273	6242.891	12689.16	19.436	4	Afghanistan
17	1976	6573.732	6369.361	12943.09	19.825	4	Afghanistan
18	1977	6689.144	6482.15	13171.29	20.175	4	Afghanistan
19	1978	6776.023	6565.176	13341.2	20.435	4	Afghanistan
20	1979	6813.205	6597.855	13411.06	20.542	4	Afghanistan
21	1980	6788.273	6568.227	13356.5	20.458	4	Afghanistan
22	1981	6698.73	6472.949	13171.68	20.175	4	Afghanistan
23	1982	6557.673	6324.845	12882.52	19.732	4	Afghanistan
24	1983	6388.704	6149.028	12537.73	19.204	4	Afghanistan
25	1984	6224.263	5980.043	12204.31	18.694	4	Afghanistan
26	1985	6091.889	5846.315	11938.2	18.286	4	Afghanistan
27	1986	5988.52	5747.657	11736.18	17.977	4	Afghanistan
28	1987	5918.156	5686.382	11604.54	17.775	4	Afghanistan
29	1988	5922.669	5695.339	11618.01	17.796	4	Afghanistan
30	1989	6054.345	5814.528	11868.87	18.18	4	Afghanistan

Data file “TotalPopulation_all.csv” in MS-excel

CODES:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
population = pd.read_csv('/content/WPP2019_TotalPopulationBySex.csv',encoding='cp1252')
population.head()
```

	LocID	Location	VarID	Variant	Time	MidPeriod	PopMale	PopFemale	PopTotal	PopDensity
0	4	Afghanistan	2	Medium	1950	1950.5	4099.243	3652.874	7752.117	11.874
1	4	Afghanistan	2	Medium	1951	1951.5	4134.756	3705.395	7840.151	12.009
2	4	Afghanistan	2	Medium	1952	1952.5	4174.450	3761.546	7935.996	12.156
3	4	Afghanistan	2	Medium	1953	1953.5	4218.336	3821.348	8039.684	12.315
4	4	Afghanistan	2	Medium	1954	1954.5	4266.484	3884.832	8151.316	12.486

CODES:

```
population = population.loc[population.Variant == 'Medium']
population.drop([ 'LocID','VarID','MidPeriod','Variant'],inplace=True,axis=1)
population.head()
```

	Location	Time	PopMale	PopFemale	PopTotal	PopDensity
0	Afghanistan	1950	4099.243	3652.874	7752.117	11.874
1	Afghanistan	1951	4134.756	3705.395	7840.151	12.009
2	Afghanistan	1952	4174.450	3761.546	7935.996	12.156
3	Afghanistan	1953	4218.336	3821.348	8039.684	12.315
4	Afghanistan	1954	4266.484	3884.832	8151.316	12.486

```python
mask=False
for col in population.columns:
   mask=mask|population[col].isnull()
datanulls=population[mask]
population.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

	Location	Time	PopMale	PopFemale	PopTotal	PopDensity
0	Afghanistan	1950	4099.243	3652.874	7752.117	11.874
1	Afghanistan	1951	4134.756	3705.395	7840.151	12.009
2	Afghanistan	1952	4174.450	3761.546	7935.996	12.156
3	Afghanistan	1953	4218.336	3821.348	8039.684	12.315
4	Afghanistan	1954	4266.484	3884.832	8151.316	12.486
...	...	...	...	...	...	...
280194	Zimbabwe	2096	15008.463	15932.316	30940.779	79.981
280195	Zimbabwe	2097	15008.822	15943.386	30952.208	80.011
280196	Zimbabwe	2098	15007.570	15952.241	30959.811	80.031
280197	Zimbabwe	2099	15004.963	15959.089	30964.052	80.041
280198	Zimbabwe	2100	15001.252	15964.169	30965.421	80.045

66893 rows × 6 columns

```
population = population.loc[(population.Time > 1960) & (population.Time < 2020)]
population.to_csv('TotalPopulation.csv')
population.head()
```

	Location	Time	PopMale	PopFemale	PopTotal	PopDensity
11	Afghanistan	1961	4730.250	4439.156	9169.406	14.045
12	Afghanistan	1962	4816.050	4535.392	9351.442	14.324
13	Afghanistan	1963	4907.030	4636.170	9543.200	14.618
14	Afghanistan	1964	5003.245	4741.527	9744.772	14.926
15	Afghanistan	1965	5104.765	4851.553	9956.318	15.250

```python
Country_name = ['Location', 'Afghanistan', 'Albania', 'Algeria', 'Angola', 'Argentina', 'Armenia', 'Aruba', 'Austria',
'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium','Belize','Benin', 'Bhutan', 'Botswana',
'Brazil', 'Bulgaria', 'Burundi', 'Cambodia', 'Cameroon', 'Canada', 'Caribbean', 'Chad', 'Chile', 'China', 'Colombia',
'Comoros', 'Congo', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark', 'Djibouti', 'Dominica', 'Ecuador',
'Egypt', 'Eritrea', 'Ethiopia', 'Fiji', 'Finland', 'France', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
'Greenland', 'Guadeloupe', 'Guatemala', 'Guinea', 'Haiti', 'Honduras', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iraq',
'Ireland', 'Israel', 'Italy', 'Japan', 'Kazakhstan', 'Kenya', 'Kuwait', 'Kyrgyzstan', 'Latin America and the Caribbean',
'Latvia', 'Lebanon', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Malawi', 'Malaysia', 'Maldives',
'Mali', 'Malta', 'Martinique', 'Mauritania', 'Mauritius', 'Mayotte', 'Mexico', 'Micronesia', 'Mongolia', 'Morocco',
'Mozambique', 'Myanmar', 'Namibia', 'Nepal', 'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger',
'Nigeria', 'North Macedonia', 'Norway', 'Oceania', 'Oman', 'Pakistan', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
'Philippines', 'Poland', 'Polynesia', 'Portugal', 'Puerto Rico', 'Qatar', 'Romania', 'Rwanda', 'Saint Vincent and the
Grenadines', 'Samoa', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone',
'Singapore', 'Slovakia', 'Slovenia', 'South Africa', 'Spain', 'Sri Lanka', 'State of Palestine', 'Sudan', 'Suriname',
'Sweden', 'Switzerland', 'Tajikistan', 'Thailand', 'Togo', 'Tonga', 'Tunisia', 'Turkey', 'Turkmenistan', 'Uganda',
'Ukraine', 'United Arab Emirates', 'United Kingdom', 'United States of America', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Viet
Nam', 'Yemen', 'Zambia', 'Zimbabwe']
with open('TotalPopulation.csv', "r") as f:
    data = list(csv.reader(f))
with open('TotalPopulation.csv', "w") as f:
    writer = csv.writer(f)
    for row in data:
        if row: # only use non-empty lines (or use if len(row)>2)
            if row[1] in Country_name:
                writer.writerow(row)
from google.colab import files
files.download('TotalPopulation.csv')
```

```
new_file=pd.read_csv('/content/TotalPopulation.csv' ,encoding='cp1252')
new_file.query('Time <= 2008', inplace = True)
new_file.drop(['Unnamed: 0'],axis=1,inplace=True)
new_file.to_csv('Train_data.csv')
from google.colab import files
files.download('Train_data.csv')
new_file_=pd.read_csv('/content/TotalPopulation.csv' ,encoding='cp1252')
new_file_.query('Time >= 2009', inplace = True)
new_file_.drop(['Unnamed: 0'],axis=1,inplace=True)
new_file_.to_csv('Test_data.csv')
from google.colab import files
files.download('Test_data.csv')
```

Visualisation

- To select one attribute out of four [male population, female population, total population, population density] for hypothetical testing

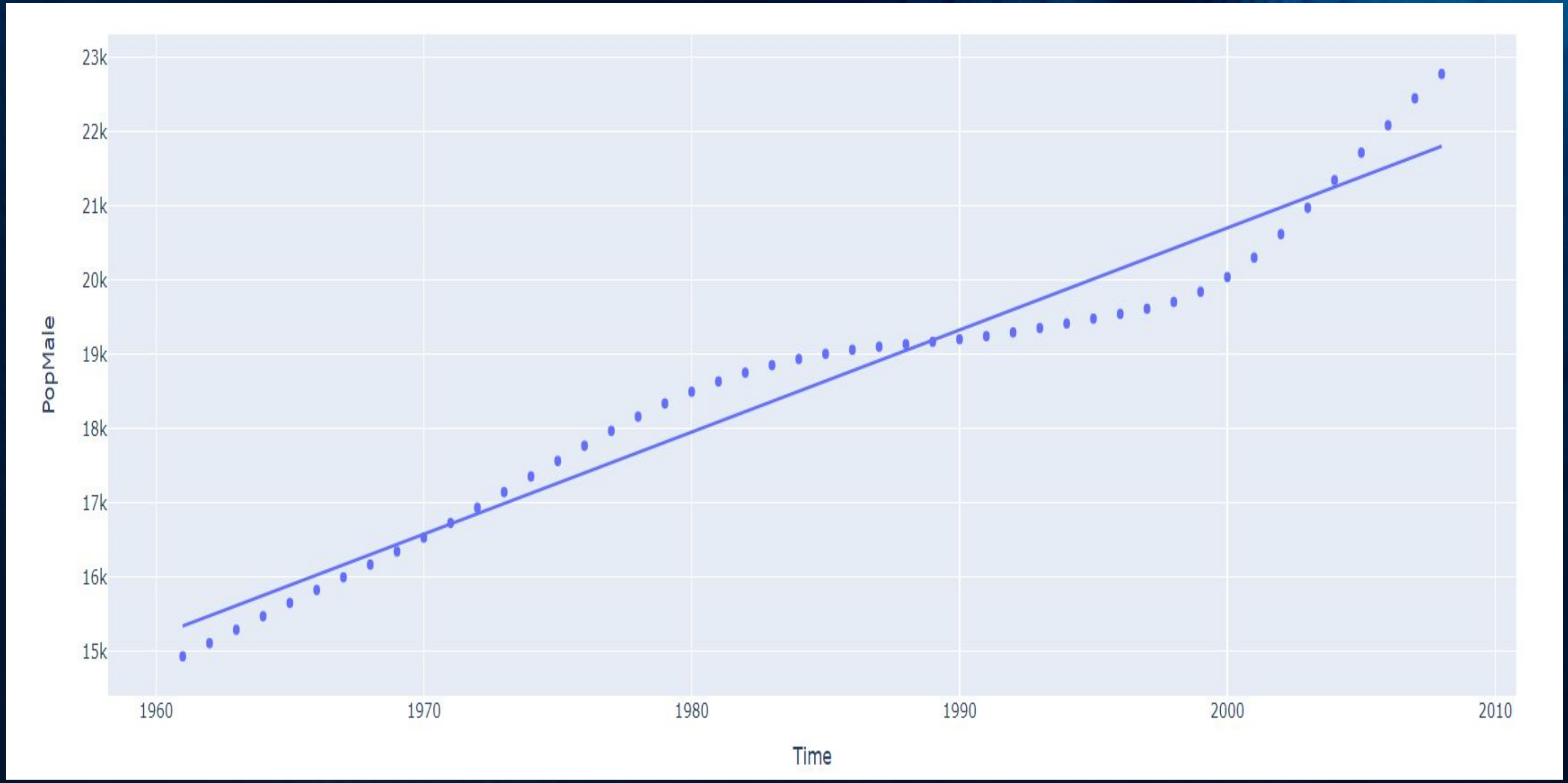
- Plotted scatter plot of all 4 attributes for 10 different countries

- Also calculated R^2 for each case using plotly

```
import csv
import pandas as pd
import plotly.express as px
data = pd.read_csv('/content/Train_data.csv',encoding='cp1252')
```

Spain

```python
data_spain = data.loc[data.Location == 'Spain']
fig2 = px.scatter(data_spain, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```



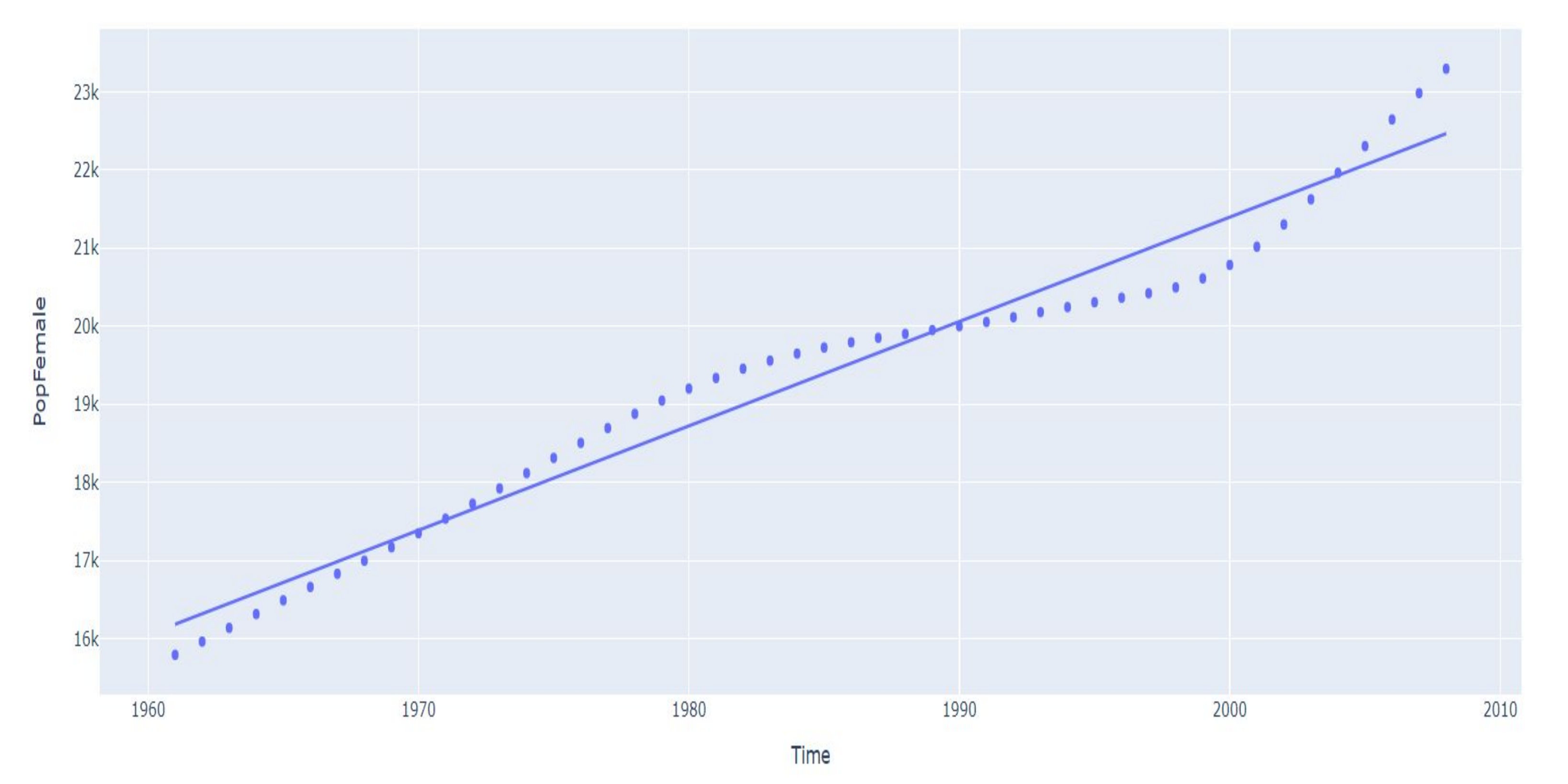
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.952
Model:	OLS	**Adj. R-squared:**	0.951
Method:	Least Squares	**F-statistic:**	917.5
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.93e-32
Time:	03:55:11	**Log-Likelihood:**	-358.75
No. Observations:	48	**AIC:**	721.5
Df Residuals:	46	**BIC:**	725.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.541e+05	9003.415	-28.227	0.000	-2.72e+05	-2.36e+05
x1	137.4192	4.537	30.290	0.000	128.287	146.551

Omnibus:	3.217	**Durbin-Watson:**	0.058
Prob(Omnibus):	0.200	**Jarque-Bera (JB):**	1.679
Skew:	0.117	**Prob(JB):**	0.432
Kurtosis:	2.114	**Cond. No.**	2.84e+05

```python
fig3 = px.scatter(data_spain, x="Time", y="PopFemale", trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```



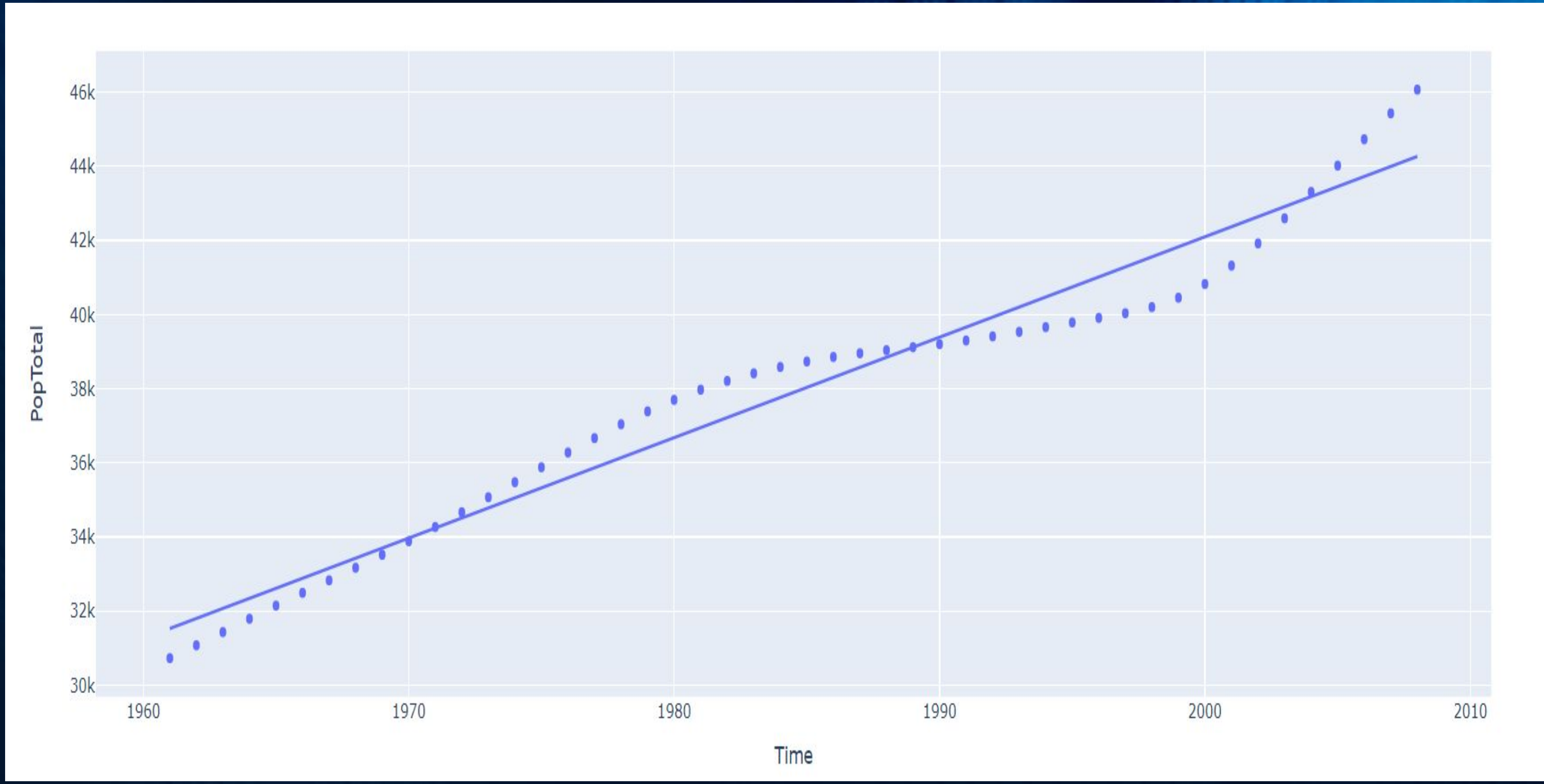
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.961
Model:	OLS	**Adj. R-squared:**	0.961
Method:	Least Squares	**F-statistic:**	1144.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	3.79e-34
Time:	03:55:16	**Log-Likelihood:**	-352.08
No. Observations:	48	**AIC:**	708.2
Df Residuals:	46	**BIC:**	711.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.457e+05	7835.730	-31.363	0.000	-2.62e+05	-2.3e+05
x1	133.5721	3.948	33.830	0.000	125.624	141.520

Omnibus:	3.692	**Durbin-Watson:**	0.059
Prob(Omnibus):	0.158	**Jarque-Bera (JB):**	1.748
Skew:	0.071	**Prob(JB):**	0.417
Kurtosis:	2.076	**Cond. No.**	2.84e+05

```python
fig4 = px.scatter(data_spain, x="Time", y="PopTotal", trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```



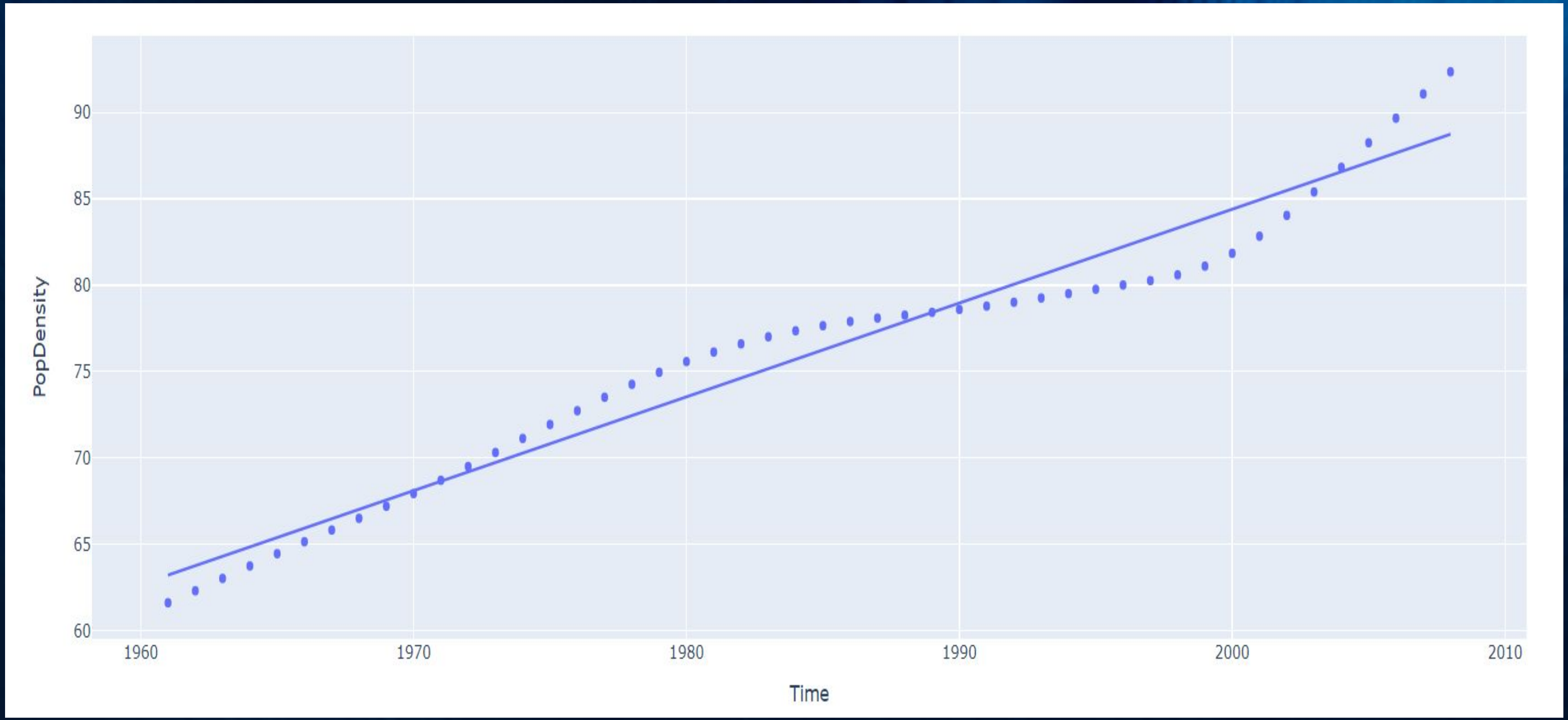
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.956
Method:	Least Squares	**F-statistic:**	1022.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.65e-33
Time:	03:55:21	**Log-Likelihood:**	-388.77
No. Observations:	48	**AIC:**	781.5
Df Residuals:	46	**BIC:**	785.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.999e+05	1.68e+04	-29.710	0.000	-5.34e+05	-4.66e+05
x1	270.9913	8.478	31.963	0.000	253.925	288.057

Omnibus:	3.444	**Durbin-Watson:**	0.059
Prob(Omnibus):	0.179	**Jarque-Bera (JB):**	1.714
Skew:	0.098	**Prob(JB):**	0.424
Kurtosis:	2.095	**Cond. No.**	2.84e+05

