1. Year-Extent Code

```
import pandas as pd
import numpy as np
import glob
import matplotlib.pyplot as plt
# Import all the monthly data and make a yearly mean extent data
path="../../documents/Machine Learning/Project/Monthly Arctic Ice Extent Data/"
files=glob.glob(path+"*.csv")
files.sort()
all_ice_extent=[]
monthly_ice_extent_info=[[] for i in range(12)]
count=0
for file in files:
    monthly_ice_extent_info[count]=pd.read_csv(file)
    all_ice_extent.append(monthly_ice_extent_info[count])
    count=count+1
all_ice_extent_concat = pd.concat(all_ice_extent, axis=0, ignore_index=True, sort=False)
yearly_ice_extent_info=pd.pivot_table(all_ice_extent_concat,index="year",values="
extent")
yearly_ice_extent_info.to_csv("../../documents/Machine
Learning/Project/yearly_ice_extent_info.csv")    #To export the year-extent table
yearly_ice_extent_info.head(100)
```

	extent
year	
1978	12.660000
1979	12.350000
1980	12.348333
1981	12.146667
1982	12.467500
1983	12.353333
1984	11.920000
1985	12.015833
1986	12.224167
1987	12.162727
1988	11.938182
1989	11.986667
1990	11.717500
1991	11.770000
1992	12.121667
1993	11.946667
1994	12.032500
1995	11.438333
1996	11.848333
1997	11.691667
1998	11.781667
1999	11.713333
2000	11.519167
2001	11.622500
2002	11.385000
2003	11.419167
2004	11.250833
2005	10.927500
2006	10.794167
2007	10.498333
2008	10.990000
2009	10.955833
2010	10.734167
2011	10.505833
2012	10.420000
2013	10.919167
2014	10.812500
2015	10.589167
2016	10.175833
2017	10.415000
2018	10.378333
2019	10.052727

```
# Building a 10th-order polynomial model to fit the monthly data in 2018
all_ice_extent_concat_sort=all_ice_extent_concat.sort_values(by=["year","
mo"],ascending=True)
x=range(1,13)
monthly_ice_extent_info_2018=all_ice_extent_concat_sort.loc[(all_ice_extent_concat_so
rt["year"]==2018)]
y=monthly_ice_extent_info_2018[" extent"]
degree1=10
coef1=np.polyfit(x,y,degree1)
print('Coefficients: %s' % coef1)
curve=[]
for i in range(12):
    value=coef1[-1]
    for d in range(degree1):
        value =value+x[i]**(degree1-d) * coef1[d]
    curve.append(value)
plt.figure(1)
plt.title("2018 Original Month Data")
plt.plot(x,y)
plt.xlabel("Month")
plt.ylabel("Extent")
plt.savefig("2018 Original Month Data.png")
plt.figure(2)
plt.plot(x,curve, color='red', linewidth=3)
plt.title("Model")
plt.xlabel("Month")
plt.ylabel("Extent")
plt.savefig("2018 Original Month Data Model.png")
plt.show()
```

```
Coefficients: [-7.62924383e-06  4.93094595e-04 -1.38059
896e-02  2.19446010e-01
 -2.18080859e+00  1.40697262e+01 -5.92508913e+01  1.59
484431e+02
 -2.60368913e+02  2.31004555e+02 -6.98841667e+01]
```

```
# Another way to get the polynomial model
p1= np.poly1d(coef1)
yvals = p1(x)
plt.plot(x,y)
plt.plot(x,yvals,'r',label='polyfit values')
```

```
plt.show()

# Building a 18th-order polynomial model to fit the monthly data between 2014-2018(not referred in the paper)
x=range(1,61)
year_xticks=[]
for year in range(2014,2019):
    for month in range(12):
        if month==0:
            year_xticks.append(str(year))
        else:
            year_xticks.append("")
monthly_ice_extent_info_last_5_years=all_ice_extent_concat_sort.loc[(all_ice_extent_concat_sort["year"]<=2018)&(all_ice_extent_concat_sort["year"]>=2014)]
y=monthly_ice_extent_info_last_5_years[" extent"]
degree2=18
coef2=np.polyfit(x,y,degree2)
print('Coefficients: %s' % coef2)
curve=[]
for i in range(len(x)):
    value=coef2[-1]
    for d in range(degree2):
        value =value+x[i]**(degree2-d) * coef2[d]
    curve.append(value)
plt.figure(1)
plt.title("2014-2018 Original Month Data")
plt.plot(x,y)
plt.xticks(range(len(x)),year_xticks,color='blue')
plt.xlabel("Year")
plt.ylabel("Extent")
plt.savefig("2014-2018 Original Month Data.png")
plt.figure(2)
plt.plot(x,curve, color='red', linewidth=3)
plt.xticks(range(len(x)),year_xticks,color='blue')
plt.xlabel("Year")
plt.ylabel("Extent")
plt.title("Model")
plt.savefig("2014-2018 Original Month Data Model.png")
plt.show()
```

```
Coefficients: [-4.22260881e-23  1.68206838e-20 -2.628126
45e-18  1.57740777e-16
```

