importing all required libraries

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import sklearn.linear_model as lm
5 %matplotlib inline
6 from sklearn.metrics import mean_squared_error
7 from sklearn.metrics import mean_absolute_error
```

installing icecream to make code easrier

```
1 !pip install icecream
2 from icecream import ic
3
```

```
Looking in indexes: https://pypi.org/simple, h
Requirement already satisfied: icecream in /us
Requirement already satisfied: colorama>=0.3.9
Requirement already satisfied: pygments>=2.2.0
Requirement already satisfied: executing>=0.3.
Requirement already satisfied: asttokens>=2.0.
Requirement already satisfied: six in /usr/loc
```

reading csv file

```
1 df=pd.read_csv('singdata1950-2022.csv', header=N
2
```

making a scaling factor

```
1 year=df.iloc[1:,0].values.astype('int')
2 population=df.iloc[1:,1].values.astype('int')
3 #first is for rows and second is columns
4 scaling_factor=max(population)
5 population=population/max(population)
6 print(year)
7 print(population)
```

```
[1950 1951 1952 1953 1954 1955 1956 1957 1958
 1964 1965 1966 1967 1968 1969 1970 1971 1972
 1978 1979 1980 1981 1982 1983 1984 1985 1986
 1992 1993 1994 1995 1996 1997 1998 1999 2000
 2006 2007 2008 2009 2010 2011 2012 2013 2014
 2020 2021]
[0.17920358 0.18726871 0.19759558 0.20895688 (
 0.240481   0.25351302 0.26628941 0.2782819  (
 0.31471522 0.31734516 0.32288555 0.33082794 (
 0.35276158 0.35810911 0.36372086 0.37045226 (
```

singdata1950-2022.csv

1 to 10 of 72 entries Filter

Year	Population
1950	1022100
1951	1068100
1952	1127000
1953	1191800
1954	1248200
1955	1305500
1956	1371600
1957	1445929
1958	1518800
1959	1587200

Show 10 per page

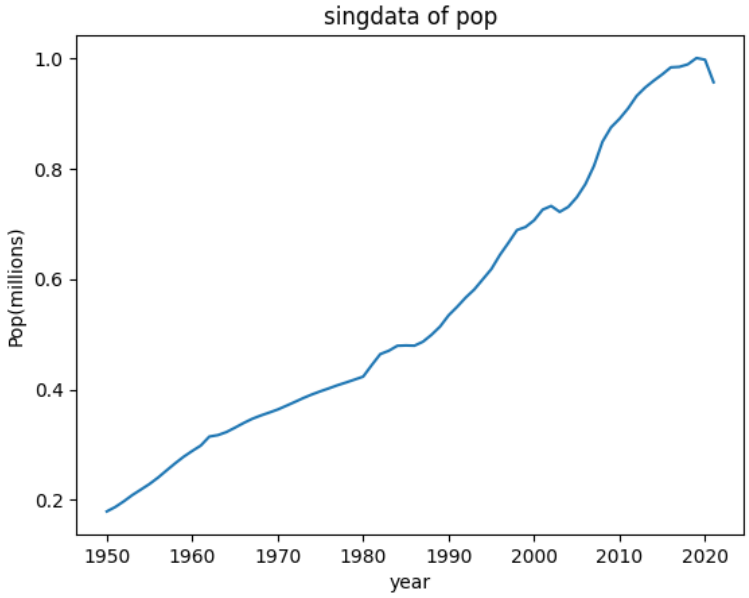
1 2 3 4 5 6 7 8

```
0.39094819 0.39669898 0.40208157 0.4076921  (
0.42323412 0.44407896 0.46400175 0.47006725 (
0.47923905 0.48650047 0.49900475 0.5138714  (
0.56643446 0.58094695 0.59945764 0.61794746 (
0.68855361 0.69407822 0.70620466 0.72551274 (
0.73053627 0.747911   0.77168611 0.80451363 (
0.89009741 0.90884988 0.93142329 0.94662868 (
0.98311829 0.98398967 0.98862239 1.         (
```

ploting