```python
fig5 = px.scatter(data_spain, x="Time", y="PopDensity", trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```



OLS Regression Results

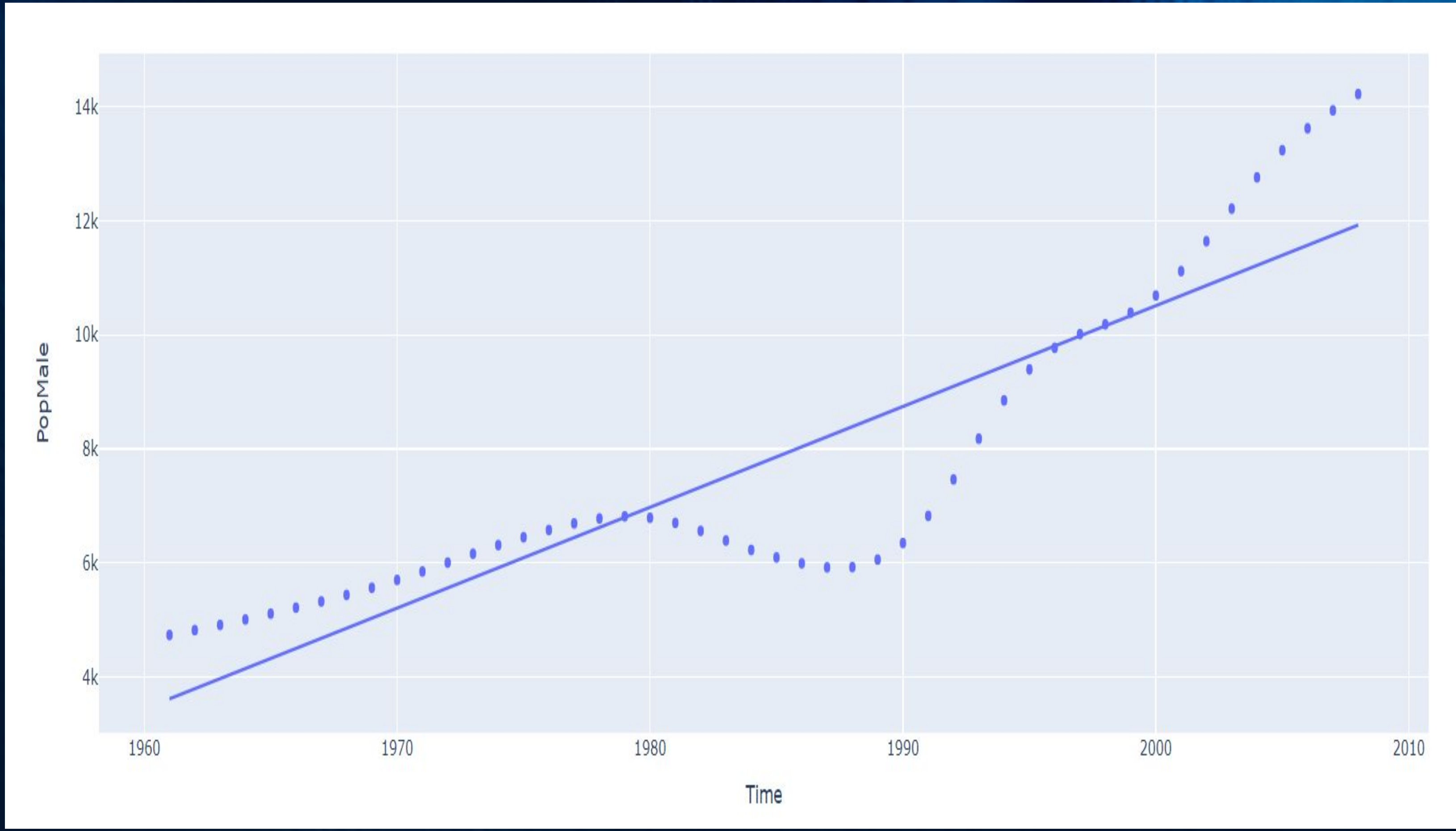
Dep. Variable:	y	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.956
Method:	Least Squares	**F-statistic:**	1022.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.64e-33
Time:	03:55:23	**Log-Likelihood:**	-90.580
No. Observations:	48	**AIC:**	185.2
Df Residuals:	46	**BIC:**	188.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1002.1728	33.732	-29.710	0.000	-1070.071	-934.275
x1	0.5433	0.017	31.963	0.000	0.509	0.577

Omnibus:	3.446	**Durbin-Watson:**	0.059
Prob(Omnibus):	0.179	**Jarque-Bera (JB):**	1.714
Skew:	0.098	**Prob(JB):**	0.424
Kurtosis:	2.095	**Cond. No.**	2.84e+05

Afghanistan

```
data_Afghanistan = data.loc[data.Location == 'Afghanistan']
fig2 = px.scatter(data_Afghanistan, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```



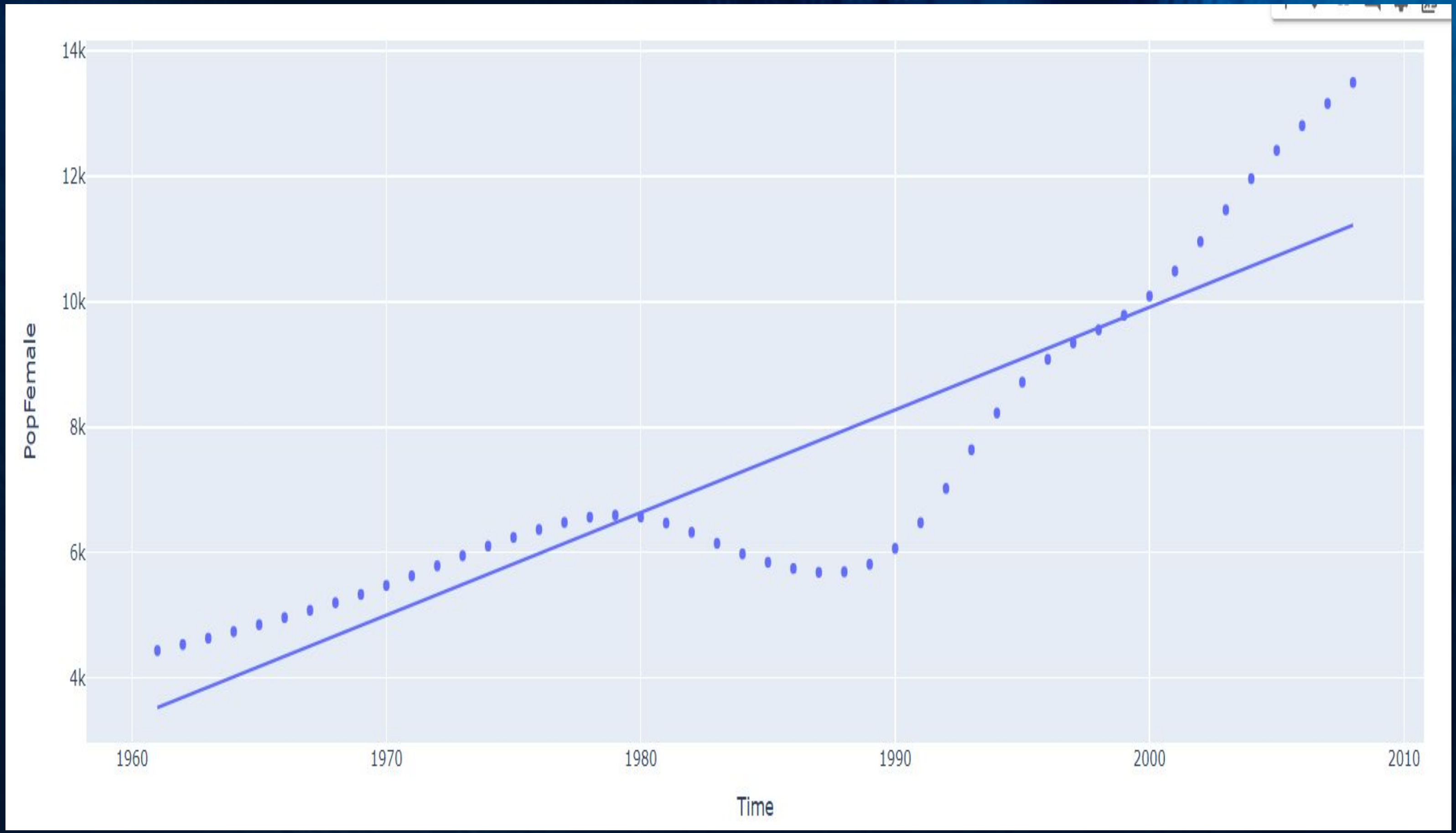
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.797
Model:	OLS	**Adj. R-squared:**	0.793
Method:	Least Squares	**F-statistic:**	181.1
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.47e-17
Time:	03:55:30	**Log-Likelihood:**	-409.83
No. Observations:	48	**AIC:**	823.7
Df Residuals:	46	**BIC:**	827.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.433e+05	2.61e+04	-13.158	0.000	-3.96e+05	-2.91e+05
x1	176.9276	13.148	13.456	0.000	150.462	203.393

Omnibus:	2.054	**Durbin-Watson:**	0.034
Prob(Omnibus):	0.358	**Jarque-Bera (JB):**	1.969
Skew:	-0.457	**Prob(JB):**	0.374
Kurtosis:	2.616	**Cond. No.**	2.84e+05

```
fig3 = px.scatter(data_Afghanistan, x="Time", y="PopFemale", trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```



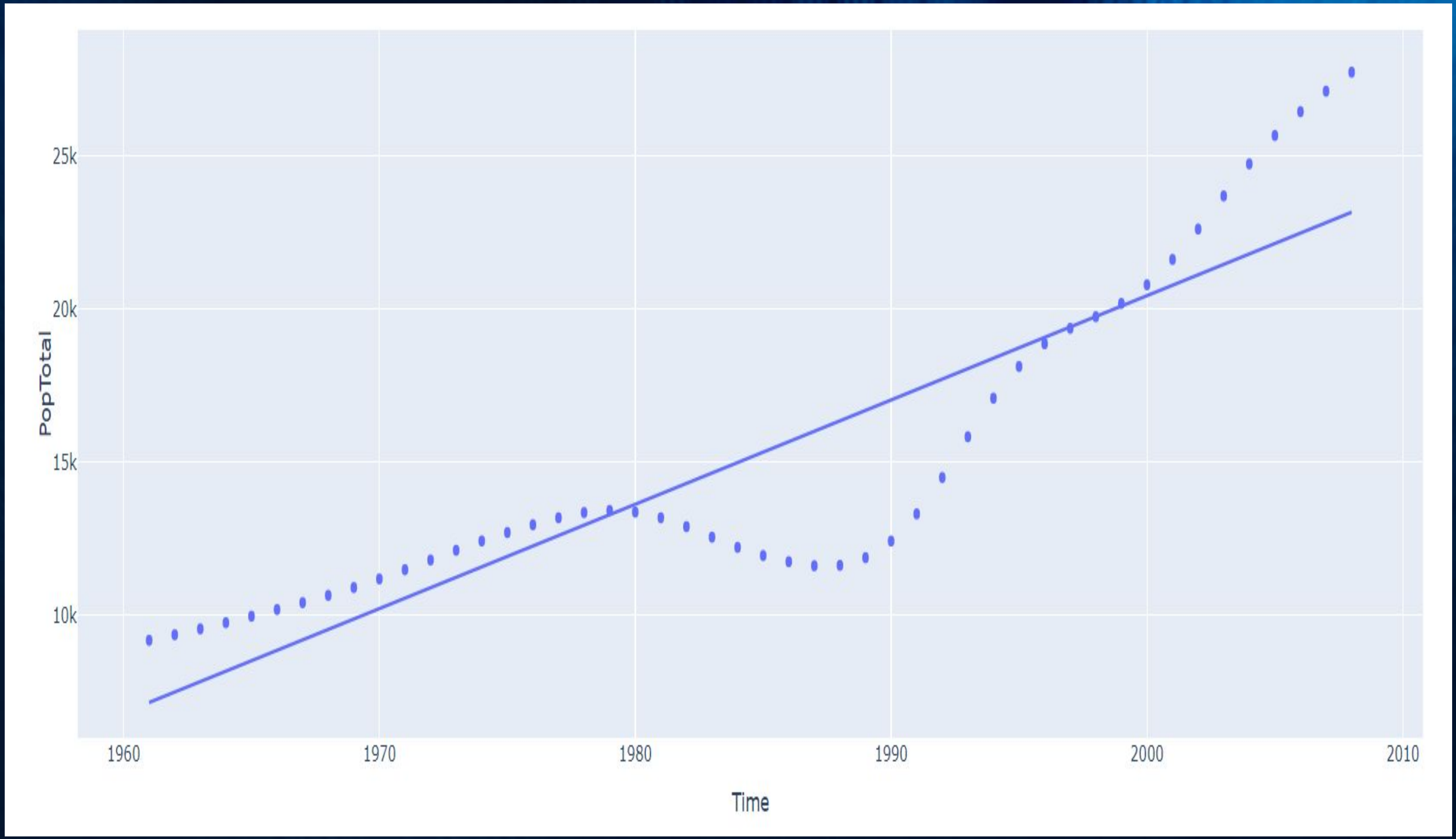
OLS Regression Results

Dep. Variable:	y	R-squared:	0.797
Model:	OLS	Adj. R-squared:	0.792
Method:	Least Squares	F-statistic:	180.1
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	1.63e-17
Time:	03:55:35	Log-Likelihood:	-406.23
No. Observations:	48	AIC:	816.5
Df Residuals:	46	BIC:	820.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.175e+05	2.42e+04	-13.114	0.000	-3.66e+05	-2.69e+05
x1	163.6878	12.198	13.419	0.000	139.134	188.242

Omnibus:	1.713	Durbin-Watson:	0.033
Prob(Omnibus):	0.425	Jarque-Bera (JB):	1.653
Skew:	-0.418	Prob(JB):	0.438
Kurtosis:	2.641	Cond. No.	2.84e+05

```python
fig4 = px.scatter(data_Afghanistan, x="Time", y="PopTotal", trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```



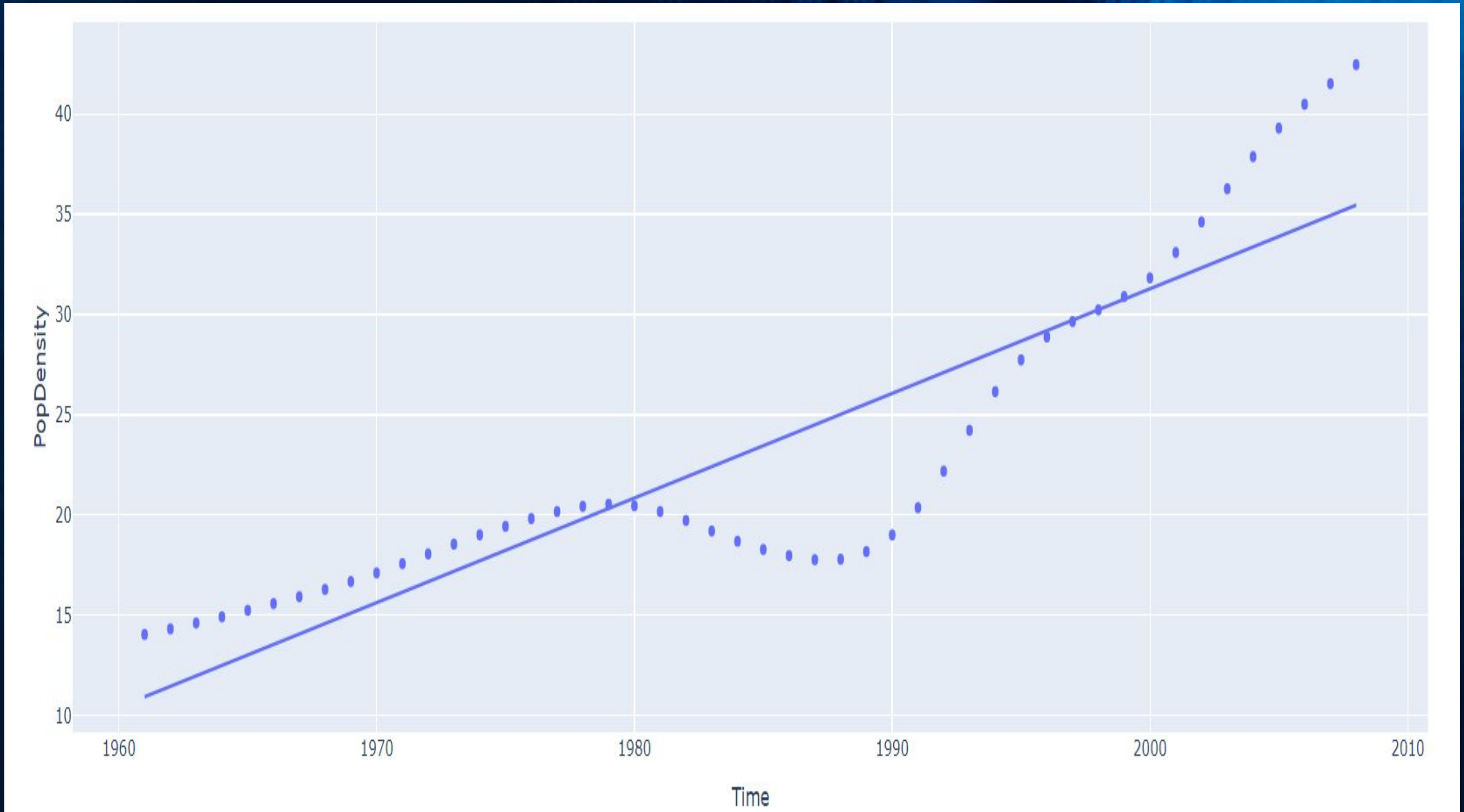
OLS Regression Results

Dep. Variable:	y	R-squared:	0.797
Model:	OLS	Adj. R-squared:	0.793
Method:	Least Squares	F-statistic:	180.8
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	1.51e-17
Time:	03:55:41	Log-Likelihood:	-441.31
No. Observations:	48	AIC:	886.6
Df Residuals:	46	BIC:	890.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6.608e+05	5.03e+04	-13.144	0.000	-7.62e+05	-5.6e+05
x1	340.6153	25.333	13.446	0.000	289.623	391.608

Omnibus:	1.894	Durbin-Watson:	0.033
Prob(Omnibus):	0.388	Jarque-Bera (JB):	1.820
Skew:	-0.440	Prob(JB):	0.402
Kurtosis:	2.629	Cond. No.	2.84e+05

```
fig5 = px.scatter(data_Afghanistan, x="Time", y="PopDensity", trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```



OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.797
Model:	OLS	**Adj. R-squared:**	0.793
Method:	Least Squares	**F-statistic:**	180.8
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.51e-17
Time:	03:55:47	**Log-Likelihood:**	-130.20
No. Observations:	48	**AIC:**	264.4
Df Residuals:	46	**BIC:**	268.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1012.1691	77.005	-13.144	0.000	-1167.172	-857.166
x1	0.5217	0.039	13.446	0.000	0.444	0.600

Omnibus:	1.894	**Durbin-Watson:**	0.033
Prob(Omnibus):	0.388	**Jarque-Bera (JB):**	1.820
Skew:	-0.440	**Prob(JB):**	0.403
Kurtosis:	2.629	**Cond. No.**	2.84e+05

Data statistics

-SELECT ONE ATTRIBUTE FROM FOUR TO MAKE A MODEL FOR PREDICTION BY MAKING OBSERVATIONS FROM GRAPHS PLOTTED DURING VISUALIZATION.

-THROUGH STATISTICS, FIND OUT THE NAMES OF COUNTRIES POSSIBLE WHOSE TOTAL POPULATION WILL BE IN THE RANGE 5000 TO 15000 IN 2011.