8.53155467e-15 -2.43094702e-12 2.28086919e-10 -1.308

55751e-08

5.13952817e-07 -1.43522525e-05 2.88276494e-04 -4.16

029714e-03

4.28465189e-02 -3.12901836e-01 1.61379438e+00 -5.740

47838e+00

1.26512510e+01 -1.39385812e+01 1.93740278e+01]

```
# Ouput of the extent data in different seasons between 1978-2019
seasonal_ice_extent_info1=pd.DataFrame(columns=("year","season","extent"))
seasonal_ice_extent_info2=pd.DataFrame(columns=("year","Spring","Summer","Autum
n","Winter"))
seasonal_matrix=np.zeros((len(range(1978,2020)),4))
for year in range(1978,2020):
    for season_count in range(0,4):
        if season_count==0:
            season="Spring"
        elif season_count==1:
            season="Summer"
        elif season_count==2:
            season="Autumn"
        else:
            season="Winter"

seasonal_ice_extent=all_ice_extent_concat_sort.loc[(all_ice_extent_concat_sort["year"]=
=year)&(all_ice_extent_concat_sort["mo"]>3*season_count)&(all_ice_extent_concat_sort[
" mo"]<=3*(season_count+1))]
seasonal_ice_extent_info1=seasonal_ice_extent_info1.append({"year":year,"season":sea
son,"extent":seasonal_ice_extent[" extent"].mean()},ignore_index=True)
seasonal_matrix[year-1978][season_count]=seasonal_ice_extent[" extent"].mean()
seasonal_ice_extent_info2=seasonal_ice_extent_info2.append({"year":str(year),"Spring":
seasonal_matrix[year-1978][0],"Summer":seasonal_matrix[year-1978][1],"Autumn":seaso
nal_matrix[year-1978][2],"Winter":seasonal_matrix[year-1978][3]},ignore_index=True)
seasonal_ice_extent_info2.head(50)
```

	year	Spring	Summer	Autumn	Winter
0	1978	NaN	NaN	NaN	12.660000
1	1979	15.976667	13.946667	8.466667	11.010000
2	1980	15.620000	13.806667	8.583333	11.383333
3	1981	15.380000	13.746667	8.416667	11.043333
4	1982	15.730000	13.973333	8.603333	11.563333
5	1983	15.680000	13.653333	8.716667	11.363333
6	1984	15.116667	13.580000	8.186667	10.796667
7	1985	15.360000	13.883333	7.946667	10.873333
8	1986	15.526667	13.473333	8.513333	11.383333
9	1987	15.613333	13.813333	8.413333	10.135000
10	1988	15.770000	13.540000	8.356667	11.363333
11	1989	15.290000	13.203333	8.340000	11.113333
12	1990	15.410000	13.173333	7.396667	10.890000
13	1991	15.010000	13.480000	7.800000	10.790000
14	1992	15.193333	13.350000	8.573333	11.370000
15	1993	15.466667	13.440000	7.736667	11.143333
16	1994	15.280000	13.510000	8.236667	11.103333
17	1995	15.026667	12.953333	7.270000	10.503333
18	1996	14.816667	13.130000	8.640000	10.806667
19	1997	15.110000	13.160000	7.796667	10.700000
20	1998	15.356667	13.396667	7.823333	10.550000
21	1999	15.023333	13.536667	7.613333	10.680000
22	2000	14.860000	13.126667	7.643333	10.446667
23	2001	14.976667	13.276667	7.753333	10.483333
24	2002	14.986667	12.950000	7.233333	10.370000
25	2003	15.020000	13.043333	7.423333	10.190000
26	2004	14.643333	12.666667	7.420000	10.273333
27	2005	14.240000	12.720000	6.816667	9.933333