```
1 plt.xlabel('year')
2 plt.ylabel('Pop(millions)')
3 plt.title('singdata of pop')
4 plt.plot(year,population)
```

```
[<matplotlib.lines.Line2D at 0x7f96727d8370>]
```



making test data

```
1 test_year=year[-3:].reshape(-1,1)
2 print(test_year)
```

```
[[2019]
 [2020]
 [2021]]
```

removing test data from the array

```
1 test_pop=population[-3:].reshape(-1,1)
2 print(test_pop)
```

```
[[1.        ]
 [0.99688581]
 [0.95616727]]
```

training the ml

```
1 train_pop, train_year = population[:-3].reshape(
2 print(train_year)
```

```
[1961]
[1962]
[1963]
[1964]
[1965]
[1966]
[1967]
[1968]
[1969]
[1970]
[1971]
[1972]
[1973]
[1974]
[1975]
[1976]
[1977]
[1978]
[1979]
[1980]
[1981]
[1982]
[1983]
[1984]
[1985]
[1986]
[1987]
[1988]
[1989]
[1990]
[1991]
[1992]
[1993]
```

```
[2005]
[2006]
[2007]
[2008]
[2009]
[2010]
[2011]
[2012]
[2013]
[2014]
[2015]
[2016]
[2017]
[2018]]
```

finding the intercept and slope

```
lrp=lm.LinearRegression()
lrp.fit(train_year, train_pop)
slope=lrp.coef_[0]
intercept=lrp.intercept_
print("y="+str(slope)+'x'+str(intercept))
```

```
y=[0.01167588]x[-22.635137]
```

predicting the data then finding mean squared error and mean absolute error

```
train_pred=lrp.predict(train_year)
mse_train=mean_squared_error(train_pred, train_p
mae_train=mean_absolute_error(train_pred, train_

test_pred=lrp.predict(test_year)
mse_test=mean_squared_error(test_pred,test_pop)
mae_test=mean_absolute_error(test_pred, test_pop

ic(mse_train)
ic(mse_test)
ic(mae_train)
ic(mae_test)
```

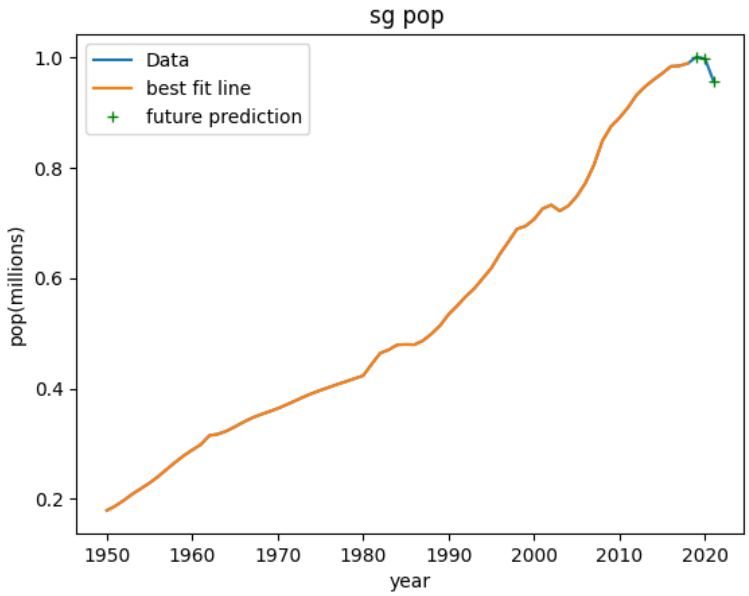
```
ic| mse_train: 0.0021247105871611523
ic| mse_test: 0.0020006968057079723
ic| mae_train: 0.04036377304821222
ic| mae_test: 0.03797408578655052
0.03797408578655052
```

plotting the best fit line