Following are the R squared values for 10 countries for four attributes

Male Population, Female Population, Total Population and Population density

Countries	Attribute's R^2 data			
	PopMale	PopFemale	PopTotal	PopDensity
spain	0.952	0.961	0.957	0.957
Afghanistan	0.797	0.797	0.797	0.797
India	0.993	0.994	0.993	0.993
Turkey	0.999	0.999	0.999	0.999
Zimbabwe	0.961	0.984	0.975	0.975
UAE	0.804	0.913	0.841	0.841
Belgium	0.958	0.965	0.964	0.963
Cambodia	0.847	0.905	0.879	0.876
Denmark	0.920	0.931	0.928	0.927
Egypt	0.989	0.991	0.990	0.990

Country	Ranking of Attribute
spain	PopDensity = PopTotal > PopFemale > PopMale
Afghanistan	PopDensity = PopTotal = PopFemale = PopMale
India	PopFemale > PopDensity = PopTotal = PopMale
Turkey	PopDensity = PopTotal = PopFemale = PopMale
Zimbabwe	PopFemale > PopDensity = PopTotal > PopMale
UAE	PopFemale > PopDensity = PopTotal > PopMale
Belgium	PopFemale >PopTotal > PopDensity > PopMale
Cambodia	PopFemale > PopTotal > PopDensity > PopMale
Denmark	PopFemale > PopTotal > PopDensity > PopMale
Egypt	PopFemale > PopTotal = PopDensity > PopMale

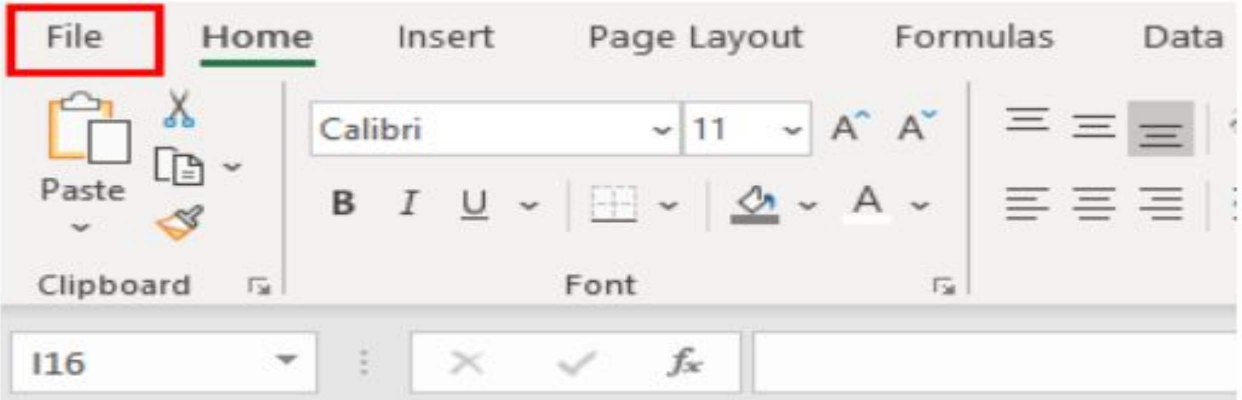
A good model is considered to have higher R-squared (R^2) according to https://statisticsbyjim.com/regression/interpret-r-squared-regression/

Based on the above ranking, We are selecting the Total Population as the attribute for making further predictions.

In this part, we will mathematically predict the names of possible countries having total populations in the range 5,000 to 15,000 in 2011 through seeing the data.

1

- Click on the ‘File’ menu.



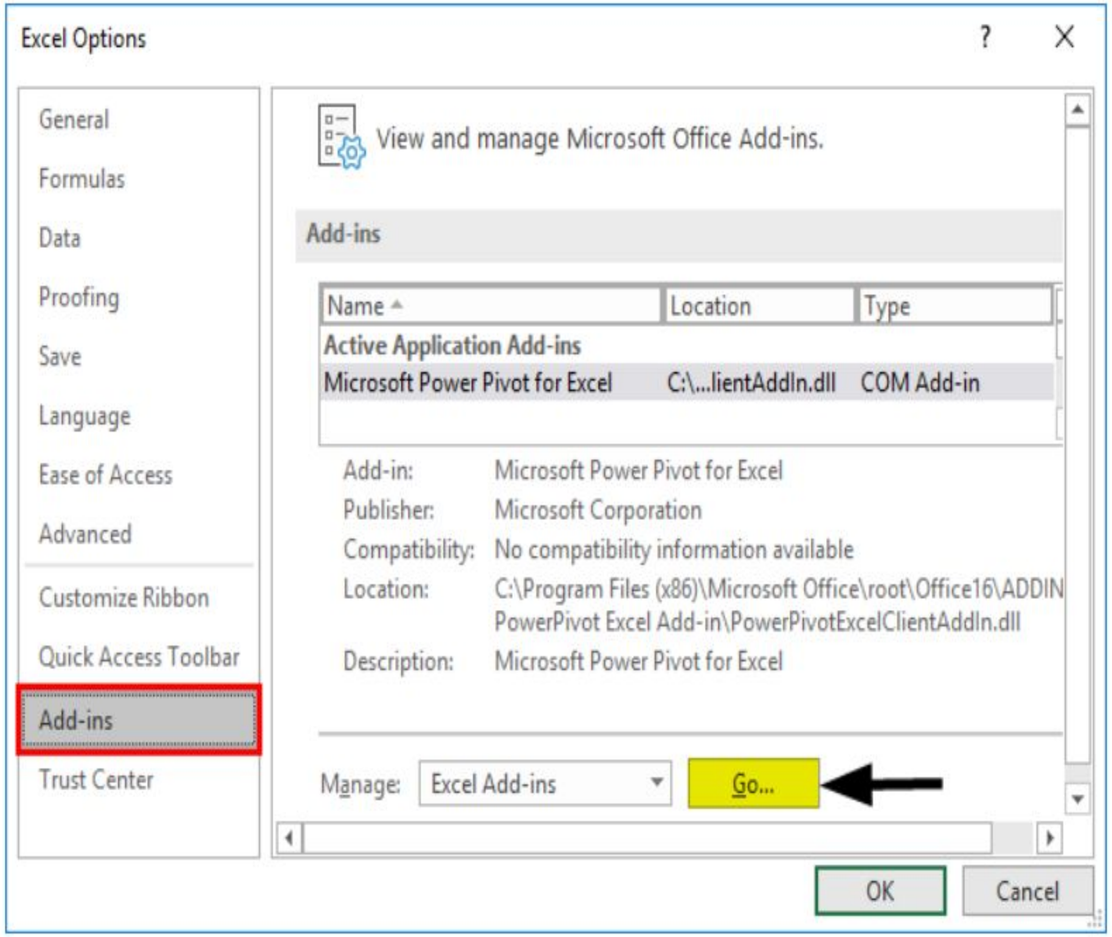
	A	B
1	Age (x)	Average Amount Spent on Medical Expenses (Per Month in Rs) (y)
2	15	100
3	20	135
4	25	135
5	37	150
6	40	250
7	45	270

2

After that, click on ‘Options’.

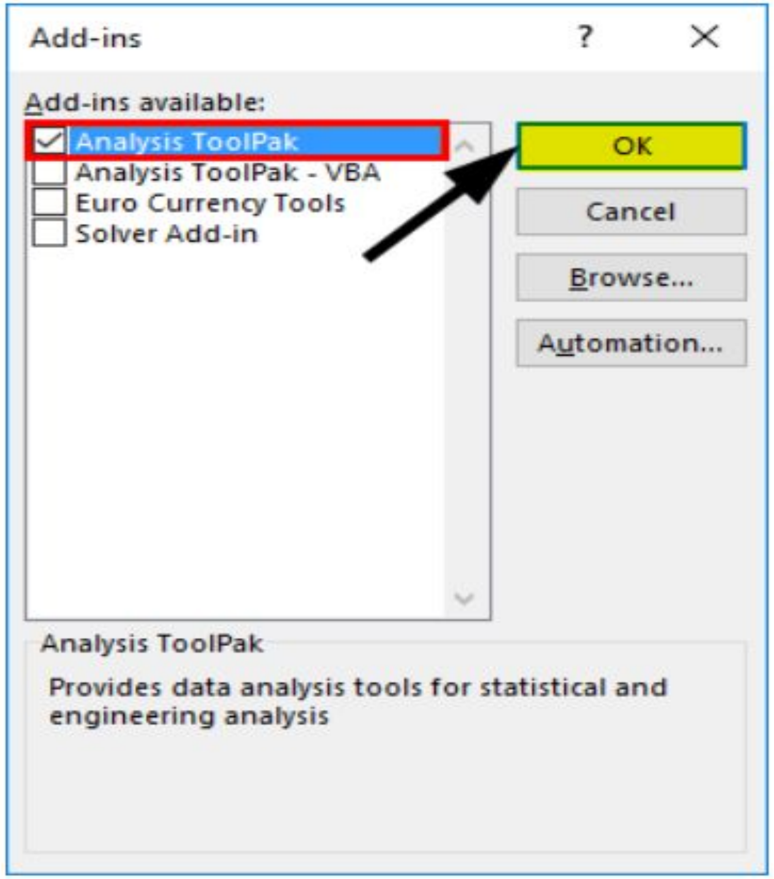
3

- Select 'Excel Add-Ins' in the 'Manage' box, and click on 'Go.'



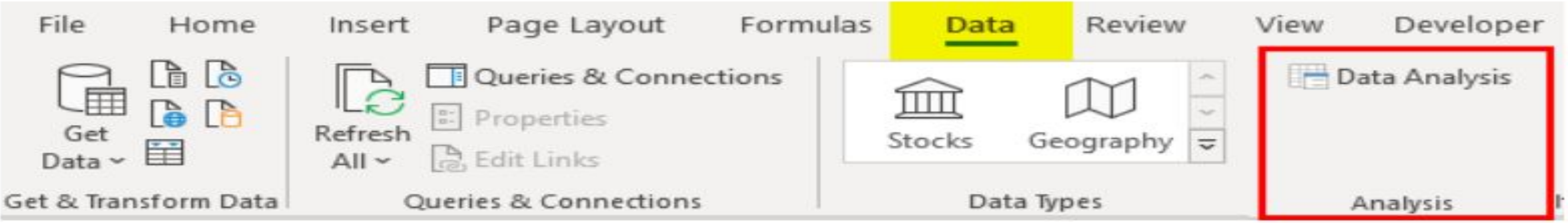
4

- Select 'Analysis ToolPak' -> 'OK'



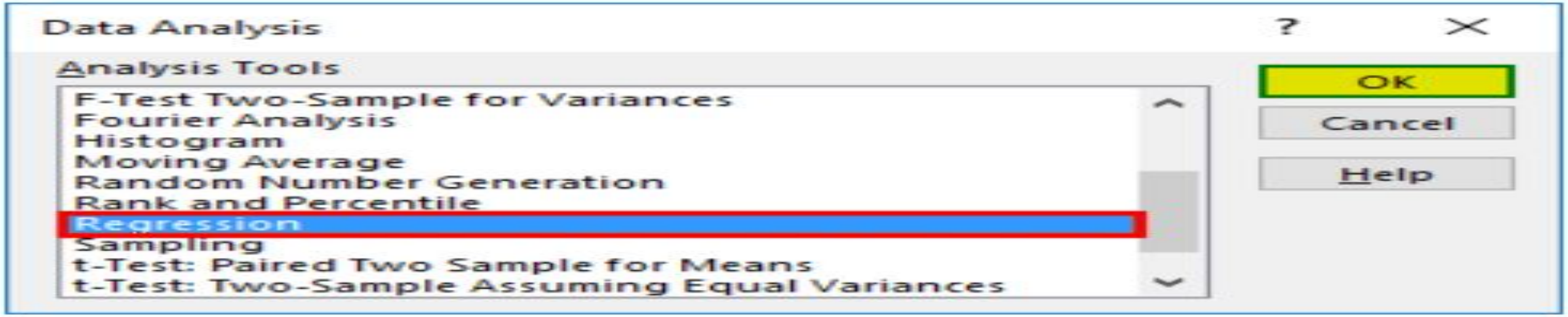
5

- Click on ‘Data Analysis’ in the ‘Data’ tab

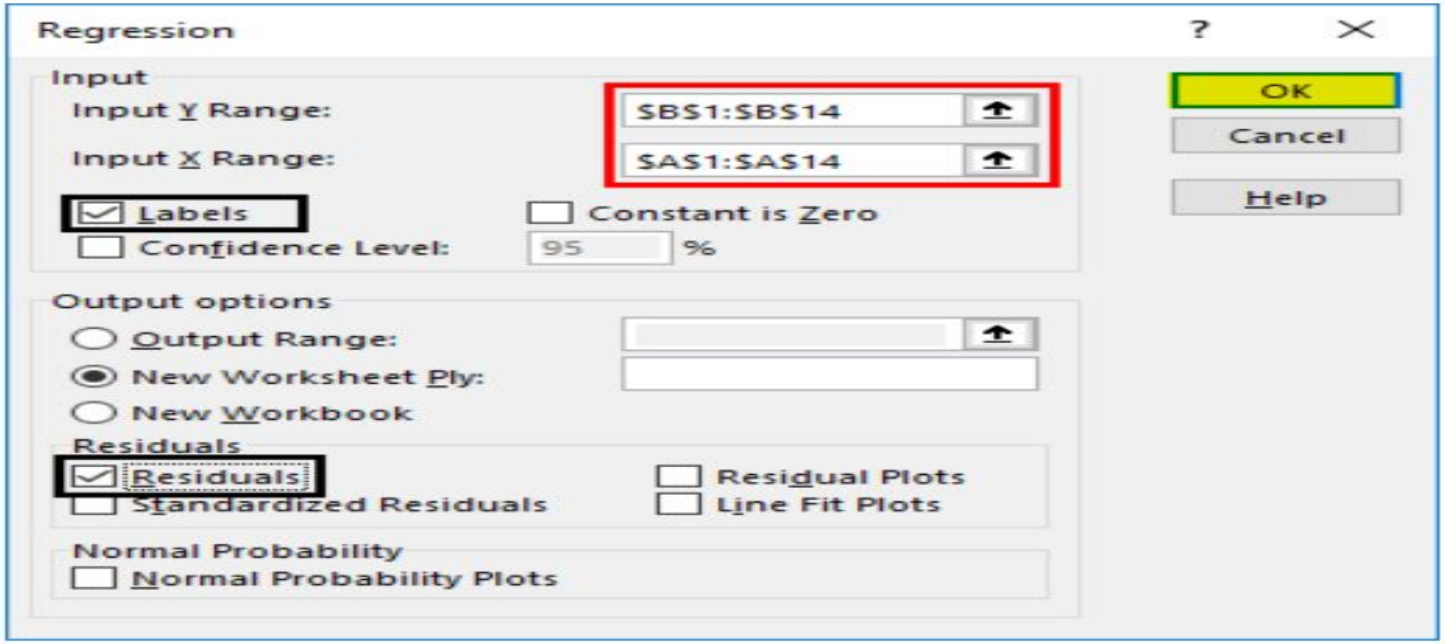


6

- Select ‘Regression’ -> ‘OK’



7



8

- **Coefficients** are the most important part used to build regression equation.

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-355.3228922	215.361787	-1.649888296	0.1271999	-829.33099	118.685206	-829.33099	118.685206
Age (x)	16.89145326	4.3904991	3.847274049	0.0027122	7.228029892	26.5548766	7.22802989	26.5548766

9

So, our regression equation would be: y= 16.891 x – 355.32.

Country	Equation of line obtained	Total Population(y) for 2011(x) in thousands
Afghanistan	y= 340.62x - 660807	24179.82
Albania	y= 32.337x - 61466	3563.707
Algeria	y = 533.47x - 1E+06	72808.17
Angola	y = 339.1x - 661978	19952.1
Argentina	y = 424.29x - 811824	41423.19
Armenia	y = 21.086x - 38911	3492.946
Aruba	y = 0.9916x - 1897.4	96.70
Austria	y = 21.908x - 35744	8312.988
Azerbaijan	y = 100.55x - 192945	9261.05
Bahamas	y = 4.5862x - 8870.1	352.7482
Bahrain	y = 16.703x - 32692	897.733
Bangladesh	y = 2098x - 4E+06	219078
Barbados	y = 1.0314x - 1791.2	282.9454
Belarus	y = 31.795x - 53620	10319.745
Belgium	y = 26.011x - 41669	10639.121
Belize	y = 4.2501x - 8257.6	289.3511
Benin	y = 130.7x - 254647	8190.7
Bhutan	y = 9.7622x - 18925	706.7842
Botswana	y = 32.05x - 62490	1962.55
Brazil	y = 2603.7x - 5E+06	236040.7
Bulgaria	y = -12.036x + 32314	8109.604

Burundi	y = 105.58x - 204612	7709.38
Cambodia	y = 168.34x - 325151	13380.74
Cameroon	y = 296.85x - 578354	18611.35
Canada	y = 312.65x - 594640	34099.15
Caribbean	y = 424.04x - 810065	42679.44
Chad	y = 160.96x - 313630	10060.56
Chile	y = 181.88x - 348551	17209.68
China	y = 15339x - 3E+07	846729
Colombia	y = 599.79x - 1E+06	206177.69
Comoros	y = 10.113x - 19693	644.243
Congo	y = 60.395x - 117676	3778.345
Costa Rica	y = 68.207x - 132541	4623.277
Croatia	y = 2.7508x - 945.99	4585.8688
Cuba	y = 82.406x - 153630	12088.466
Cyprus	y = 10.697x - 20471	1040.667
Czechia	y = 13.853x - 17327	10531.383
Denmark	y = 15.154x - 24961	5513.694
Djibouti	y = 17.463x - 34216	902.093
Dominica	y = 0.0798x - 87.815	72.6628
Ecuador	y=222.58x-432463	15145.38
Egypt	y=1238x-2E+06	489618
Eritrea	y=42.957x-83313	3073.527
Ethiopia	y=1509.4x-3E+06	35403.4

Fiji	y=8.2547x-15724	876.2017
Finland	y=18.511x-31837	5388.621
France	y=305.21x-550768	63009.31
Gabon	y=27.125x-52910	1638.375
Gambia	y=33.662x-65910	1784.282
Georgia	y=-11. 569x+27641	4375.741
Germany	y=112.43x-144214	81882.73
Ghana	y=402.55x-784920	20108.05
Greece	y=51.511x-92501	11087.621
Greenland	y=0.3318x-608.59	58.6598
Guadeloupe	y=2.2892x-4188.8	414.7812
Guatemala	y=225.79x-439453	14610.69
Guinea	y=154.95x-301408	10196.45
Haiti	y=130.91x-253298	9962.01
Honduras	y=137.5x-268289	8223.5
Hungary	y=-10. 402x+30954	10035.578
Iceland	y=2.7541x-5222.9	316
India	y=16573x-3E+07	3328303
Indonesia	y=3162.2x-6E+06	359184
Iraq	y=524.55x-1E+06	54870
Ireland	y=35.785x-67562	4401
Israel	y=110.72x-215409	7249
Italy	y=149.16x-240305	59655
Japan	y=567.98x-1E+06	142207

Kazakhstan	y=94.938x-173900	17020
Kenya	y=676x-1321864	37572
Kuwait	y=58.216x-113986	3086
Kyrgyzstan	y=65.727x-126541	5635
Latin America and the Caribbean	y=7721x-1.5E+07	526931
Lebanon	y=57x-111196	3431
Liberia	y=85x-88900	82035
Libya	y=104x-204013	5131
Liechtenstein	y=0.4x-752	52
Lithuania	y=12x-18860	5272
Luxembourg	y=4x-5913	2128
Malawi	y=217x-424149	12238
Malaysia	y=408x-793790	26698
Maldives	y=5.5x-10740	320
Mali	y=174x-337280	12632
Malta	y=2.3x-4131	494
Martinique	y=2.2x-3948	476
Mauritania	y=51x-98359	4202
Mauritius	y=12x-22267	1865
Mayotte	y=4x-7343	701
Mexico	y=1562x-3025602	115580
Micronesia	y=7.5x-14476	606
Mongolia	y=37x-71491	2916
Morocco	y=417x-806937	31650

Mozambique	y=293x-569992	19231
Myanmar	y=628x-1211127	51781
Namibia	y=31x-60780	1561
Nepal	y=374x-724896	27218
Netherlands	y=100x-185546	15554
New Caledonia	y=3.6x-6873	366
New Zealand	y=36x-68126	4270
Nicaragua	y=85x-166543	4392
Niger	y=232x-454112	12440
Nigeria	y=2192x-4264229	143883
North Macedonia	y=11x-20450	1671
Norway	y=22x-39300	4942
Oceania	y=394x-757126	35208
Oman	y=50x-98405	2145
Pakistan	y=2732x-5324632	169420
Panama	y=50x-96949	3604
Papua New Guinea	y=100x-194306	6794
Paraguay	y=92x-179125	5887
Peru	y=413x-800571	29972
Philippines	y=1383x-2688873	92340
Poland	y=191x-344960	39141
Polynesia	y=7x-13708	369
Portugal	y=43x-76218	10255
Puerto Rico	y=30x-58107	2223
Qatar	y=20.5x-40393	832
Romania	y=52x-82170	22558
Rwanda	y=134x-261035	8439
Saint Vincent and the Grenadines	y=0.57x-1039	107

Samoa	y=1.2x-2243	170
Sao Tome and Principe	y=2.31x-4483	162
Saudi Arabia	y=493x-966407	25016
Senegal	y=186x-362322	11724
Serbia	y=41x-72940	9511
Seychelles	y=0.97x-1872	78
Sierra Leone	y=73x-141217	5586
Singapore	y=62x-121635	3047
Slovakia	y=26x-48510	3776
Slovenia	y=10x-18236	1874
South Africa	y=736x-1429562	50534
Spain	y=268x-495462	43486
Sri Lanka	y=212x-405749	20583
State of Palestine	y=62x-122825	1857
Sudan	y=561x-1095361	32810
Suriname	y=4x-7741	303
Sweden	y=31x-54012	8329
Switzerland	y=38x-69151	7267
Tajikistan	y=110x-214416	6794
Thailand	y=847x-1631388	71929
Togo	y=95x-186904	4141
Tonga	y=0.58x-1077	89
Tunisia	y=147x-285339	10278
Turkey	y=932x-1800931	73321
Turkmenistan	y=74x-144632	4182

Uganda	y=488x-954321	27047
Ukraine	y=58x-67035	49603
United Arab Emirates	y=111x-219954	3267
United Kingdom	y=139x-220826	58703
United States of America	y=2382x-4485136	305066
Uruguay	y=16x-30052	2124
Uzbekistan	y=416x-808636	27940
Vanuatu	y=3.36x-6540	216
Viet Nam	y=1187x-2295099	91958
Yemen	y=368x-720724	19324
Zambia	y=207x-405542	10735
Zimbabwe	y=207x-402744	13533

The Countries having population between 5000 thousands to 15000 thousands in 2011 are Algeria, Austria, Azerbaijan, Belarus, Belgium, Benin, Bulgaria, Burundi, Cambodia, Chad, Cuba, Czechia, Denmark, Finland, Guatemala, Guinea, Haiti, Honduras, Hungary, Israel, Kyrgyzstan, Libya, Lithuania, Malawi, Mali, Niger, Papua New Guinea, Paraguay, Portugal, Rwanda, Senegal, Serbia, Sierra Leone, Sweden, Switzerland, Tajikistan, Tunisia, Zambia and Zimbabwe.

Hypothetical Statement

FROM VISUALIZATION, WE HAVE SEEN TOTAL POPULATION AS AN ATTRIBUTE WHICH WILL BE HELPFUL IN MAKING PREDICTIONS OUT OF FOUR ATTRIBUTES.

THROUGH STATISTICS, WE HAVE PREDICTED THE COUNTRIES HAVING THEIR TOTAL POPULATION IN THE RANGE 5000 THOUSANDS TO 15000 THOUSANDS IN 2011.