28	2006	14.070000	12.450000	6.940000	9.716667
29	2007	14.250000	12.616667	5.850000	9.276667
30	2008	14.673333	12.843333	6.426667	10.016667
31	2009	14.566667	13.003333	6.623333	9.630000
32	2010	14.486667	12.706667	6.270000	9.473333
33	2011	14.123333	12.513333	5.926667	9.460000
34	2012	14.493333	12.770000	5.320000	9.096667
35	2013	14.483333	12.886667	6.450000	9.856667
36	2014	14.276667	12.606667	6.470000	9.896667
37	2015	14.123333	12.413333	6.200000	9.620000
38	2016	14.020000	12.003333	5.946667	8.733333
39	2017	13.866667	12.380000	6.080000	9.333333
40	2018	13.783333	12.236667	6.223333	9.270000
41	2019	14.170000	12.046667	5.646667	7.495000

```
# Plotting the figure with the seasonal data(not shown in the paper)
plt.figure(figsize=(50,5))
year_season=[]
for year in range(1978,2020):
    for season_count in range(4):
        if season_count==0:
            year_season.append(str(year)+"-Spring")
        elif season_count==1:
            year_season.append(str(year)+"-Summer")
        elif season_count==2:
            year_season.append(str(year)+"-Autumn")
        else:
            year_season.append(str(year)+"-Winter")
plt.plot(range(len(seasonal_ice_extent_info1["extent"])),seasonal_ice_extent_info1["exten
t"])
plt.xticks(range(len(seasonal_ice_extent_info1["extent"])),year_season,color='blue',rotati
on=60)
plt.title("Seasonality")
plt.xlabel("Season")
plt.ylabel("Extent")
```

```
plt.savefig("Seasonal data.png",bbox_inches='tight')
plt.show()

# Method 1: using Seaborn library to plot the linear regression model
import seaborn as sns
%matplotlib inline
plt.plot(range(1978,2020),yearly_ice_extent_info[" extent"])
sns.regplot(x=np.arange(1978,2020),y=" extent",data=yearly_ice_extent_info)
plt.xlabel('Year')
plt.ylabel('Extent')
plt.title('Yearly Arctic Ice Extent')
plt.savefig("Yearly Arctic Ice Extent.png")
plt.show()

# Plotting the heatmap of monthly extent data
import seaborn as sns
ice_extent=all_ice_extent_concat.pivot(" mo","year"," extent")
ax=sns.heatmap(ice_extent,cmap="YlGnBu_r",vmin=7.5,vmax=15)
ax.set_title("Monthly Arctic Ice Extent")
fig=ax.get_figure()
fig.savefig("Monthly Arctic Ice Extent.png")
ax.set_ylabel("month")

# Method 2: Least Squares Estimation
theta1=(len(range(1978,2020))*np.dot(range(1978,2020),yearly_ice_extent_info["
extent"])-(np.arange(1978,2020).sum())*yearly_ice_extent_info["
extent"].sum())/(len(range(1978,2020))*((np.arange(1978,2020)**2).sum())-(np.arange(1978,
2020).sum())**2)
print("theta1="+str(theta1))
theta0=yearly_ice_extent_info[" extent"].mean()-theta1*(np.arange(1978,2020).mean())
print("theta0="+str(theta0))
```