```
plt.xlabel('year')
plt.ylabel('pop(millions)')
plt.title('sg pop ')
```

```
plt.plot(year,population, label='Data')
plt.plot(train_year, train_pop, label='best fit
plt.plot(test_year, test_pop,'g+', label='future
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9672812230>
```



prediccting population and scaling according to millions

```
lrp.predict([[2030],[2050]])*scaling_factor
```

```
array([[6085177.04360004],
       [7417061.20874436]])
```

same steps for india

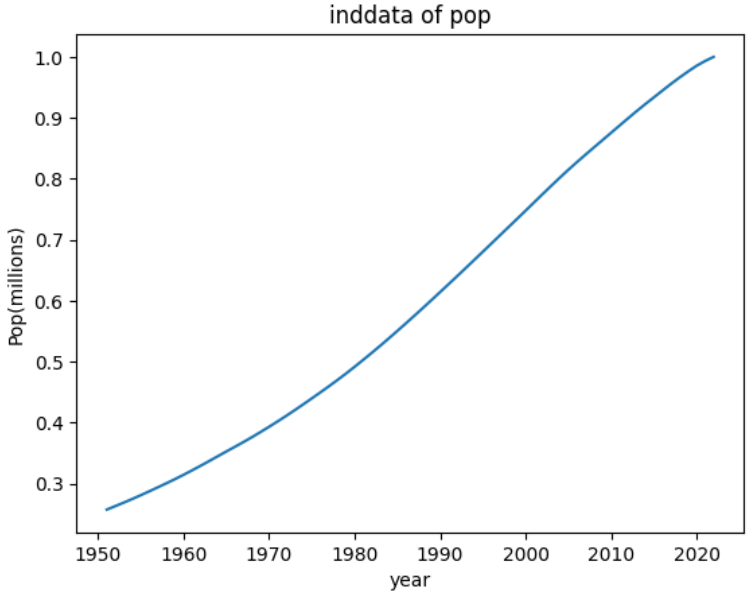
```
#for india

df1=pd.read_csv('indiadata1950-2022.csv')
```

```
year1=df1.iloc[1:,0].values.astype('int')
population1=df1.iloc[1:,1].values.astype('int')
#first is for rows and second is columns
scaling_factor1=max(population1)
population1=population1/max(population1)
print(year1)
print(population1)
```

```
[1951 1952 1953 1954 1955 1956 1957 1958 1959
 1965 1966 1967 1968 1969 1970 1971 1972 1973
 1979 1980 1981 1982 1983 1984 1985 1986 1987
 1993 1994 1995 1996 1997 1998 1999 2000 2001
 2007 2008 2009 2010 2011 2012 2013 2014 2015
 2021 2022]
[0.25750019 0.26319803 0.26900573 0.2750062  (
 0.29420215 0.30080711 0.30758439 0.31467896 (
 0.33724433 0.34509495 0.35289572 0.36057175 (
 0.38479043 0.39338968 0.40220856 0.411268   (
 0.43997744 0.44980491 0.45984897 0.47013856 (
 0.50302201 0.5145239  0.52627764 0.53832176 (
 0.57559382 0.58830473 0.60120576 0.61421722 (
 0.65366133 0.66700526 0.68042435 0.69383279 (
 0.73420812 0.74770938 0.76135431 0.7750027  (
 0.81474779 0.82726219 0.83948231 0.85150836 (
 0.88741532 0.8993165  0.9110616  0.92243244 (
 0.95556119 0.96600989 0.97596545 0.98533274 (
```

```
1 plt.xlabel('year')
2 plt.ylabel('Pop(millions)')
3 plt.title('inddata of pop')
4 plt.plot(year1,population1)
```