STATEMENT FOR HYPOTHESIS:

"NOW WE WILL CREATE A MODEL IN WHICH WE WILL USE TOTAL POPULATION AS AN ATTRIBUTE AND TRY TO PREDICT THE COUNTRIES HAVING THEIR TOTAL POPULATION IN THE RANGE 5000 THOUSANDS TO 15000 THOUSANDS IN 2011."

Prediction Task

Now we will create Model using all 4 classifiers stated in Data Collection part and compare their accuracy for usage of all four attributes together.

1 Decision Tree Classification

```
#Importing all necessary modules
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix

#Reading data file
training_set = pd.read_csv("/content/TotalPopulation_all.csv")
test_set = pd.read_csv("/content/TotalPopulation_all.csv")

# 70% training and 30% test
X_train, X_test, Y_train, Y_test = train_test_split(training_set.iloc[:,0:5].values,training_set.iloc[:,5].values, test_size=0.3, random_state=1)
```

```python
#Making prediction
classifier = DecisionTreeClassifier()
classifier.fit(X_train,Y_train)
Y_pred = classifier.predict(X_test)

#output
print("X_test value:\n")
print(X_test)
print("Y_test value:\n")
print(Y_test)
print("Y_pred value:\n")
print(Y_pred)

#Computing accuracy
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy For The Given Dataset : ", accuracy)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```

```
X_test value:

[[2.01000000e+03 9.24269890e+04 8.69976540e+04 1.79424643e+05
  2.32753000e+02]
 [1.98200000e+03 1.05047600e+03 1.02327200e+03 2.07374800e+03
  2.78950000e+01]
 [1.96300000e+03 2.31229110e+04 2.44379140e+04 4.75608250e+04
  8.68600000e+01]
 ...
 [1.96800000e+03 4.22714500e+03 4.09462800e+03 8.32177300e+03
  3.65730000e+01]
 [1.98000000e+03 1.60679500e+03 1.57483200e+03 3.18162700e+03
  8.00800000e+00]
 [1.99300000e+03 1.94613000e+02 2.02199000e+02 3.96812000e+02
  1.53209000e+02]]

Y_test value:

[586 591 250 ... 288 600 442]

Y_pred value:

[586 430 250 ... 800 600 442]

Accuracy For The Given Dataset :  0.8217228464419476
Accuracy: 0.8217228464419476
```

2 Random Forest Classification

2

```
#Importing all necessary modules
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix

#Reading data file
training_set = pd.read_csv("/content/TotalPopulation_total.csv")
test_set = pd.read_csv("/content/TotalPopulation_total.csv")

# 70% training and 30% test
X_train, X_test, Y_train, Y_test= train_test_split(training_set.iloc[:,0:5].values,training_set.iloc[:,5].values, test_size=0.3, random_state=1)

#Making prediction
classifier = RandomForestClassifier()
classifier.fit(X_train,Y_train)
Y_pred = classifier.predict(X_test)

#output
print("X_test value:\n")
print(X_test)
print("Y_test value:\n")
print(Y_test)
print("Y_pred value:\n")
print(Y_pred)

#Computing accuracy
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy For The Given Dataset : ", accuracy)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```

3 Support Vector Machines

```
#Importing all necessary modules
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix

#Reading data file
training_set = pd.read_csv("/content/TotalPopulation_total.csv")
test_set = pd.read_csv("/content/TotalPopulation_total.csv")

# 70% training and 30% test
X_train, X_test, Y_train, Y_test= train_test_split(training_set.iloc[:,0:5].values,training_set.iloc[:,5].values, test_size=0.3, random_state=1)

#Making prediction
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(X_train,Y_train)
Y_pred = classifier.predict(X_test)

#output
print("X_test value:\n")
print(X_test)
print("Y_test value:\n")
print(Y_test)
print("Y_pred value:\n")
print(Y_pred)

#Computing accuracy
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy For The Given Dataset : ", accuracy)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```

4 Logistic Regression

```
#Importing all necessary modules
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import Perceptron
#Reading data file
training_set = pd.read_csv("/content/TotalPopulation_total.csv")
test_set = pd.read_csv("/content/TotalPopulation_total.csv")

# 70% training and 30% test
X_train, X_test, Y_train, Y_test= train_test_split(training_set.iloc[:,0:5].values,training_set.iloc[:,5].values, test_size=0.3, random_state=1)

#Making prediction
classifier = Perceptron(random_state=0)
classifier.fit(X_train,Y_train)
Y_pred = classifier.predict(X_test)

#output
print("X_test value:\n")
print(X_test)
print("Y_test value:\n")
print(Y_test)
print("Y_pred value:\n")
print(Y_pred)

#Computing accuracy
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy For The Given Dataset : ", accuracy)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```

Classifier	Accuracy for all four attributes used together
Logistic Regression	1.6%
Support Vector Machines	21%
Decision Tree Classification	82%
Random Forest Classification	78%

We will be using Decision Tree Classification as it has highest accuracy.

Using Decision Tree Classification for model making and using only Total Population as attribute along with year

```
#Importing all necessary modules
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
from sklearn.tree  import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix

#Reading data file
training_set = pd.read_csv("/content/TotalPopulation_total.csv")
test_set = pd.read_csv("/content/TotalPopulation_total.csv")

# 70% training and 30% test
X_train, X_test, Y_train, Y_test = train_test_split(training_set.iloc[:,0:2].values,training_set.iloc[:,2].values, test_size=0.3, random_state=1)

#Making prediction
classifier = DecisionTreeClassifier()
classifier.fit(X_train,Y_train)
Y_pred = classifier.predict(X_test)
```

```python
#output
print("X_test value:\n")
print(X_test)
print(" ")
print("Y_test value:\n")
print(Y_test)
print(" ")
print("Y_pred value:\n")
print(Y_pred)

#Computing accuracy
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy For The Given Dataset : ", accuracy)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```

```
X_test value:

[[   2010.      179424.643]
 [   1982.        2073.748]
 [   1963.       47560.825]
 ...
 [   2012.        3045.561]
 [   2014.       10549.007]
 [   1964.         295.509]]

Y_test value:

[586 591 250 ... 440 332 312]

Y_pred value:

[586 430 566 ... 440 646 740]

Accuracy For The Given Dataset :  0.24837074583635046
Accuracy: 0.24837074583635046
```

Attributes taken in Decision Tree Classification	Accuracy Obtained
All Four	82%
Total Population	24%
Male Population	26%
Female Population	25%
Population Density	22%

The accuracy obtained by using all four attributes in our model is greater than obtained by using "Total Population" as an attribute.
Thus Experimentally, Our model proves our Hypothetical statement which is based on using "Total Population" as an attribute to make our model Wrong.

Hypothetical Statement is Wrong.

print(“Thank You!!”)

```python
data_India = data.loc[data.Location == 'India']
data_India.head()
fig2 = px.scatter(data_India, x="Time", y="PopMale",
trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

```python
fig5 = px.scatter(data_India, x="Time",
y="PopFemale", trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	6856.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.04e-51
Time:	03:55:53	**Log-Likelihood:**	-508.54
No. Observations:	48	**AIC:**	1021.
Df Residuals:	46	**BIC:**	1025.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.648e+07	2.04e+05	-80.777	0.000	-1.69e+07	-1.61e+07
x1	8511.6136	102.796	82.801	0.000	8304.696	8718.532

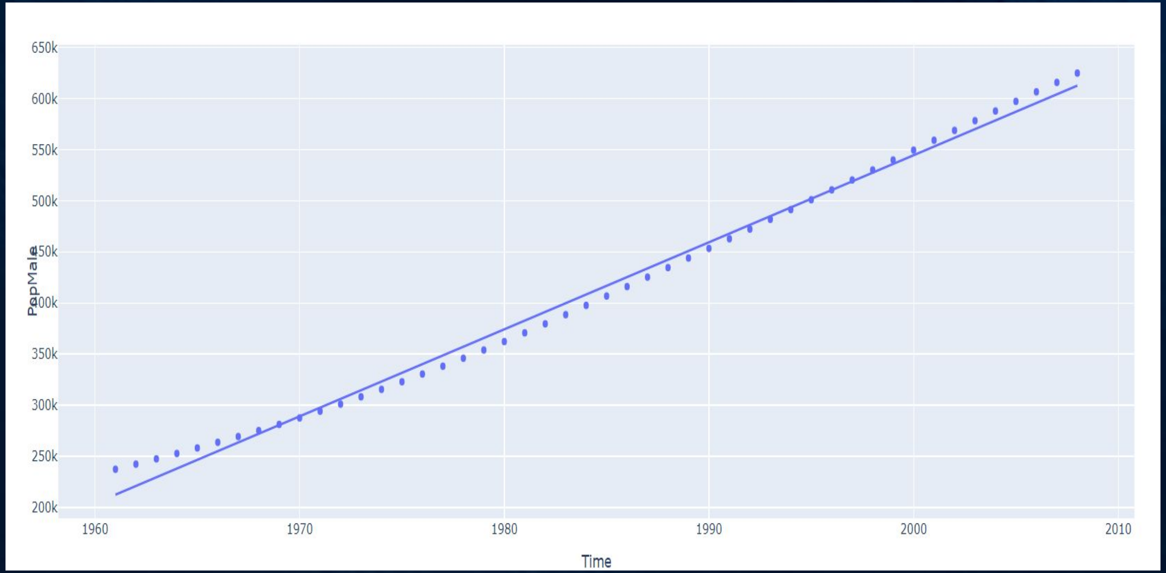
Omnibus:	3.745	**Durbin-Watson:**	0.026
Prob(Omnibus):	0.154	**Jarque-Bera (JB):**	3.544
Skew:	0.614	**Prob(JB):**	0.170
Kurtosis:	2.486	**Cond. No.**	2.84e+05

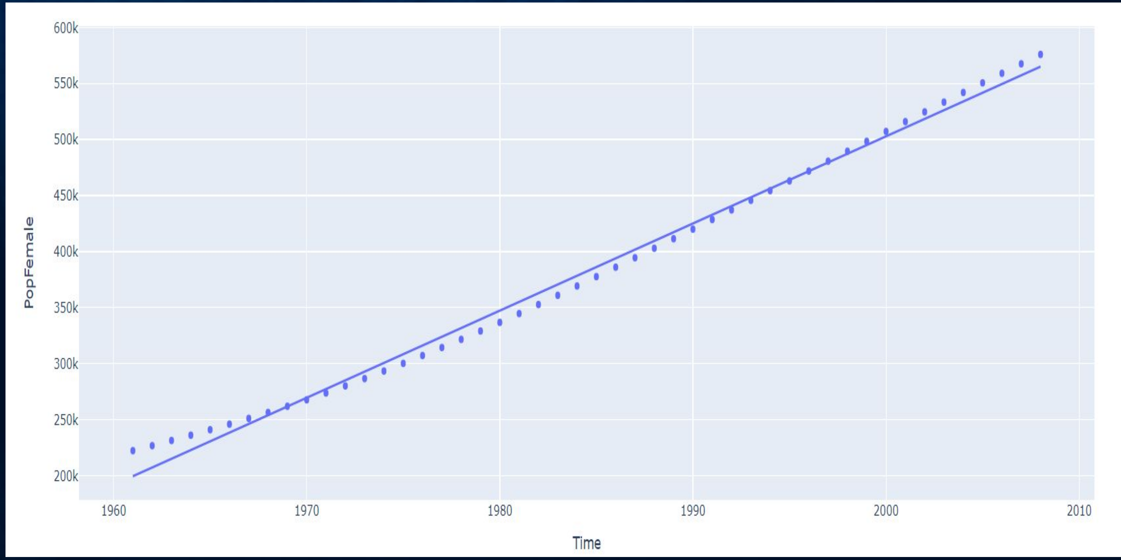
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.994
Model:	OLS	**Adj. R-squared:**	0.994
Method:	Least Squares	**F-statistic:**	7194.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	3.46e-52
Time:	03:55:59	**Log-Likelihood:**	-503.05
No. Observations:	48	**AIC:**	1010.
Df Residuals:	46	**BIC:**	1014.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.505e+07	1.82e+05	-82.717	0.000	-1.54e+07	-1.47e+07
x1	7777.6148	91.696	84.819	0.000	7593.040	7962.190

Omnibus:	4.059	**Durbin-Watson:**	0.027
Prob(Omnibus):	0.131	**Jarque-Bera (JB):**	3.903
Skew:	0.673	**Prob(JB):**	0.142
Kurtosis:	2.627	**Cond. No.**	2.84e+05





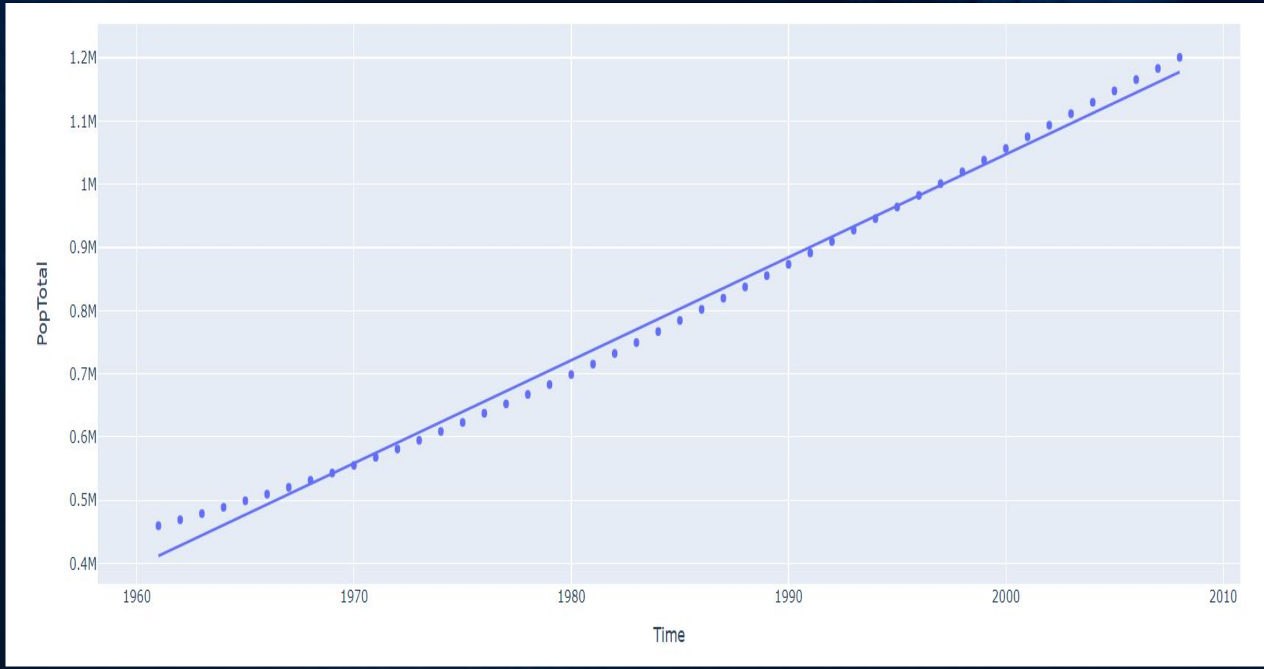
```python
fig5 = px.scatter(data_India, x="Time", y="PopTotal", trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	7016.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	6.12e-52
Time:	03:56:06	**Log-Likelihood:**	-539.14
No. Observations:	48	**AIC:**	1082.
Df Residuals:	46	**BIC:**	1086.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.153e+07	3.86e+05	-81.703	0.000	-3.23e+07	-3.08e+07
x1	1.629e+04	194.466	83.764	0.000	1.59e+04	1.67e+04

Omnibus:	3.862	**Durbin-Watson:**	0.026
Prob(Omnibus):	0.145	**Jarque-Bera (JB):**	3.701
Skew:	0.642	**Prob(JB):**	0.157
Kurtosis:	2.551	**Cond. No.**	2.84e+05



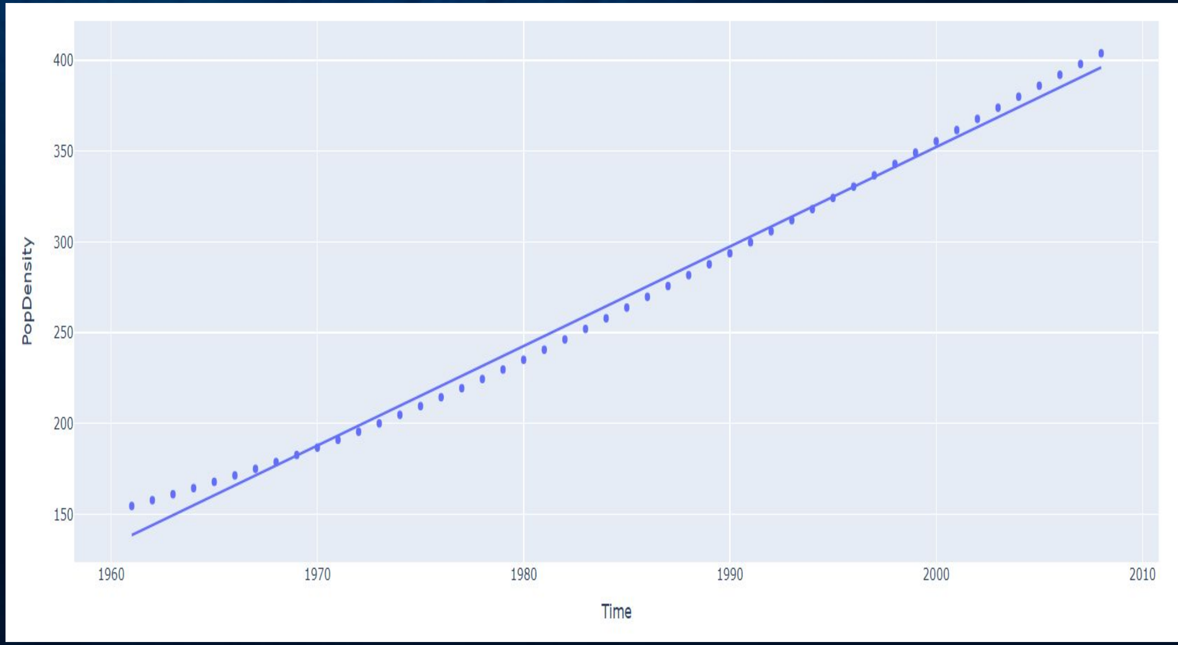

```python
fig5 = px.scatter(data_India, x="Time", y="PopDensity", trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	7016.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	6.13e-52
Time:	03:56:11	**Log-Likelihood:**	-155.26
No. Observations:	48	**AIC:**	314.5
Df Residuals:	46	**BIC:**	318.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.061e+04	129.804	-81.701	0.000	-1.09e+04	-1.03e+04
x1	5.4787	0.065	83.763	0.000	5.347	5.610

Omnibus:	3.862	**Durbin-Watson:**	0.026
Prob(Omnibus):	0.145	**Jarque-Bera (JB):**	3.701
Skew:	0.642	**Prob(JB):**	0.157
Kurtosis:	2.551	**Cond. No.**	2.84e+05



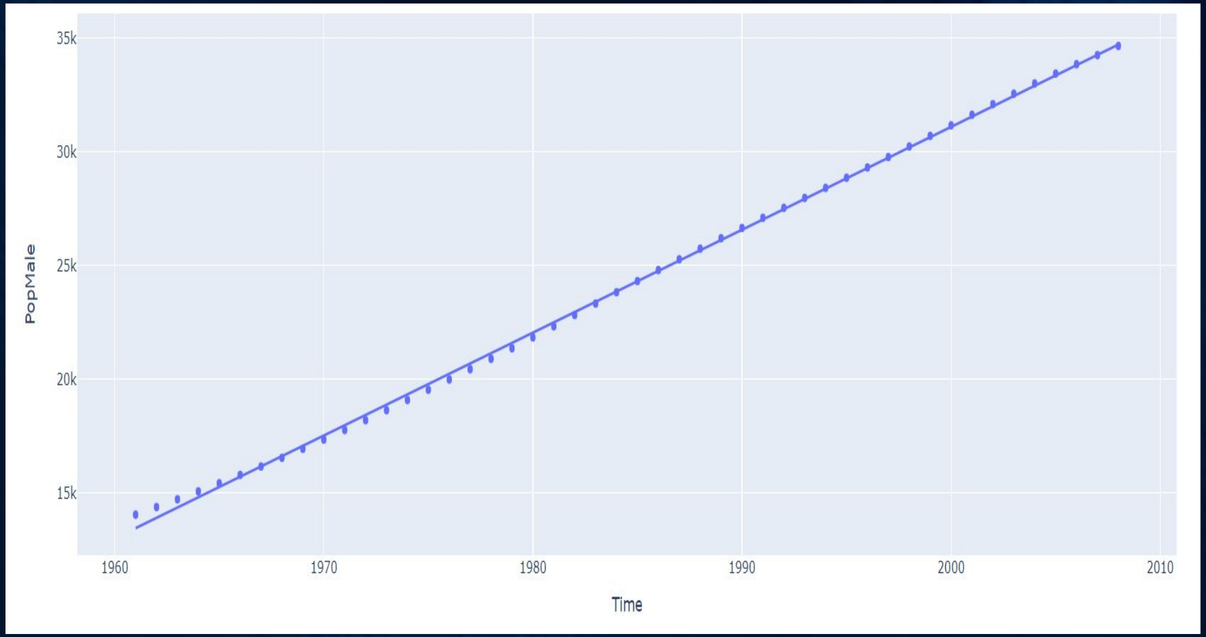
```python
data_Turkey = data.loc[data.Location == 'Turkey']
data_Turkey.head()
fig2 = px.scatter(data_Turkey, x="Time", y="PopMale",
trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	5.511e+04
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.80e-72
Time:	14:33:38	**Log-Likelihood:**	-317.66
No. Observations:	48	**AIC:**	639.3
Df Residuals:	46	**BIC:**	643.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-8.738e+05	3824.883	-228.465	0.000	-8.82e+05	-8.66e+05
x1	452.4624	1.927	234.761	0.000	448.583	456.342

Omnibus:	10.075	**Durbin-Watson:**	0.059
Prob(Omnibus):	0.006	**Jarque-Bera (JB):**	10.154
Skew:	0.847	**Prob(JB):**	0.00624
Kurtosis:	4.486	**Cond. No.**	2.84e+05