theta1=−0.05572477059714807

theta0=122.81834220578858

```
plt.plot(range(1978,2020),yearly_ice_extent_info[" extent"])
plt.plot(range(1978,2020),theta1*range(1978,2020)+theta0)
plt.scatter(range(1978,2020),yearly_ice_extent_info[" extent"])
plt.title("Least Square Estimation")
plt.xlabel("Year")
plt.ylabel("Extent")
plt.savefig("Least Square Estimation.png")
```

```
plt.show()

# Method 3: Gradient Descent
iters=10000 #Iterating Time
alpha=0.001    #Learning Rate
theta0=1
theta1=-1
loss_value=[]
m=len(range(1978,2020))
def normalization(X):      #Normalization to condense the data into the range 0 to 1.
    minVal=X.min()
    maxVal=X.max()
    diff=maxVal-minVal
    if diff != 0:
        X = (X-minVal)/diff
    else:
        X=0
    return X,diff,minVal
[X,diffx,minValx]=normalization(np.arange(1978,2020))
[Y,diffy,minValy]=normalization(yearly_ice_extent_info[" extent"])
for i in range(iters):
    error=theta1*X+theta0-Y
    cost=np.power(error,2).sum()/m
    loss_value.append(cost)
    theta0=theta0-(alpha*error.sum()/m)
    theta1=theta1-alpha*(((error*X).sum()/m))

# Revertion of parameters and plotting figures of linear regression model.
theta1=theta1*diffy/diffx
theta0=theta0*diffy+minValy-theta1*minValx
print("theta1="+str(theta1))
print("theta0="+str(theta0))
plt.figure(1)
plt.plot(range(iters),loss_value)
plt.title("Loss Value")
plt.xlabel("Iteration Time")
plt.ylabel("Loss Value")
plt.savefig("Loss Value.png")
plt.figure(2)
plt.plot(np.arange(1978,2020),theta1*np.arange(1978,2020)+theta0)
plt.scatter(np.arange(1978,2020),yearly_ice_extent_info[" extent"])
plt.title("Gradient Descent")
plt.xlabel("Year")
plt.ylabel("Extent")
```

```
plt.savefig("Gradient Descent.png")
plt.show()
```

theta1=−0.05912366314200111

theta0=129.61618456222237

```
# Prediction of the first year with ice-free month and the first ice-free year
import sympy
from sympy.abc import x
import math
diff1=monthly_ice_extent_info_2018["      extent"].mean()-monthly_ice_extent_info_2018["
extent"].min()
diff2=monthly_ice_extent_info_2018["        extent"].max()-monthly_ice_extent_info_2018["
extent"].mean()
predicted_ice_free_year1=sympy.solve(theta1*x+theta0-diff1,x)
predicted_ice_free_year1=math.ceil(predicted_ice_free_year1[0])
print(predicted_ice_free_year1)
predicted_ice_free_year2=sympy.solve(theta1*x+theta0+diff2,x)
predicted_ice_free_year2=math.ceil(predicted_ice_free_year2[0])
print(predicted_ice_free_year2)
```

2098

2259

```
# Plotting the monthly data in 2098 and 2259
curve1=[]
curve2=[]
x=range(1,13)
for i in range(12):
    value1=coef1[-1]
    value2=coef1[-1]
    for d in range(degree1):
        value1=value1+x[i]**(degree1-d) * coef1[d]
        value2=value2+x[i]**(degree1-d) * coef1[d]
    curve1.append(value1+theta1*(predicted_ice_free_year1-2018))
    curve2.append(value2+theta1*(predicted_ice_free_year2-2018))
plt.figure(1)
plt.plot(x,curve1)
plt.xlabel("Month")
plt.ylabel("Extent")
plt.title("First Year With Ice-Free Period-2098")
plt.savefig("First Year With Ice-Free Period-2098.png")
```

```
plt.figure(2)
plt.plot(x,curve2)
plt.xlabel("Month")
plt.ylabel("Extent")
plt.title("Ice Free Year")
plt.savefig("Ice Free Year.png")

#Plotting the predicted monthly data between 2019-2023
curve3=[]
x=range(1,61)
year_xticks=[]
for year in range(2019,2024):
    for month in range(12):
        if month==0:
            year_xticks.append(str(year))
        else:
            year_xticks.append("")
for i in range(1,6):
    for j in range(12):
        value3=coef1[-1]
        for d in range(degree1):
            value3=value3+x[j]**(degree1-d) * coef1[d]
        curve3.append(value3+theta1*i)
plt.xticks(range(len(x)),year_xticks,color='blue')
plt.plot(x,curve3)
plt.xlabel("Year")
plt.ylabel("Extent")
plt.title("Predicted Arctic Ice extent from 2019 to 2023")
plt.savefig("Predicted Arctic Ice extent from 2019 to 2023.png")
```