```
[<matplotlib.lines.Line2D at 0x7f9672a085b0>]
```




```
1 test_year1=year[-3:].reshape(-1,1)
2 print(test_year1)
```

```
[[2019]
 [2020]
```

```
    [2021]]
```

```
test_pop1=population[-3:].reshape(-1,1)
print(test_pop1)
```

```
   [[1.        ]
    [0.99688581]
    [0.95616727]]
```

```
train_pop1, train_year1 = population[:-3].reshap
print(train_year1)
```

```
    [1961]
    [1962]
    [1963]
    [1964]
    [1965]
    [1966]
    [1967]
    [1968]
    [1969]
    [1970]
    [1971]
    [1972]
    [1973]
    [1974]
    [1975]
    [1976]
    [1977]
    [1978]
    [1979]
    [1980]
    [1981]
    [1982]
    [1983]
    [1984]
    [1985]
    [1986]
    [1987]
    [1988]
    [1989]
    [1990]
    [1991]
    [1992]
    [1993]
    [1994]
    [1995]
```

```
[2007]
[2008]
[2009]
[2010]
[2011]
[2012]
[2013]
[2014]
[2015]
[2016]
[2017]
[2018]]
```

```
lrp1=lm.LinearRegression()
lrp1.fit(train_year1, train_pop1)
slope1=lrp.coef_[0]
intercept1=lrp1.intercept_
print("y="+str(slope1)+'x'+str(intercept1))
```

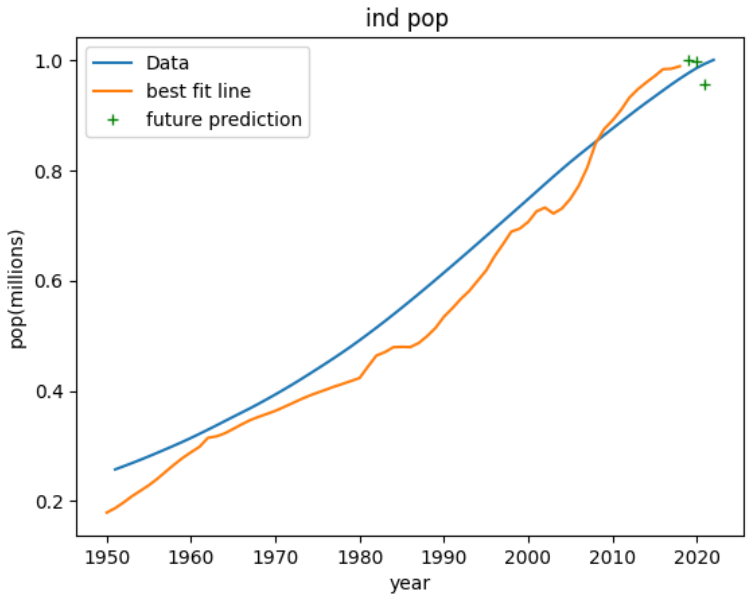
```
y=[0.01167588]x[-22.635137]
```

```
train_pred1=lrp1.predict(train_year1)
mse_train1=mean_squared_error(train_pred1, train
mae_train1=mean_absolute_error(train_pred1, trai

test_pred1=lrp1.predict(test_year1)
mse_test1=mean_squared_error(test_pred1,test_pop
mae_test1=mean_absolute_error(test_pred1, test_p

ic(mse_train1)
ic(mse_test1)
ic(mae_train1)
ic(mae_test1)
```

```
ic| mse_train1: 0.0021247105871611523
ic| mse_test1: 0.0020006968057079723
ic| mae_train1: 0.04036377304821222
ic| mae_test1: 0.03797408578655052
0.03797408578655052
```

```
plt.xlabel('year')
plt.ylabel('pop(millions)')
plt.title('ind pop ')
plt.plot(year1,population1, label='Data')
plt.plot(train_year1, train_pop1, label='best fi
plt.plot(test_year1, test_pop1,'g+', label='futu
plt.legend()

```

```
<matplotlib.legend.Legend at 0x7f967286b9d0>
```




```
1 lrp1.predict([[2030],[2050]])*scaling_factor1
```

```
array([[1.51199182e+09],
       [1.84292680e+09]])
```

✓ 0s completed at 09:06