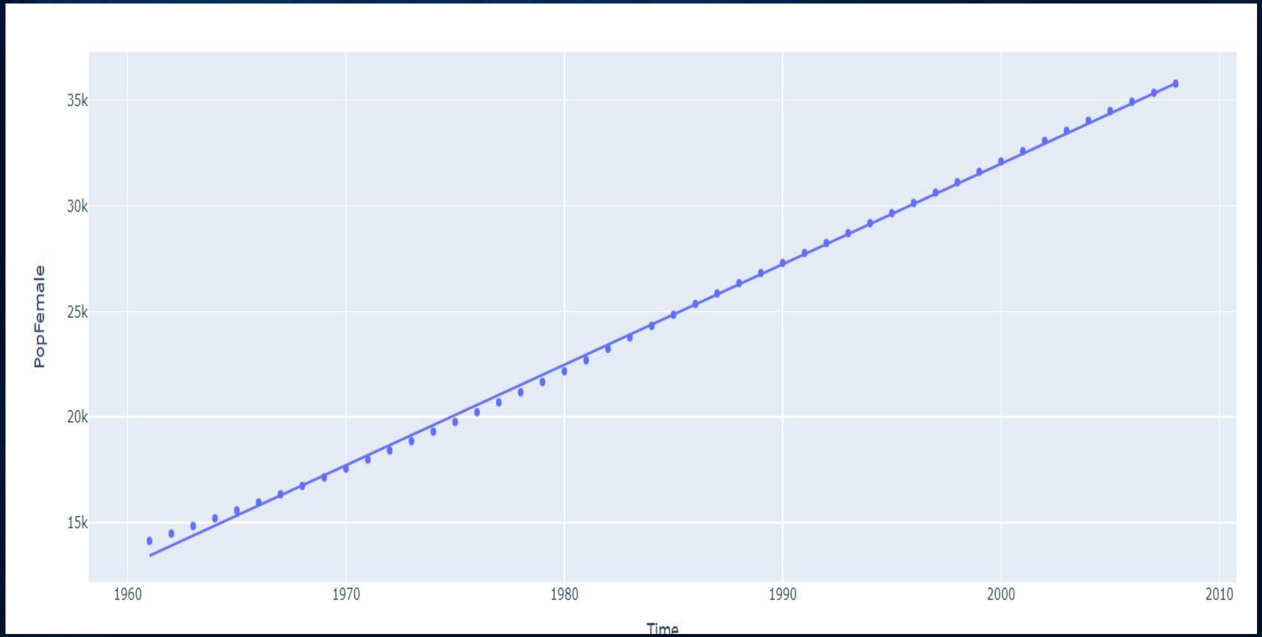

```python
fig3 = px.scatter(data_Turkey, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	3.840e+04
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	7.23e-69
Time:	14:33:57	**Log-Likelihood:**	-328.76
No. Observations:	48	**AIC:**	661.5
Df Residuals:	46	**BIC:**	665.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-9.199e+05	4820.060	-190.859	0.000	-9.3e+05	-9.1e+05
x1	475.9706	2.429	195.970	0.000	471.082	480.860

Omnibus:	5.810	**Durbin-Watson:**	0.048
Prob(Omnibus):	0.055	**Jarque-Bera (JB):**	4.725
Skew:	0.602	**Prob(JB):**	0.0942
Kurtosis:	3.956	**Cond. No.**	2.84e+05



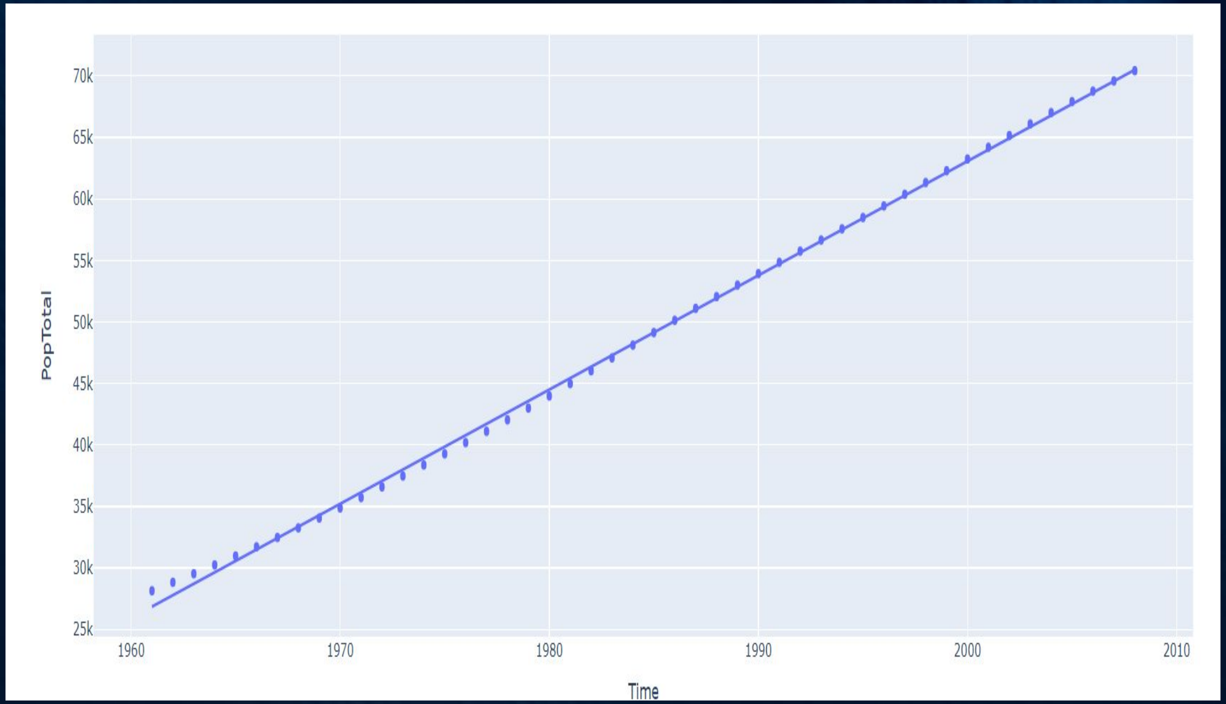
```python
fig4 = px.scatter(data_Turkey, x="Time",
y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	4.569e+04
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.34e-70
Time:	14:34:03	**Log-Likelihood:**	-356.67
No. Observations:	48	**AIC:**	717.3
Df Residuals:	46	**BIC:**	721.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.794e+06	8620.329	-208.089	0.000	-1.81e+06	-1.78e+06
x1	928.4331	4.344	213.741	0.000	919.690	937.177

Omnibus:	7.776	**Durbin-Watson:**	0.053
Prob(Omnibus):	0.020	**Jarque-Bera (JB):**	7.037
Skew:	0.720	**Prob(JB):**	0.0296
Kurtosis:	4.201	**Cond. No.**	2.84e+05



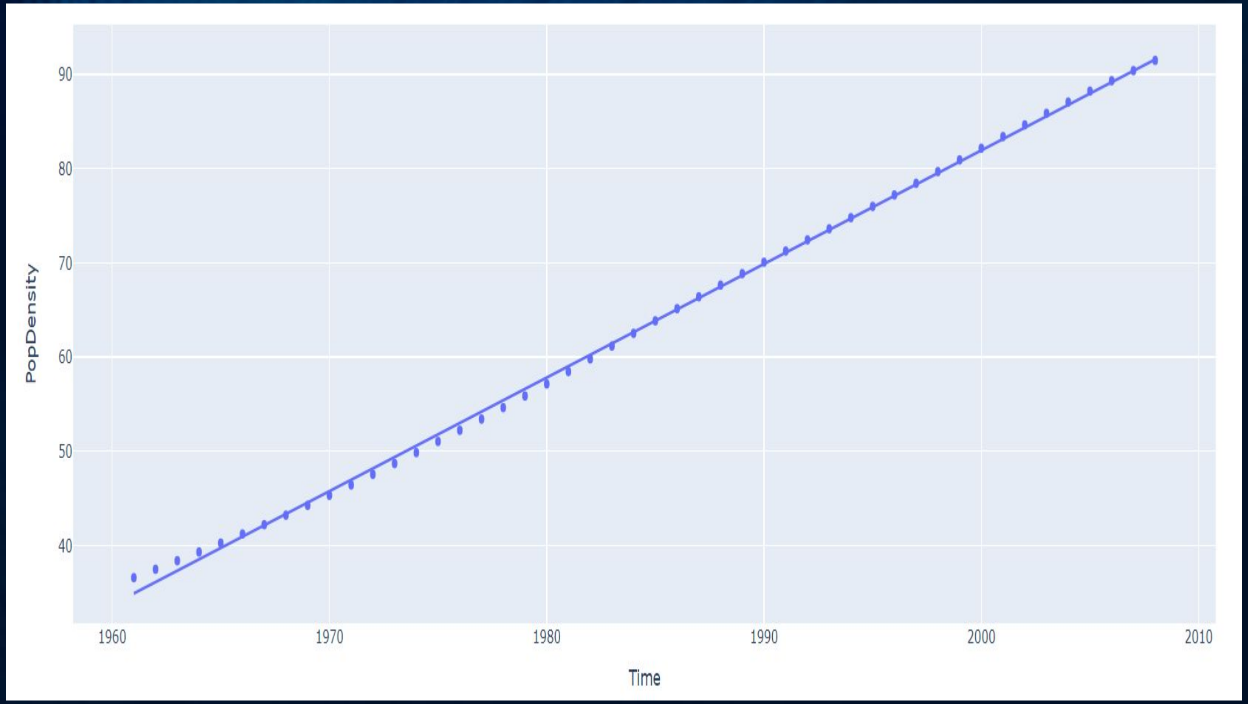

```python
fig5 = px.scatter(data_Turkey, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	4.569e+04
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.34e-70
Time:	14:34:09	**Log-Likelihood:**	-37.660
No. Observations:	48	**AIC:**	79.32
Df Residuals:	46	**BIC:**	83.06
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2330.7372	11.200	-208.096	0.000	-2353.282	-2308.192
x1	1.2063	0.006	213.748	0.000	1.195	1.218

Omnibus:	7.778	**Durbin-Watson:**	0.053
Prob(Omnibus):	0.020	**Jarque-Bera (JB):**	7.040
Skew:	0.720	**Prob(JB):**	0.0296
Kurtosis:	4.202	**Cond. No.**	2.84e+05



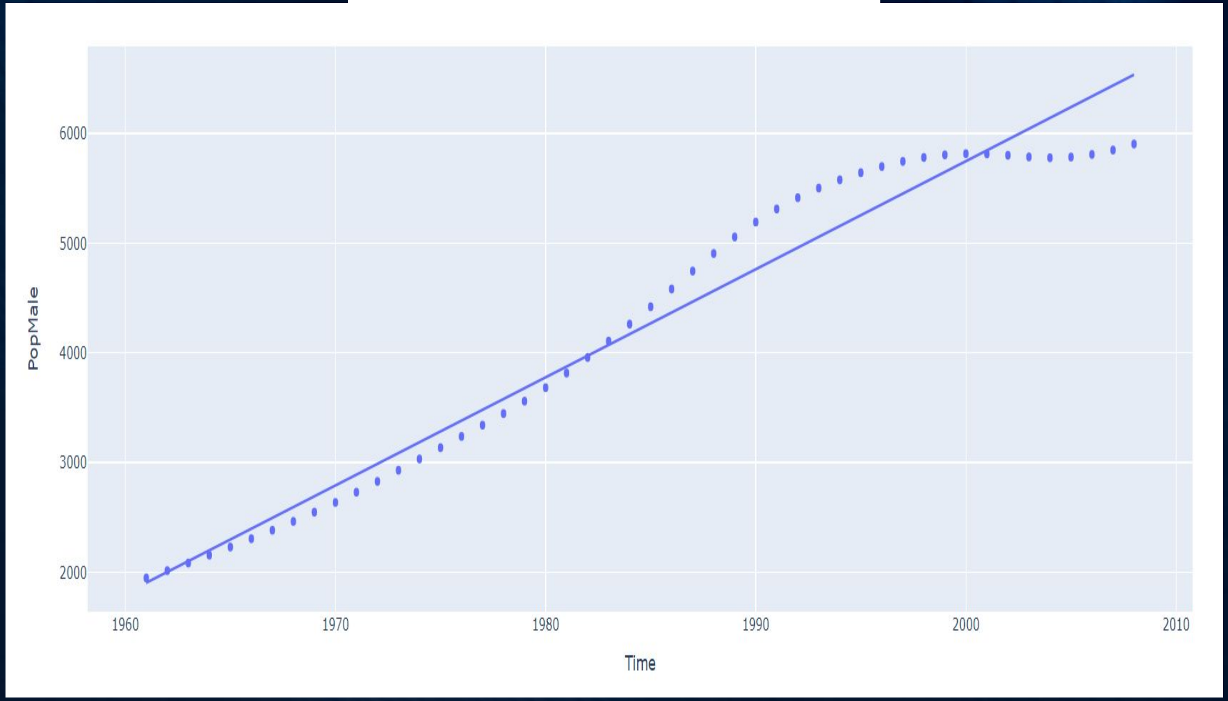
```python
data_Zimbabwe = data.loc[data.Location == 'Zimbabwe']
fig2 = px.scatter(data_Zimbabwe, x="Time", y="PopMale",
trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.961
Model:	OLS	Adj. R-squared:	0.961
Method:	Least Squares	F-statistic:	1148.
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	3.57e-34
Time:	04:00:46	Log-Likelihood:	-337.41
No. Observations:	48	AIC:	678.8
Df Residuals:	46	BIC:	682.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.913e+05	5771.491	-33.144	0.000	-2.03e+05	-1.8e+05
x1	98.5173	2.908	33.876	0.000	92.663	104.371

Omnibus:	0.150	Durbin-Watson:	0.035
Prob(Omnibus):	0.928	Jarque-Bera (JB):	0.321
Skew:	-0.103	Prob(JB):	0.852
Kurtosis:	2.657	Cond. No.	2.84e+05



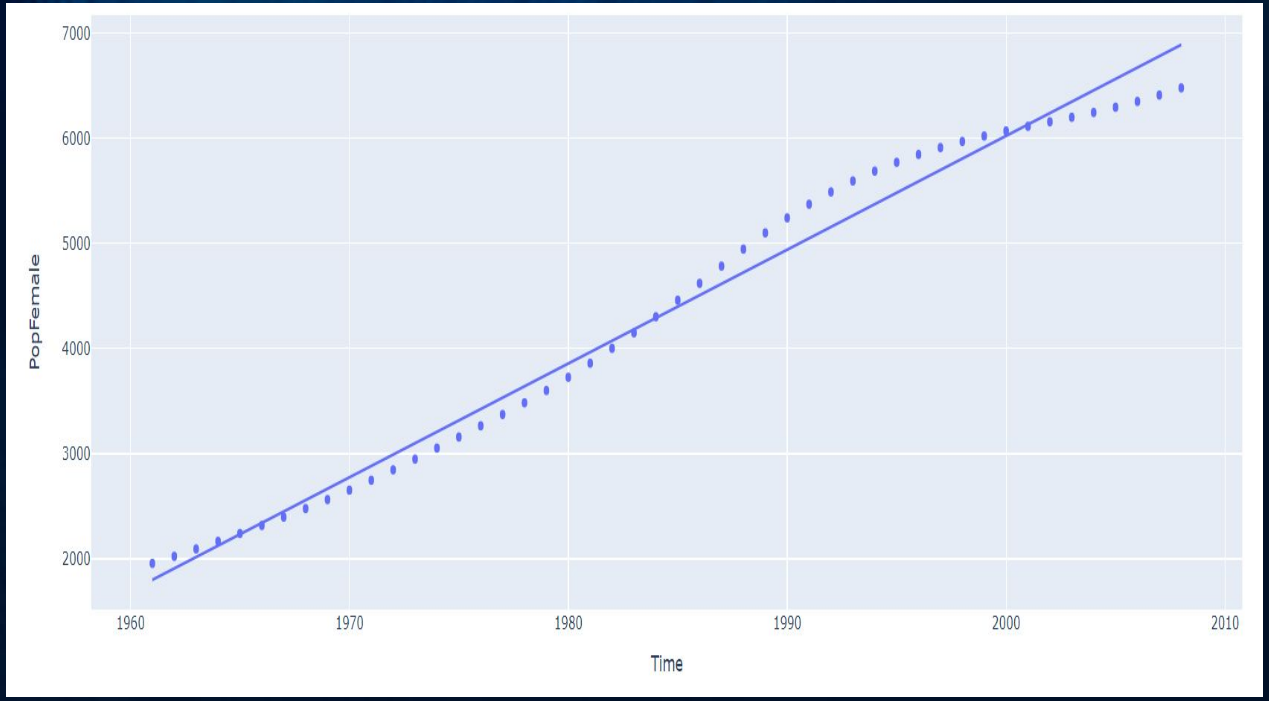

```python
fig3 = px.scatter(data_Zimbabwe, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.984
Model:	OLS	Adj. R-squared:	0.983
Method:	Least Squares	F-statistic:	2770.
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	9.40e-43
Time:	04:15:53	Log-Likelihood:	-320.75
No. Observations:	48	AIC:	645.5
Df Residuals:	46	BIC:	649.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.104e+05	4079.289	-51.565	0.000	-2.19e+05	-2.02e+05
x1	108.1855	2.056	52.632	0.000	104.048	112.323

Omnibus:	2.146	Durbin-Watson:	0.039
Prob(Omnibus):	0.342	Jarque-Bera (JB):	1.363
Skew:	0.114	Prob(JB):	0.506
Kurtosis:	2.207	Cond. No.	2.84e+05



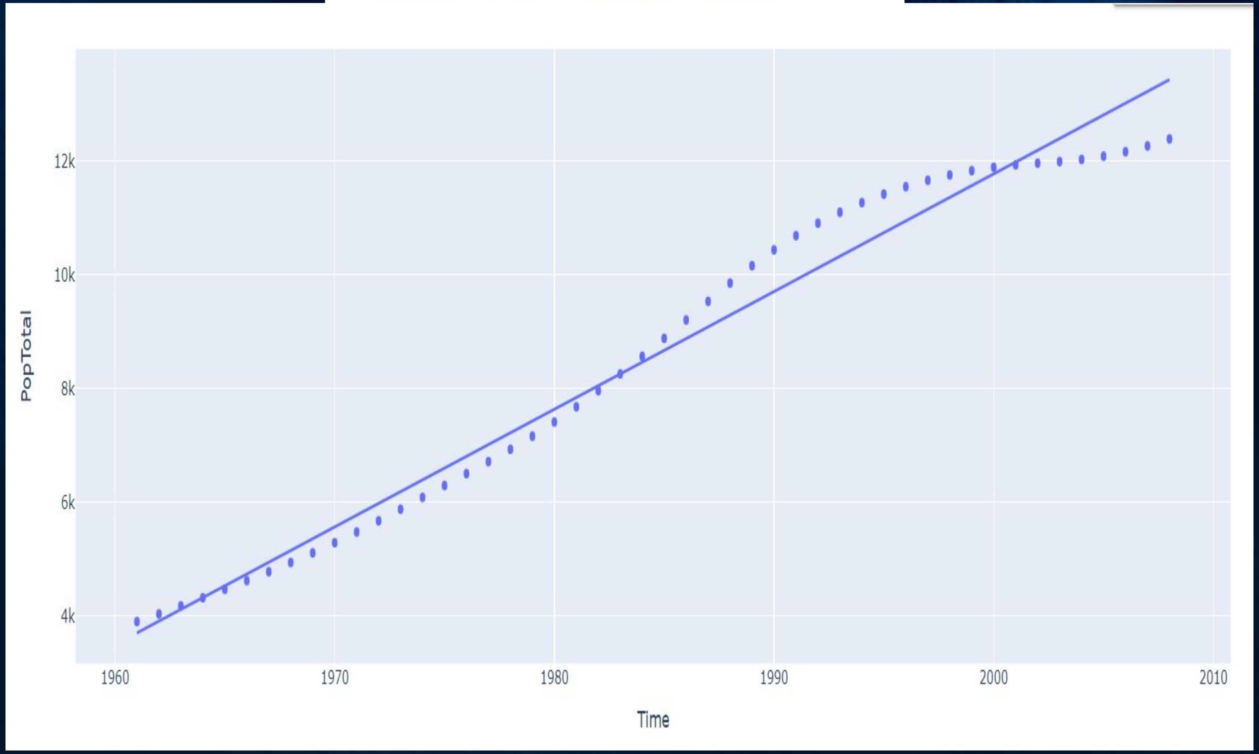
```python
fig4 = px.scatter(data_Zimbabwe, x="Time",
y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.975
Model:	OLS	**Adj. R-squared:**	0.974
Method:	Least Squares	**F-statistic:**	1759.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	2.61e-38
Time:	04:16:00	**Log-Likelihood:**	-362.72
No. Observations:	48	**AIC:**	729.4
Df Residuals:	46	**BIC:**	733.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.016e+05	9779.895	-41.068	0.000	-4.21e+05	-3.82e+05
x1	206.7029	4.928	41.944	0.000	196.783	216.622

Omnibus:	0.339	**Durbin-Watson:**	0.036
Prob(Omnibus):	0.844	**Jarque-Bera (JB):**	0.511
Skew:	-0.004	**Prob(JB):**	0.775
Kurtosis:	2.495	**Cond. No.**	2.84e+05