2. Temperature-Extent Code

```
# Original table for temperature between 1978 and 2019
import pandas as pd
import numpy as np
temperature_info=pd.read_csv("../../documents/Machine Learning/Project/Temperature Data/Monthly/GLB.Ts+dSST.csv",header=1)
temperature_info.loc[98:139].head(100)
```

	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	J-D	D-N	DJF	MAM	JJA	SON
98	1978	0.06	0.10	0.19	0.17	0.09	-0.01	0.04	-0.13	0.06	0.03	0.14	0.08	0.07	0.06	0.06	0.15	-0.04	0.08
99	1979	0.08	-0.10	0.19	0.15	0.03	0.14	0.04	0.17	0.25	0.25	0.29	0.48	0.16	0.13	0.02	0.12	0.11	0.26
100	1980	0.30	0.40	0.30	0.30	0.35	0.20	0.22	0.18	0.20	0.13	0.29	0.21	0.26	0.28	0.39	0.32	0.20	0.21
101	1981	0.52	0.42	0.48	0.32	0.24	0.29	0.32	0.35	0.15	0.12	0.23	0.41	0.32	0.30	0.39	0.35	0.32	0.16
102	1982	0.05	0.15	0.03	0.15	0.18	0.06	0.14	0.03	0.14	0.13	0.17	0.42	0.14	0.14	0.20	0.12	0.08	0.15
103	1983	0.53	0.43	0.41	0.27	0.33	0.22	0.18	0.35	0.37	0.16	0.30	0.17	0.31	0.33	0.46	0.34	0.25	0.28
104	1984	0.31	0.14	0.26	0.06	0.32	0.02	0.19	0.19	0.21	0.13	0.07	-0.04	0.15	0.17	0.21	0.21	0.13	0.14
105	1985	0.22	-0.04	0.17	0.12	0.14	0.15	0.04	0.16	0.13	0.11	0.05	0.13	0.11	0.10	0.05	0.15	0.12	0.10
106	1986	0.27	0.37	0.30	0.22	0.21	0.12	0.11	0.15	0.03	0.15	0.11	0.13	0.18	0.18	0.26	0.24	0.13	0.09
107	1987	0.32	0.43	0.18	0.24	0.25	0.34	0.40	0.24	0.35	0.32	0.29	0.46	0.32	0.29	0.30	0.23	0.33	0.32
108	1988	0.57	0.44	0.51	0.42	0.43	0.39	0.32	0.38	0.36	0.37	0.12	0.28	0.38	0.40	0.49	0.46	0.37	0.28
109	1989	0.12	0.30	0.36	0.29	0.17	0.16	0.33	0.33	0.35	0.28	0.19	0.37	0.27	0.26	0.23	0.27	0.28	0.27
110	1990	0.41	0.43	0.79	0.56	0.45	0.39	0.46	0.34	0.23	0.44	0.47	0.40	0.45	0.45	0.40	0.60	0.40	0.38
111	1991	0.42	0.50	0.35	0.51	0.34	0.53	0.47	0.40	0.44	0.28	0.29	0.31	0.40	0.41	0.44	0.40	0.46	0.34
112	1992	0.47	0.41	0.47	0.27	0.30	0.26	0.08	0.08	-0.01	0.06	0.02	0.21	0.22	0.23	0.40	0.35	0.14	0.03
113	1993	0.34	0.37	0.36	0.27	0.28	0.23	0.25	0.11	0.12	0.23	0.03	0.18	0.23	0.23	0.31	0.30	0.19	0.13
114	1994	0.26	0.02	0.30	0.40	0.27	0.44	0.30	0.22	0.32	0.41	0.44	0.38	0.31	0.30	0.15	0.33	0.32	0.39
115	1995	0.52	0.79	0.47	0.46	0.28	0.42	0.45	0.46	0.34	0.47	0.44	0.25	0.45	0.46	0.56	0.40	0.44	0.42
116	1996	0.23	0.47	0.33	0.32	0.28	0.25	0.36	0.48	0.25	0.20	0.38	0.37	0.33	0.32	0.32	0.31	0.36	0.28
117	1997	0.30	0.41	0.52	0.35	0.36	0.54	0.34	0.43	0.52	0.60	0.64	0.58	0.46	0.45	0.36	0.41	0.43	0.59
118	1998	0.59	0.88	0.63	0.64	0.66	0.76	0.68	0.66	0.42	0.43	0.43	0.56	0.61	0.61	0.68	0.65	0.70	0.43
119	1999	0.48	0.65	0.32	0.33	0.27	0.36	0.38	0.32	0.38	0.34	0.37	0.41	0.39	0.40	0.56	0.31	0.35	0.37
120	2000	0.25	0.56	0.55	0.57	0.35	0.39	0.37	0.42	0.40	0.27	0.31	0.28	0.39	0.40	0.41	0.49	0.40	0.32
121	2001	0.46	0.44	0.56	0.51	0.58	0.52	0.59	0.50	0.52	0.51	0.73	0.56	0.54	0.52	0.39	0.55	0.54	0.58
122	2002	0.77	0.79	0.88	0.58	0.63	0.53	0.60	0.53	0.63	0.54	0.59	0.43	0.63	0.64	0.71	0.70	0.55	0.59
123	2003	0.75	0.59	0.60	0.55	0.61	0.48	0.58	0.65	0.62	0.74	0.53	0.75	0.62	0.59	0.59	0.59	0.57	0.63
124	2004	0.58	0.73	0.63	0.62	0.39	0.44	0.26	0.47	0.51	0.61	0.73	0.51	0.54	0.56	0.69	0.55	0.39	0.62
125	2005	0.74	0.61	0.74	0.68	0.63	0.65	0.62	0.62	0.72	0.75	0.74	0.68	0.68	0.67	0.62	0.69	0.63	0.73