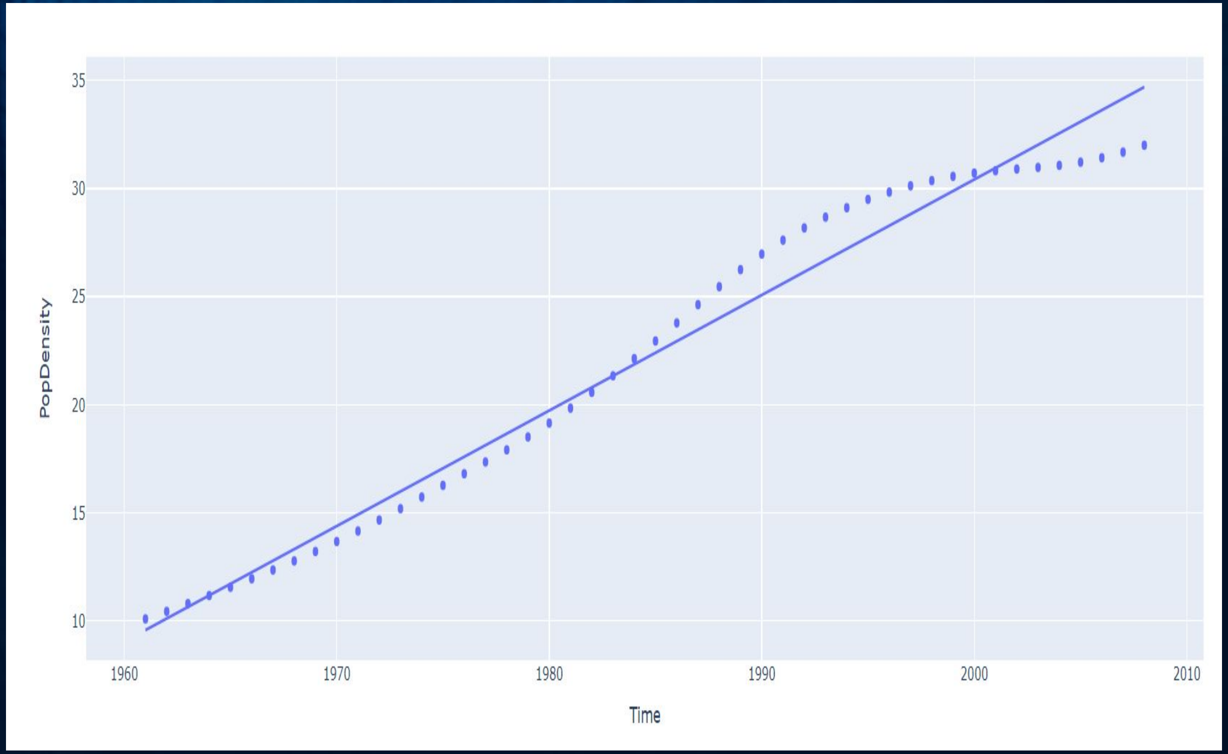

```python
fig5 = px.scatter(data_Zimbabwe, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.975
Model:	OLS	**Adj. R-squared:**	0.974
Method:	Least Squares	**F-statistic:**	1759.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	2.61e-38
Time:	04:16:06	**Log-Likelihood:**	-76.737
No. Observations:	48	**AIC:**	157.5
Df Residuals:	46	**BIC:**	161.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1038.2292	25.281	-41.068	0.000	-1089.116	-987.342
x1	0.5343	0.013	41.945	0.000	0.509	0.560

Omnibus:	0.338	**Durbin-Watson:**	0.036
Prob(Omnibus):	0.844	**Jarque-Bera (JB):**	0.510
Skew:	-0.004	**Prob(JB):**	0.775
Kurtosis:	2.495	**Cond. No.**	2.84e+05



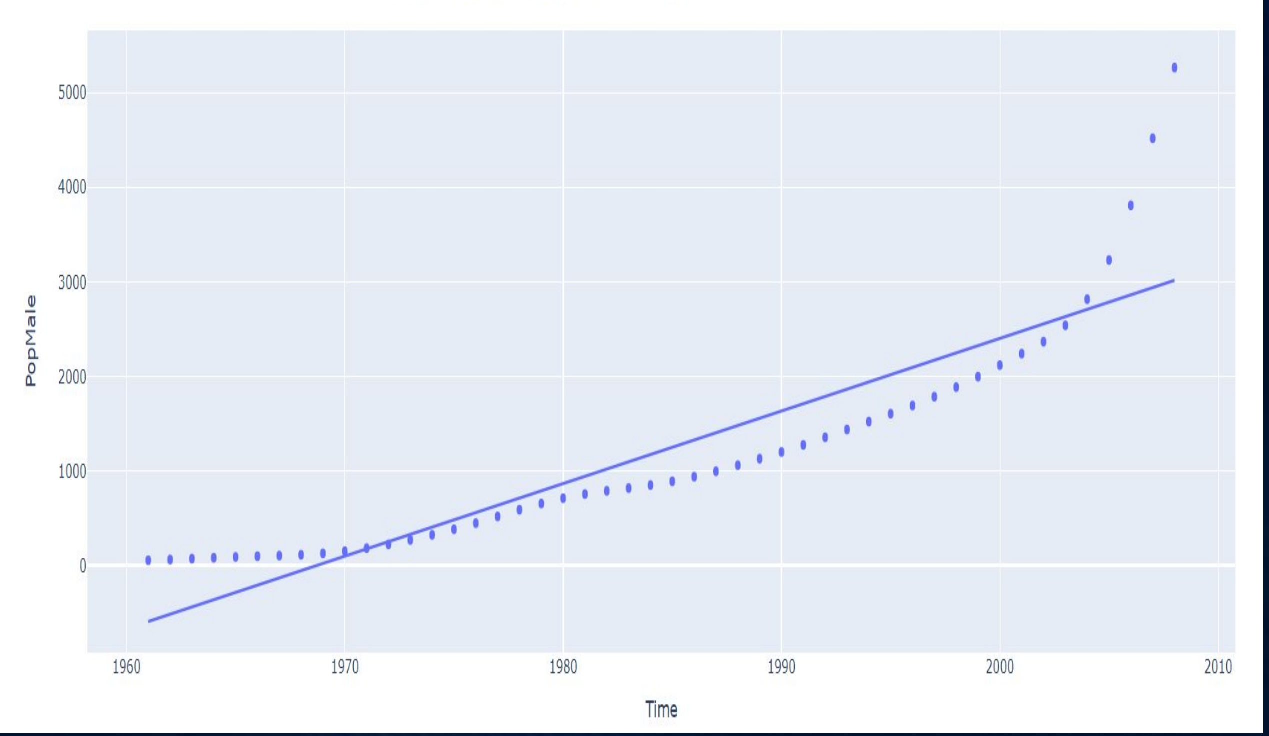
```python
data_UAE = data.loc[data.Location == 'United Arab Emirates']
data_UNFPA.head()
fig2 = px.scatter(data_UAE, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.804
Model:	OLS	**Adj. R-squared:**	0.800
Method:	Least Squares	**F-statistic:**	189.0
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	6.61e-18
Time:	04:16:15	**Log-Likelihood:**	-368.74
No. Observations:	48	**AIC:**	741.5
Df Residuals:	46	**BIC:**	745.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.512e+05	1.11e+04	-13.640	0.000	-1.74e+05	-1.29e+05
x1	76.8004	5.586	13.749	0.000	65.557	88.044

Omnibus:	40.234	**Durbin-Watson:**	0.101
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	120.240
Skew:	2.300	**Prob(JB):**	7.77e-27
Kurtosis:	9.241	**Cond. No.**	2.84e+05



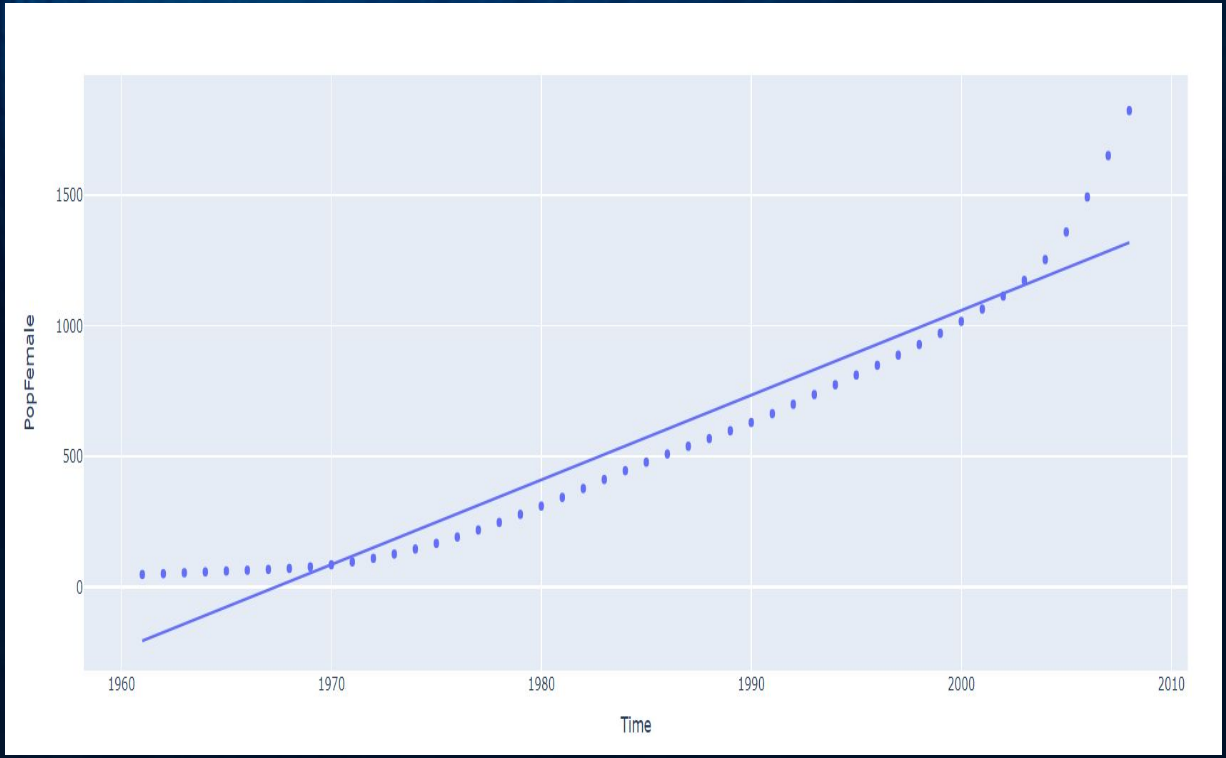

```python
fig3 = px.scatter(data_UAE, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.913
Model:	OLS	**Adj. R-squared:**	0.912
Method:	Least Squares	**F-statistic:**	485.1
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.48e-26
Time:	04:16:22	**Log-Likelihood:**	-304.71
No. Observations:	48	**AIC:**	613.4
Df Residuals:	46	**BIC:**	617.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6.377e+04	2920.604	-21.834	0.000	-6.96e+04	-5.79e+04
x1	32.4142	1.472	22.025	0.000	29.452	35.377

Omnibus:	24.499	**Durbin-Watson:**	0.070
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	36.857
Skew:	1.713	**Prob(JB):**	9.92e-09
Kurtosis:	5.586	**Cond. No.**	2.84e+05



```python
fig4 = px.scatter(data_UAE, x="Time", y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.841
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	242.8
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	5.66e-20
Time:	04:16:27	**Log-Likelihood:**	-379.63
No. Observations:	48	**AIC:**	763.3
Df Residuals:	46	**BIC:**	767.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.15e+05	1.39e+04	-15.456	0.000	-2.43e+05	-1.87e+05
x1	109.2146	7.008	15.583	0.000	95.107	123.322

Omnibus:	37.571	**Durbin-Watson:**	0.094
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	99.313
Skew:	2.198	**Prob(JB):**	2.72e-22
Kurtosis:	8.508	**Cond. No.**	2.84e+05

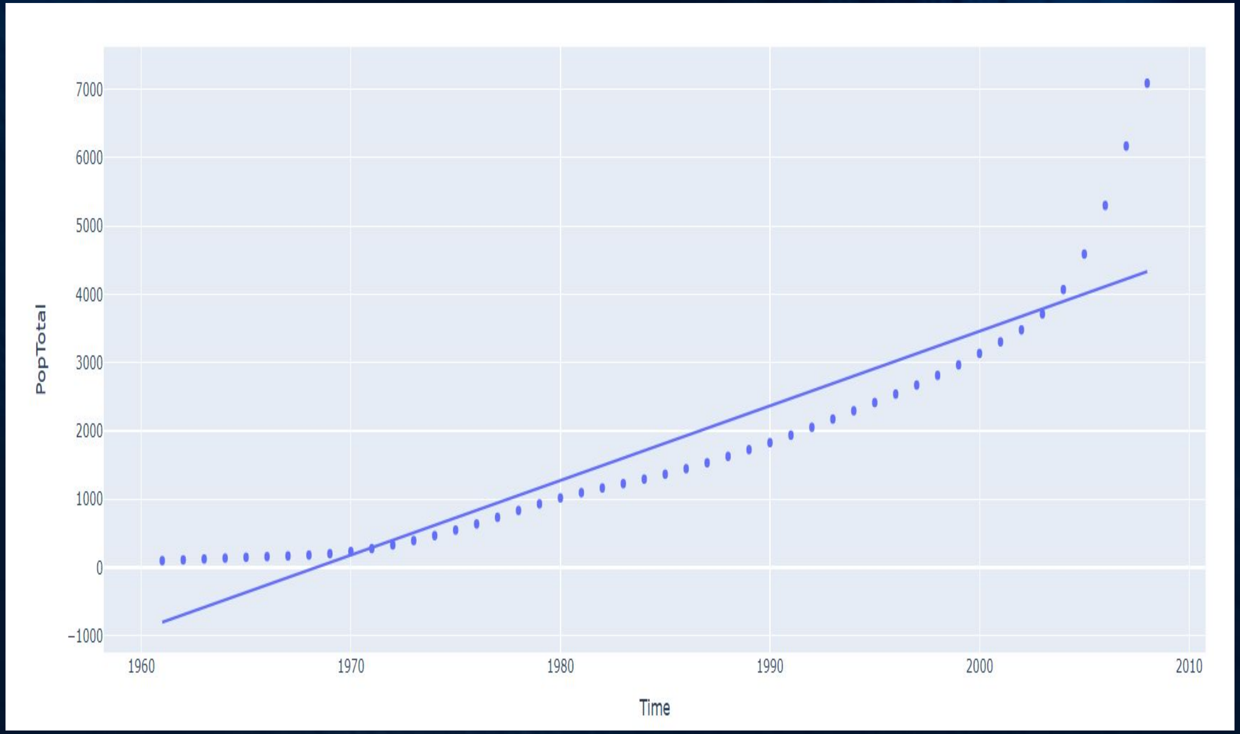
```python
fig5 = px.scatter(data_UAE, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

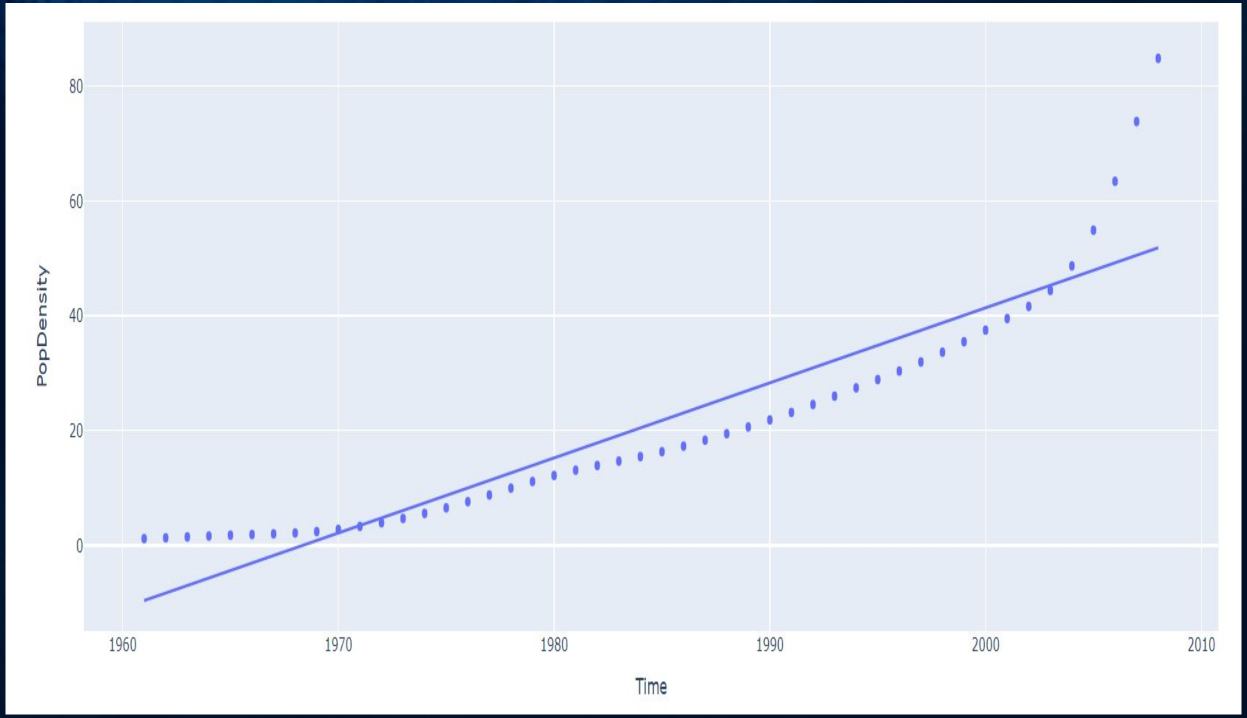
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.841
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	242.8
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	5.66e-20
Time:	04:16:34	**Log-Likelihood:**	-167.18
No. Observations:	48	**AIC:**	338.4
Df Residuals:	46	**BIC:**	342.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2571.4079	166.370	-15.456	0.000	-2906.294	-2236.522
x1	1.3064	0.084	15.583	0.000	1.138	1.475

Omnibus:	37.571	**Durbin-Watson:**	0.094
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	99.310
Skew:	2.198	**Prob(JB):**	2.72e-22
Kurtosis:	8.508	**Cond. No.**	2.84e+05





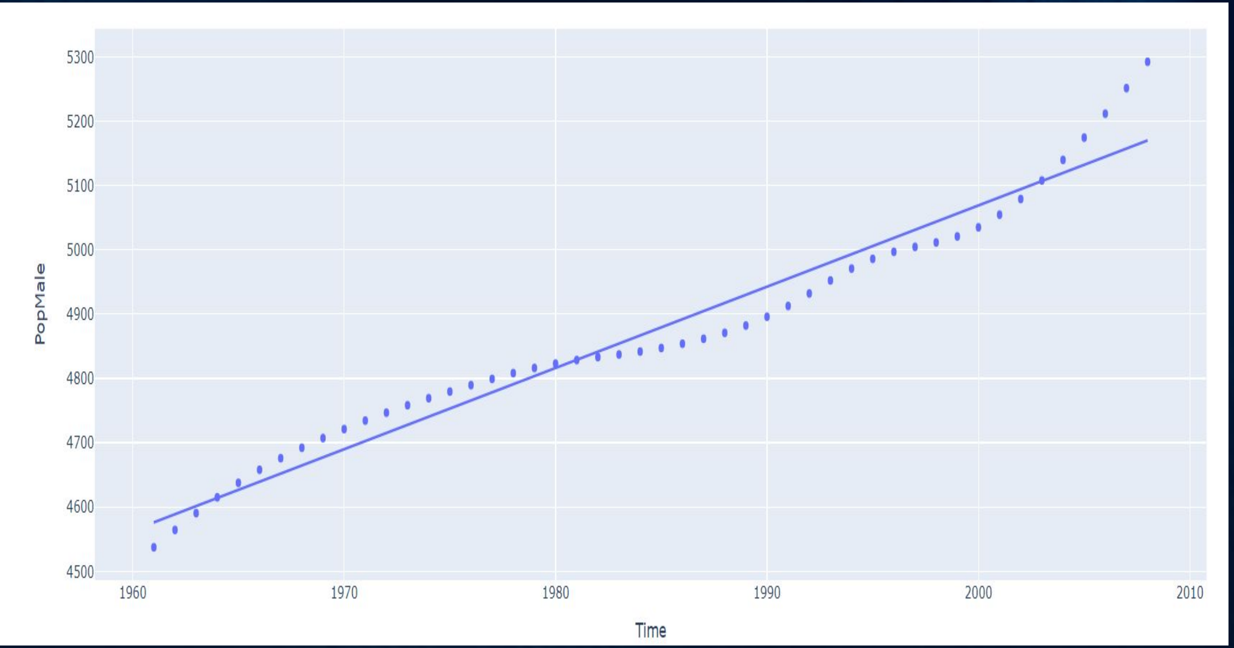
```python
data_Belgium = data.loc[data.Location == 'Belgium']
data_UNFPA.head()
fig2 = px.scatter(data_Belgium, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.958
Model:	OLS	**Adj. R-squared:**	0.957
Method:	Least Squares	**F-statistic:**	1047.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	2.71e-33
Time:	04:17:47	**Log-Likelihood:**	-241.02
No. Observations:	48	**AIC:**	486.0
Df Residuals:	46	**BIC:**	489.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.019e+04	774.780	-26.065	0.000	-2.18e+04	-1.86e+04
x1	12.6317	0.390	32.355	0.000	11.846	13.418

Omnibus:	11.296	**Durbin-Watson:**	0.076
Prob(Omnibus):	0.004	**Jarque-Bera (JB):**	11.194
Skew:	1.010	**Prob(JB):**	0.00371
Kurtosis:	4.230	**Cond. No.**	2.84e+05



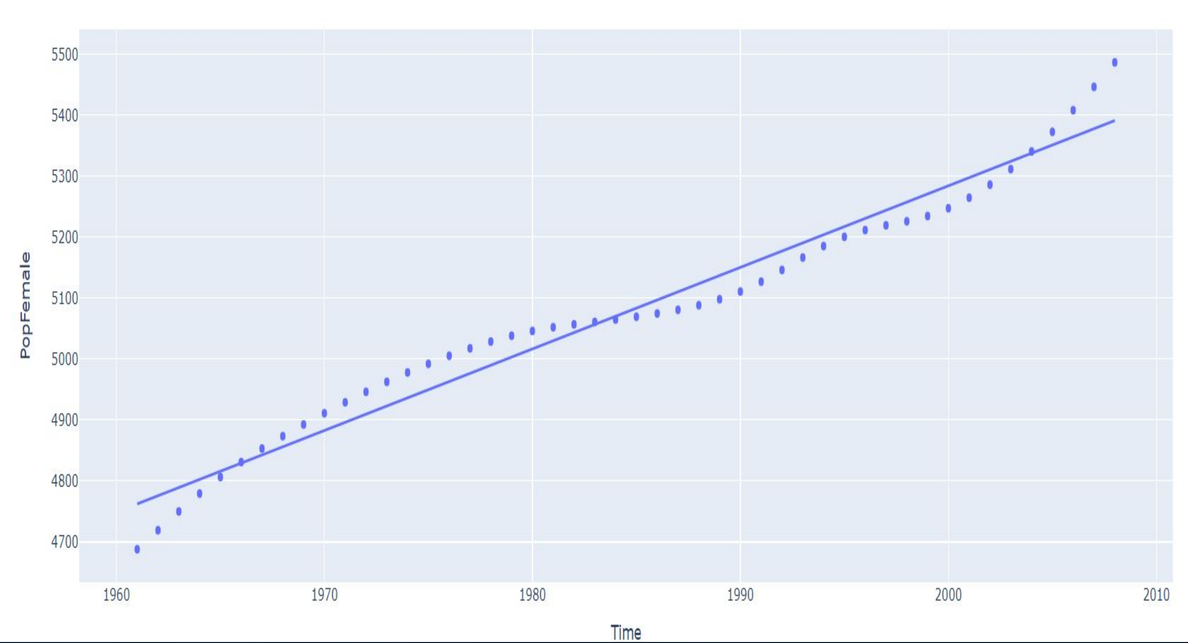

```python
fig3 = px.scatter(data_Belgium, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.965
Model:	OLS	**Adj. R-squared:**	0.964
Method:	Least Squares	**F-statistic:**	1262.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.32e-35
Time:	04:17:54	**Log-Likelihood:**	-239.29
No. Observations:	48	**AIC:**	482.6
Df Residuals:	46	**BIC:**	486.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.147e+04	747.359	-28.733	0.000	-2.3e+04	-2e+04
x1	13.3791	0.377	35.527	0.000	12.621	14.137

Omnibus:	1.305	**Durbin-Watson:**	0.082
Prob(Omnibus):	0.521	**Jarque-Bera (JB):**	1.318
Skew:	0.345	**Prob(JB):**	0.517
Kurtosis:	2.572	**Cond. No.**	2.84e+05



```python
fig4 = px.scatter(data_Belgium, x="Time", y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.964
Model:	OLS	**Adj. R-squared:**	0.963
Method:	Least Squares	**F-statistic:**	1215.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.01e-34
Time:	04:17:59	**Log-Likelihood:**	-272.12
No. Observations:	48	**AIC:**	548.2
Df Residuals:	46	**BIC:**	552.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-4.167e+04	1480.988	-28.136	0.000	-4.46e+04	-3.87e+04
x1	26.0109	0.746	34.855	0.000	24.509	27.513