126	2006	0.56	0.73	0.63	0.48	0.50	0.66	0.55	0.71	0.65	0.69	0.73	0.79	0.64	0.63	0.66	0.54	0.64	0.69
127	2007	1.01	0.70	0.72	0.75	0.68	0.61	0.60	0.60	0.60	0.59	0.58	0.49	0.66	0.69	0.84	0.72	0.60	0.59
128	2008	0.30	0.38	0.74	0.53	0.49	0.49	0.60	0.47	0.61	0.65	0.69	0.54	0.54	0.54	0.39	0.59	0.52	0.65
129	2009	0.65	0.54	0.54	0.60	0.66	0.65	0.71	0.68	0.73	0.66	0.80	0.67	0.66	0.65	0.58	0.60	0.68	0.73
130	2010	0.75	0.84	0.92	0.84	0.76	0.68	0.64	0.66	0.63	0.70	0.82	0.45	0.73	0.74	0.75	0.84	0.66	0.72
131	2011	0.52	0.48	0.65	0.65	0.52	0.61	0.70	0.73	0.58	0.66	0.59	0.60	0.61	0.59	0.48	0.60	0.68	0.61
132	2012	0.49	0.49	0.57	0.71	0.77	0.65	0.58	0.64	0.71	0.79	0.79	0.53	0.64	0.65	0.53	0.69	0.62	0.76
133	2013	0.71	0.63	0.67	0.56	0.62	0.70	0.61	0.70	0.77	0.69	0.85	0.70	0.68	0.67	0.62	0.62	0.67	0.77
134	2014	0.76	0.55	0.79	0.81	0.85	0.67	0.58	0.80	0.84	0.79	0.66	0.80	0.74	0.73	0.67	0.82	0.68	0.77
135	2015	0.86	0.89	0.96	0.77	0.79	0.81	0.74	0.82	0.84	1.08	1.06	1.16	0.90	0.87	0.85	0.84	0.79	0.99
136	2016	1.17	1.37	1.36	1.12	0.95	0.81	0.84	1.01	0.91	0.87	0.90	0.85	1.01	1.04	1.23	1.14	0.89	0.90
137	2017	1.03	1.14	1.16	0.93	0.90	0.72	0.82	0.86	0.79	0.90	0.88	0.94	0.92	0.91	1.01	1.00	0.80	0.86
138	2018	0.82	0.85	0.90	0.89	0.82	0.78	0.82	0.76	0.80	1.00	0.82	0.91	0.85	0.85	0.87	0.87	0.79	0.88
139	2019	0.93	0.95	1.18	1.02	0.86	0.92	0.94	0.93	0.92	1.02	1.02	NaN	NaN	0.97	0.93	1.02	0.93	0.99

```
# Making a new table including year, temperature and extent
useful_temperature_info=temperature_info[["Year","J-D"]][98:140]
useful_temperature_info=useful_temperature_info.reset_index(drop=True)
mean_temperature_2019=temperature_info[["Jan","Feb","Mar","Apr","May","Jun","Jul",
"Aug","Sep","Oct","Nov","Dec"]][139:140].mean(1)
useful_temperature_info["J-D"][41]=mean_temperature_2019
useful_temperature_info.rename(columns={"J-D":"Temperature"},inplace=True)
yearly_ice_extent_info=pd.read_csv("../../documents/Machine
Learning/Project/yearly_ice_extent_info.csv")
extent=yearly_ice_extent_info[" extent"]
useful_temperature_info["Extent"]=extent
useful_temperature_info.head(50)
```