Omnibus:	5.416	**Durbin-Watson:**	0.082
Prob(Omnibus):	0.067	**Jarque-Bera (JB):**	4.459
Skew:	0.726	**Prob(JB):**	0.108
Kurtosis:	3.345	**Cond. No.**	2.84e+05

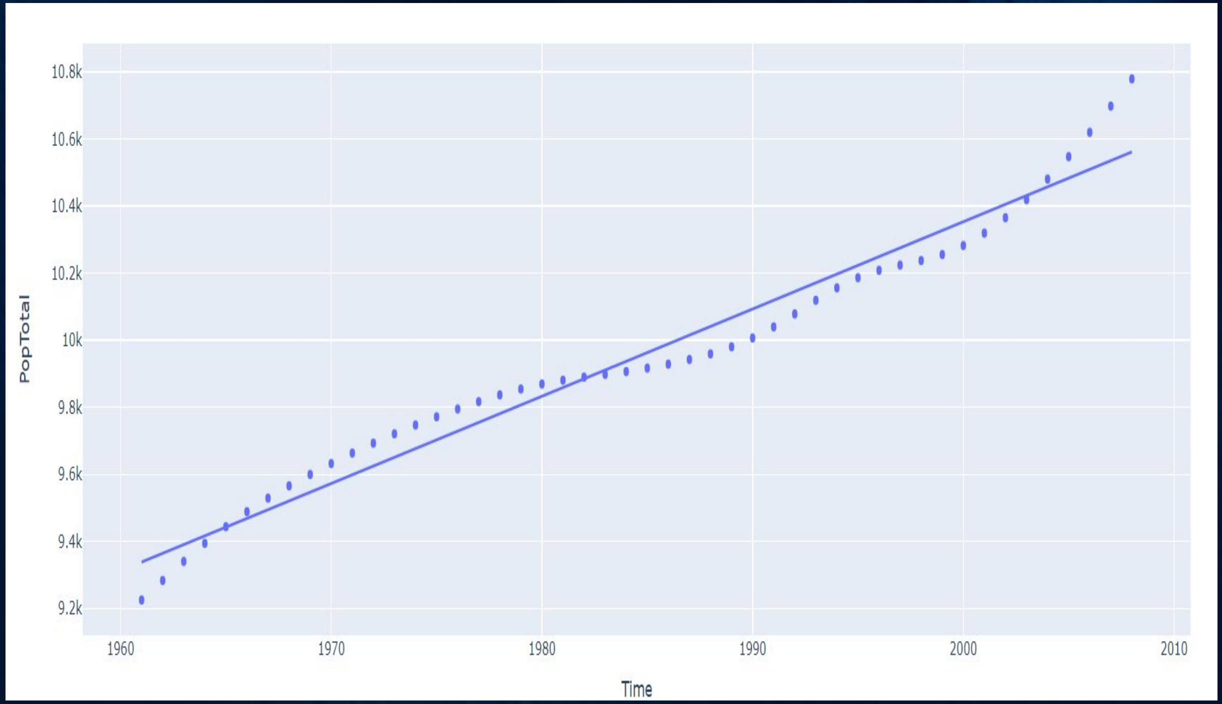
```python
fig5 = px.scatter(data_Belgium, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

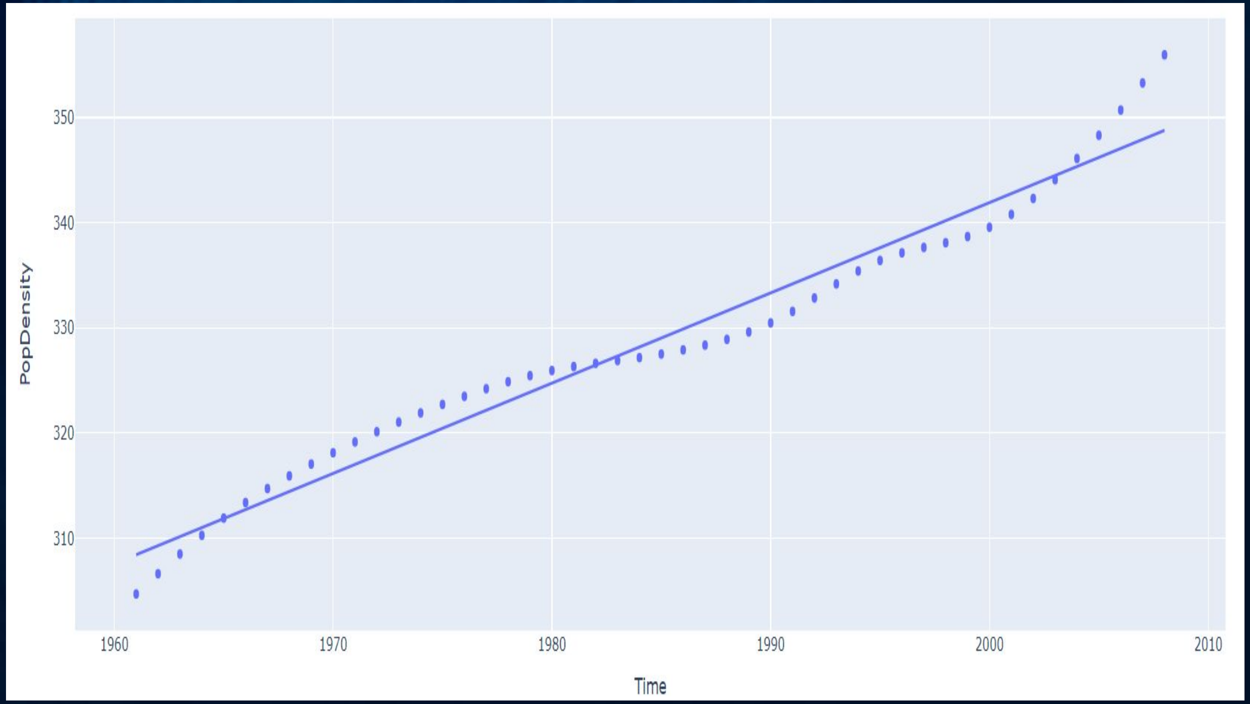
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.964
Model:	OLS	**Adj. R-squared:**	0.963
Method:	Least Squares	**F-statistic:**	1215.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.01e-34
Time:	04:18:05	**Log-Likelihood:**	-108.41
No. Observations:	48	**AIC:**	220.8
Df Residuals:	46	**BIC:**	224.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-1376.1285	48.911	-28.136	0.000	-1474.580	-1277.677
x1	0.8590	0.025	34.855	0.000	0.809	0.909

Omnibus:	5.416	**Durbin-Watson:**	0.082
Prob(Omnibus):	0.067	**Jarque-Bera (JB):**	4.459
Skew:	0.726	**Prob(JB):**	0.108
Kurtosis:	3.345	**Cond. No.**	2.84e+05





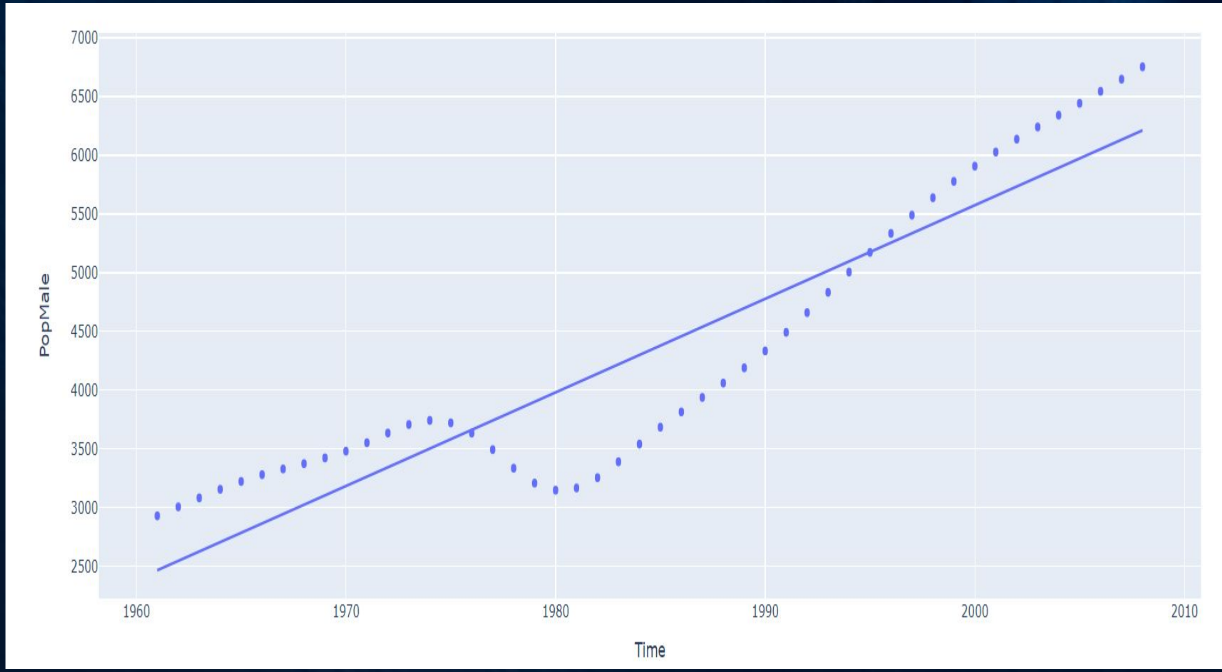
```python
data_Cambodia = data.loc[data.Location == 'Cambodia']
data_UNFPA.head()
fig2 = px.scatter(data_Cambodia, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.847
Model:	OLS	**Adj. R-squared:**	0.844
Method:	Least Squares	**F-statistic:**	254.9
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	2.19e-20
Time:	04:18:19	**Log-Likelihood:**	-363.31
No. Observations:	48	**AIC:**	730.6
Df Residuals:	46	**BIC:**	734.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.537e+05	9900.628	-15.528	0.000	-1.74e+05	-1.34e+05
x1	79.6577	4.989	15.967	0.000	69.616	89.700

Omnibus:	12.258	**Durbin-Watson:**	0.029
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	5.698
Skew:	-0.621	**Prob(JB):**	0.0579
Kurtosis:	1.858	**Cond. No.**	2.84e+05



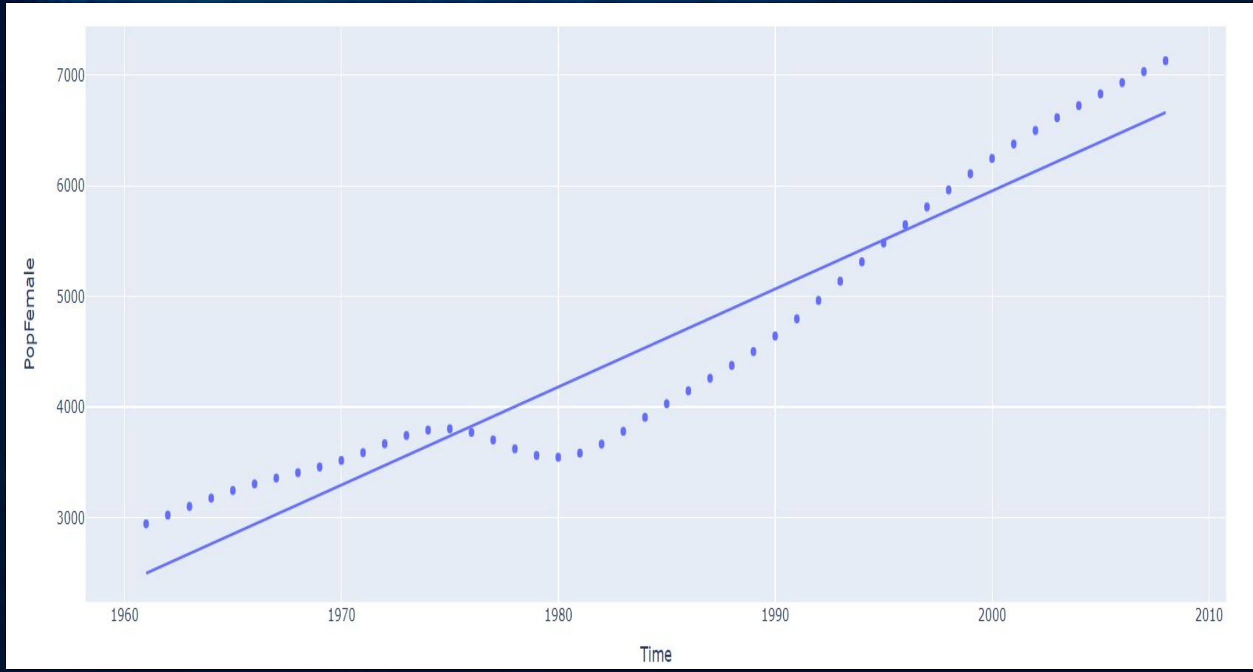

```python
fig3 = px.scatter(data_Cambodia, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.905
Model:	OLS	**Adj. R-squared:**	0.902
Method:	Least Squares	**F-statistic:**	436.0
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	4.20e-25
Time:	04:18:31	**Log-Likelihood:**	-355.59
No. Observations:	48	**AIC:**	715.2
Df Residuals:	46	**BIC:**	718.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.714e+05	8428.749	-20.336	0.000	-1.88e+05	-1.54e+05
x1	88.6818	4.247	20.880	0.000	80.133	97.231

Omnibus:	19.944	**Durbin-Watson:**	0.024
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.272
Skew:	-0.472	**Prob(JB):**	0.0717
Kurtosis:	1.679	**Cond. No.**	2.84e+05



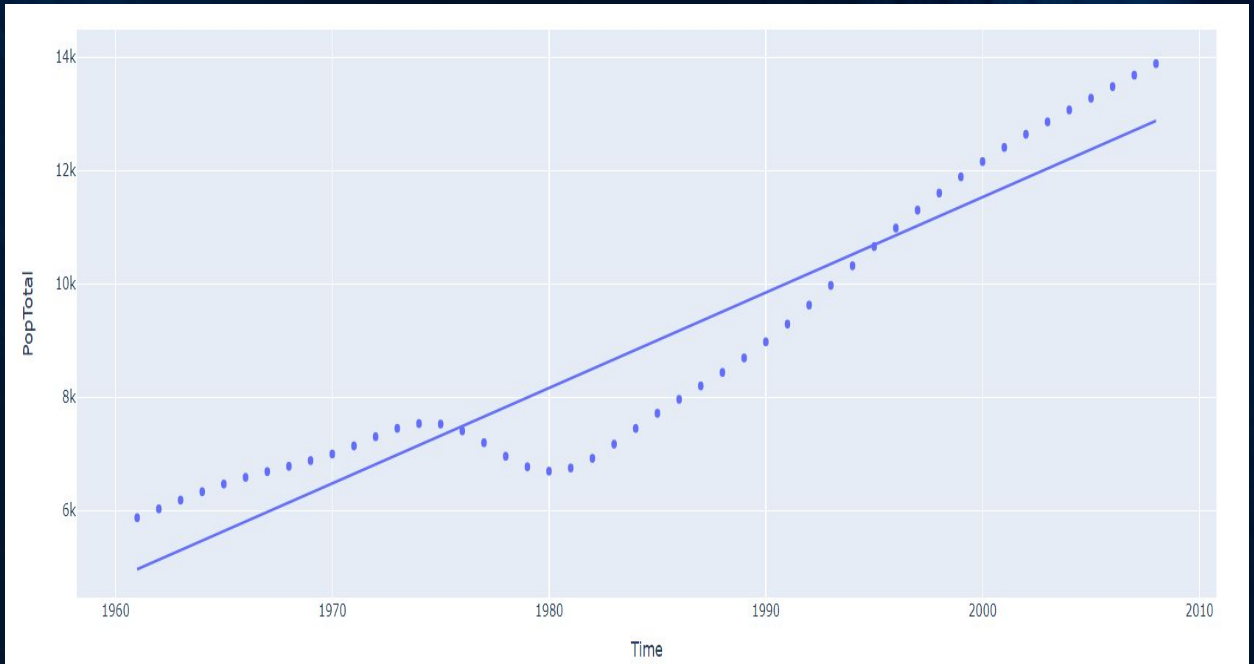
```python
fig4 = px.scatter(data_Cambodia, x="Time",
y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.879
Model:	OLS	**Adj. R-squared:**	0.876
Method:	Least Squares	**F-statistic:**	332.9
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.08e-22
Time:	04:18:36	**Log-Likelihood:**	-392.82
No. Observations:	48	**AIC:**	789.6
Df Residuals:	46	**BIC:**	793.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.252e+05	1.83e+04	-17.759	0.000	-3.62e+05	-2.88e+05
x1	168.3395	9.226	18.246	0.000	149.769	186.910

Omnibus:	15.411	**Durbin-Watson:**	0.027
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.483
Skew:	-0.552	**Prob(JB):**	0.0645
Kurtosis:	1.766	**Cond. No.**	2.84e+05



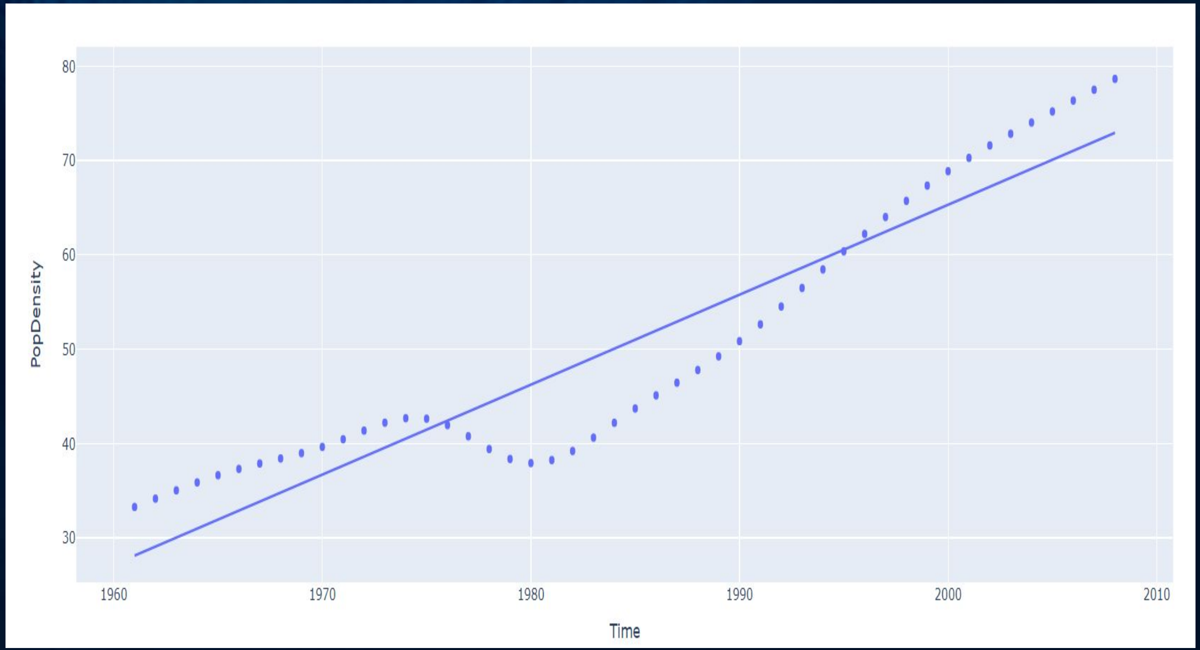

```python
fig5 = px.scatter(data_Cambodia, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.879
Model:	OLS	**Adj. R-squared:**	0.876
Method:	Least Squares	**F-statistic:**	332.9
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.08e-22
Time:	04:18:41	**Log-Likelihood:**	-144.50
No. Observations:	48	**AIC:**	293.0
Df Residuals:	46	**BIC:**	296.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1841.9995	103.726	-17.758	0.000	-2050.789	-1633.210
x1	0.9537	0.052	18.246	0.000	0.848	1.059

Omnibus:	15.411	**Durbin-Watson:**	0.027
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.483
Skew:	-0.552	**Prob(JB):**	0.0645
Kurtosis:	1.766	**Cond. No.**	2.84e+05



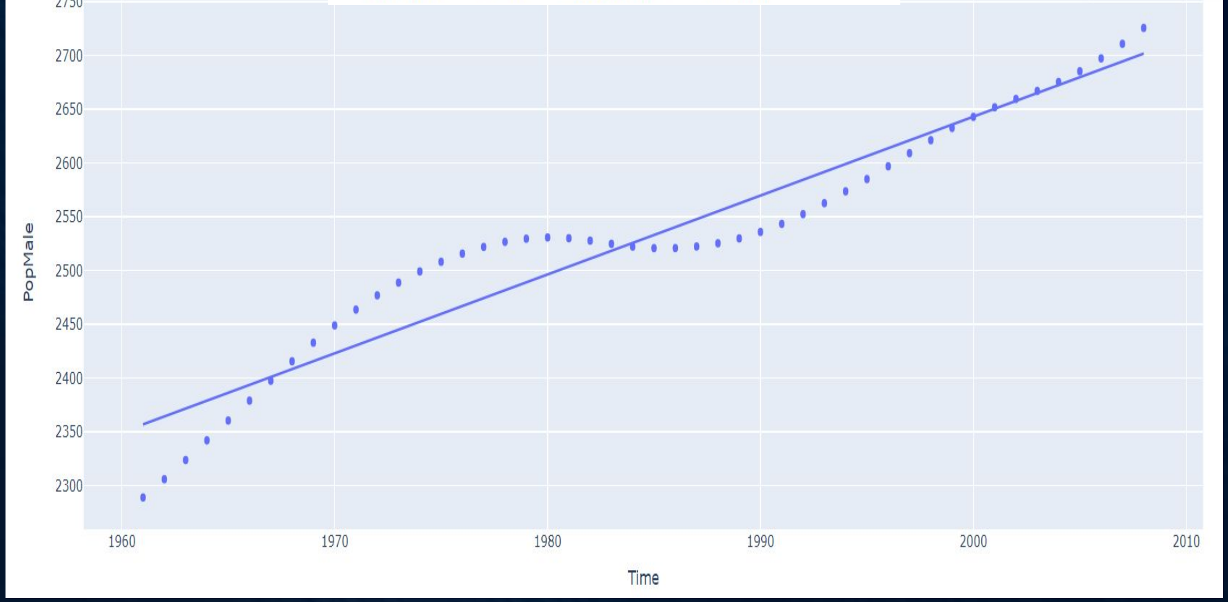
```python
data_Denmark = data.loc[data.Location == 'Denmark']
data_UNFPA.head()
fig2 = px.scatter(data_Denmark, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.920
Model:	OLS	Adj. R-squared:	0.918
Method:	Least Squares	F-statistic:	529.6
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	7.02e-27
Time:	04:18:59	Log-Likelihood:	-231.31
No. Observations:	48	AIC:	466.6
Df Residuals:	46	BIC:	470.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.203e+04	632.870	-19.016	0.000	-1.33e+04	-1.08e+04
x1	7.3389	0.319	23.013	0.000	6.697	7.981

Omnibus:	1.806	Durbin-Watson:	0.046
Prob(Omnibus):	0.405	Jarque-Bera (JB):	1.198
Skew:	-0.046	Prob(JB):	0.549
Kurtosis:	2.232	Cond. No.	2.84e+05



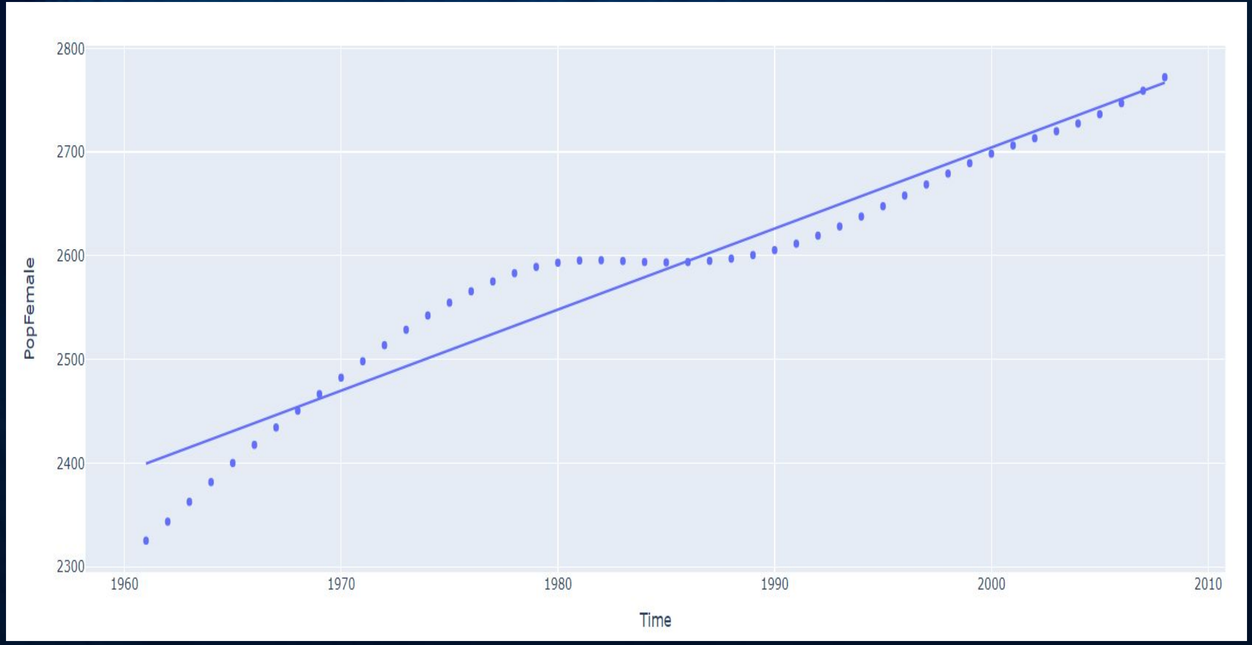

```python
fig3 = px.scatter(data_Denmark, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.931
Model:	OLS	Adj. R-squared:	0.930
Method:	Least Squares	F-statistic:	625.4
Date:	Thu, 10 Mar 2022	Prob (F-statistic):	2.02e-28
Time:	04:19:07	Log-Likelihood:	-230.34
No. Observations:	48	AIC:	464.7
Df Residuals:	46	BIC:	468.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.293e+04	620.207	-20.842	0.000	-1.42e+04	-1.17e+04
x1	7.8155	0.313	25.008	0.000	7.186	8.445

Omnibus:	0.064	Durbin-Watson:	0.040
Prob(Omnibus):	0.969	Jarque-Bera (JB):	0.053
Skew:	-0.047	Prob(JB):	0.974
Kurtosis:	2.868	Cond. No.	2.84e+05



```python
fig4 = px.scatter(data_Denmark, x="Time",
y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.928
Model:	OLS	**Adj. R-squared:**	0.927
Method:	Least Squares	**F-statistic:**	596.1
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	5.65e-28
Time:	04:19:15	**Log-Likelihood:**	-263.27
No. Observations:	48	**AIC:**	530.5
Df Residuals:	46	**BIC:**	534.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.496e+04	1231.795	-20.264	0.000	-2.74e+04	-2.25e+04
x1	15.1543	0.621	24.415	0.000	13.905	16.404

Omnibus:	0.138	**Durbin-Watson:**	0.044
Prob(Omnibus):	0.933	**Jarque-Bera (JB):**	0.340
Skew:	-0.056	**Prob(JB):**	0.844
Kurtosis:	2.603	**Cond. No.**	2.84e+05

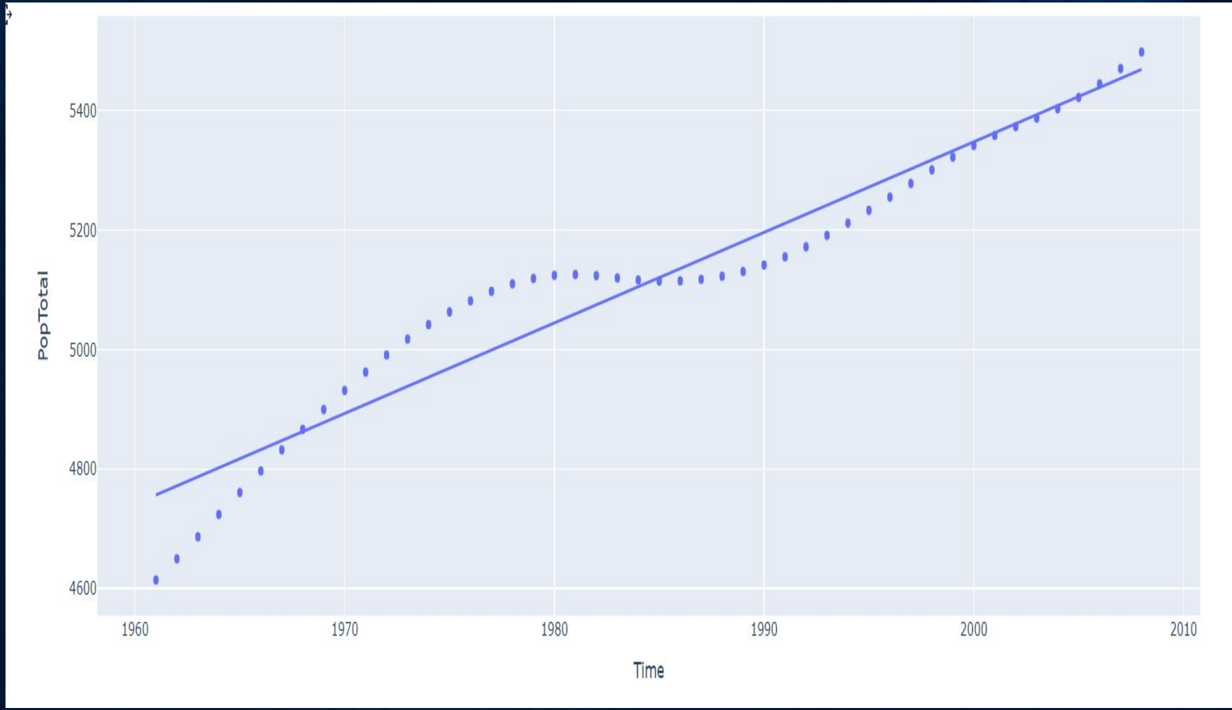
```python
fig5 = px.scatter(data_Denmark, x="Time",
y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

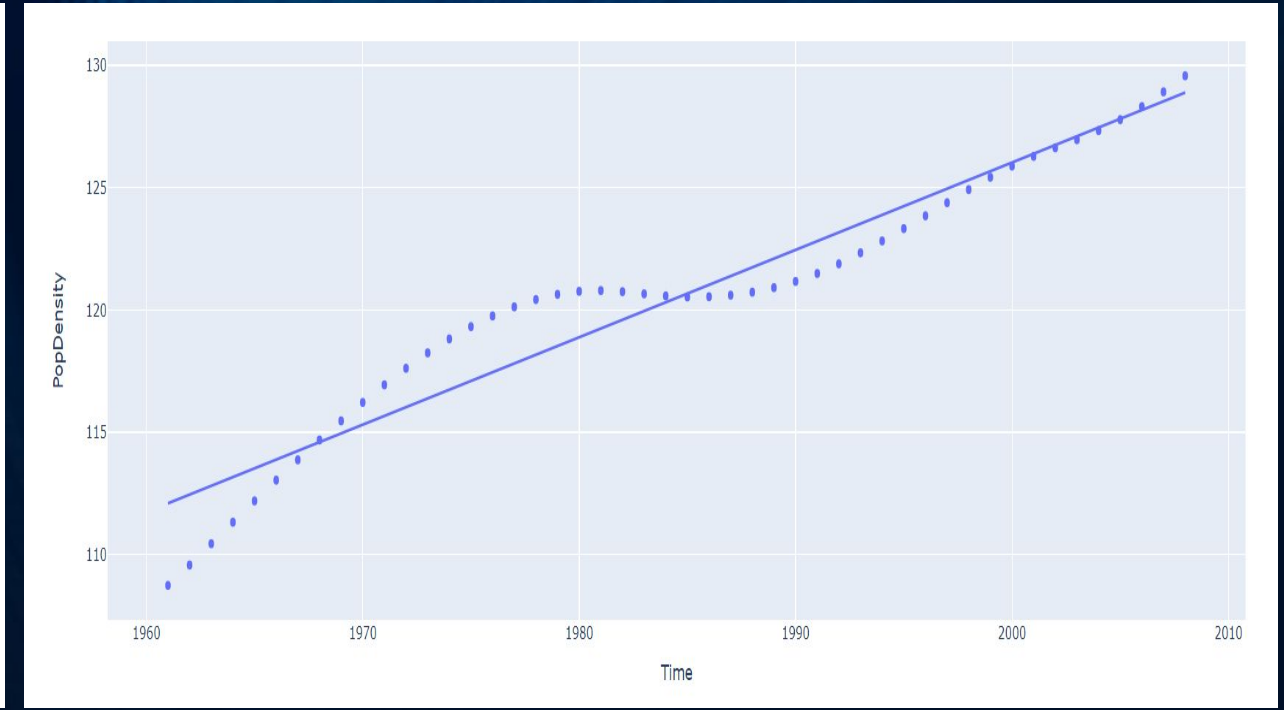
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.928
Model:	OLS	**Adj. R-squared:**	0.927
Method:	Least Squares	**F-statistic:**	596.1
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	5.64e-28
Time:	04:19:21	**Log-Likelihood:**	-83.375
No. Observations:	48	**AIC:**	170.7
Df Residuals:	46	**BIC:**	174.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-588.2879	29.030	-20.265	0.000	-646.722	-529.853
x1	0.3572	0.015	24.416	0.000	0.328	0.387

Omnibus:	0.138	**Durbin-Watson:**	0.044
Prob(Omnibus):	0.933	**Jarque-Bera (JB):**	0.340
Skew:	-0.056	**Prob(JB):**	0.844
Kurtosis:	2.603	**Cond. No.**	2.84e+05





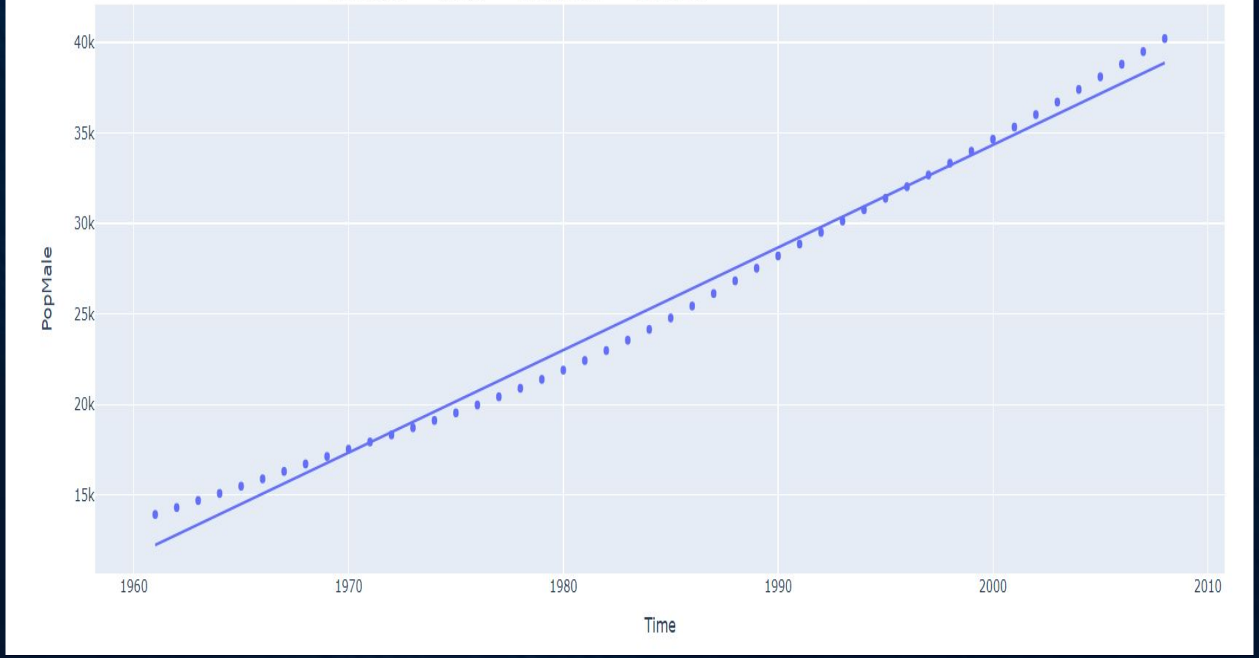
```python
data_Egypt = data.loc[data.Location == 'Egypt']
data_UNFPA.head()
fig2 = px.scatter(data_Egypt, x="Time", y="PopMale", trendline="ols")
fig2.show()
results = px.get_trendline_results(fig2)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.989
Model:	OLS	**Adj. R-squared:**	0.989
Method:	Least Squares	**F-statistic:**	4204.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	7.25e-47
Time:	04:19:35	**Log-Likelihood:**	-390.22
No. Observations:	48	**AIC:**	784.4
Df Residuals:	46	**BIC:**	788.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.099e+06	1.73e+04	-63.365	0.000	-1.13e+06	-1.06e+06
x1	566.6536	8.739	64.840	0.000	549.062	584.245

Omnibus:	7.003	**Durbin-Watson:**	0.022
Prob(Omnibus):	0.030	**Jarque-Bera (JB):**	2.755
Skew:	0.251	**Prob(JB):**	0.252
Kurtosis:	1.939	**Cond. No.**	2.84e+05



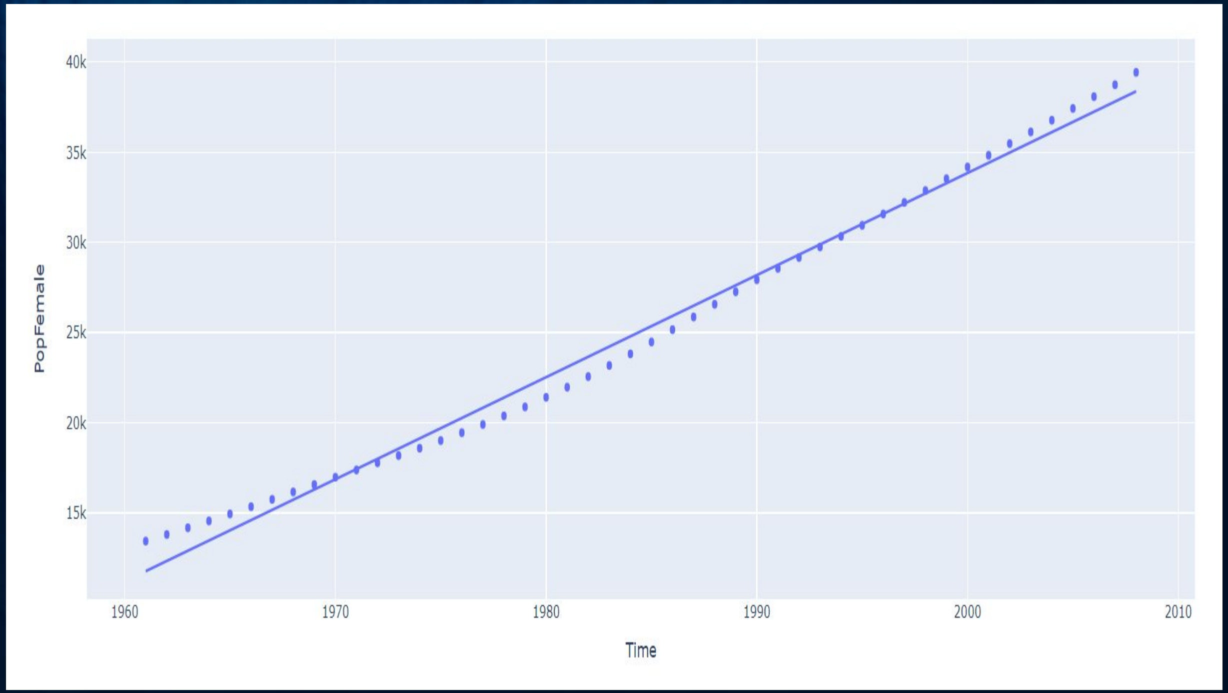

```python
fig3 = px.scatter(data_Egypt, x="Time", y="PopFemale",
trendline="ols")
fig3.show()
results = px.get_trendline_results(fig3)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.991
Model:	OLS	**Adj. R-squared:**	0.991
Method:	Least Squares	**F-statistic:**	5022.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	1.27e-48
Time:	04:19:40	**Log-Likelihood:**	-385.90
No. Observations:	48	**AIC:**	775.8
Df Residuals:	46	**BIC:**	779.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.098e+06	1.59e+04	-69.280	0.000	-1.13e+06	-1.07e+06
x1	566.0016	7.987	70.863	0.000	549.924	582.079

Omnibus:	3.314	**Durbin-Watson:**	0.024
Prob(Omnibus):	0.191	**Jarque-Bera (JB):**	1.888
Skew:	0.219	**Prob(JB):**	0.389
Kurtosis:	2.133	**Cond. No.**	2.84e+05



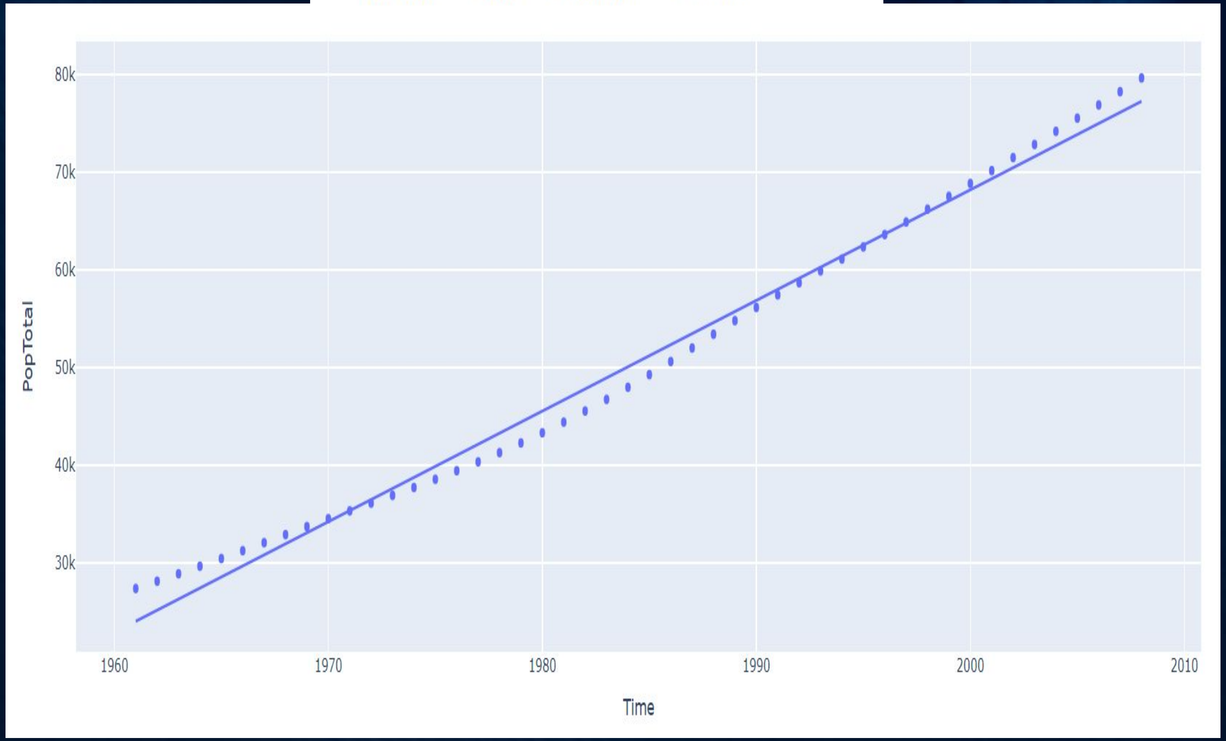
```python
fig4 = px.scatter(data_Egypt, x="Time", y="PopTotal",trendline="ols")
fig4.show()
results = px.get_trendline_results(fig4)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.990
Model:	OLS	**Adj. R-squared:**	0.990
Method:	Least Squares	**F-statistic:**	4604.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	9.19e-48
Time:	04:19:45	**Log-Likelihood:**	-421.29
No. Observations:	48	**AIC:**	846.6
Df Residuals:	46	**BIC:**	850.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.197e+06	3.31e+04	-66.319	0.000	-2.26e+06	-2.13e+06
x1	1132.6552	16.694	67.849	0.000	1099.052	1166.258

Omnibus:	5.163	**Durbin-Watson:**	0.023
Prob(Omnibus):	0.076	**Jarque-Bera (JB):**	2.376
Skew:	0.239	**Prob(JB):**	0.305
Kurtosis:	2.020	**Cond. No.**	2.84e+05




```python
fig5 = px.scatter(data_Egypt, x="Time", y="PopDensity",trendline="ols")
fig5.show()
results = px.get_trendline_results(fig5)
results.px_fit_results.iloc[0].summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.990
Model:	OLS	**Adj. R-squared:**	0.990
Method:	Least Squares	**F-statistic:**	4604.
Date:	Thu, 10 Mar 2022	**Prob (F-statistic):**	9.18e-48
Time:	04:19:50	**Log-Likelihood:**	-89.933
No. Observations:	48	**AIC:**	183.9
Df Residuals:	46	**BIC:**	187.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2207.1735	33.280	-66.321	0.000	-2274.164	-2140.184
x1	1.1378	0.017	67.850	0.000	1.104	1.172

Omnibus:	5.163	**Durbin-Watson:**	0.023
Prob(Omnibus):	0.076	**Jarque-Bera (JB):**	2.376
Skew:	0.239	**Prob(JB):**	0.305
Kurtosis:	2.020	**Cond. No.**	2.84e+05