	Year	Temperature	Extent
0	1978	0.070000	12.660000
1	1979	0.160000	12.350000
2	1980	0.260000	12.348333
3	1981	0.320000	12.146667
4	1982	0.140000	12.467500
5	1983	0.310000	12.353333
6	1984	0.150000	11.920000
7	1985	0.110000	12.015833
8	1986	0.180000	12.224167
9	1987	0.320000	12.162727
10	1988	0.380000	11.938182
11	1989	0.270000	11.986667
12	1990	0.450000	11.717500
13	1991	0.400000	11.770000
14	1992	0.220000	12.121667
15	1993	0.230000	11.946667
16	1994	0.310000	12.032500
17	1995	0.450000	11.438333
18	1996	0.330000	11.848333
19	1997	0.460000	11.691667
20	1998	0.610000	11.781667
21	1999	0.390000	11.713333
22	2000	0.390000	11.519167
23	2001	0.540000	11.622500
24	2002	0.630000	11.385000
25	2003	0.620000	11.419167
26	2004	0.540000	11.250833
27	2005	0.680000	10.927500
28	2006	0.640000	10.794167
29	2007	0.660000	10.498333
30	2008	0.540000	10.990000
31	2009	0.660000	10.955833
32	2010	0.730000	10.734167
33	2011	0.610000	10.505833
34	2012	0.640000	10.420000
35	2013	0.680000	10.919167
36	2014	0.740000	10.812500
37	2015	0.900000	10.589167
38	2016	1.010000	10.175833
39	2017	0.920000	10.415000
40	2018	0.850000	10.378333
41	2019	0.971818	10.052727

```
# Method 1: using seaborn library to plot the linear regression model
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.regplot(x="Temperature",y="Extent",data=useful_temperature_info)
plt.title("Global Temperature-Arctic Ice Extent")
plt.savefig("Global Temperature-Arctic Ice Extent.png")
plt.show()

# Plotting 3-dimensional figure for year-temperature-extent
from mpl_toolkits import mplot3d
ax = plt.axes(projection='3d')
ax.plot3D(useful_temperature_info["Year"],useful_temperature_info["Temperature"],useful_temperature_info["Extent"])
ax.set_xlabel("Year")
ax.set_ylabel("Temperature")
ax.set_zlabel("Extent")
ax.set_title("Year-Temperature-Extent")
plt.savefig("Year-Temperature-Extent.png")
plt.show()

# Method 2: Least Squares Estimation
m=len(range(1978,2020))
theta1=(m*(useful_temperature_info["Temperature"]*useful_temperature_info["Extent"]).sum()-useful_temperature_info["Temperature"].sum()*useful_temperature_info["Extent"].sum())/(m*(useful_temperature_info["Temperature"]**2).sum()-(useful_temperature_info["Temperature"].sum())**2)
theta0=useful_temperature_info["Extent"].mean()-theta1*useful_temperature_info["Temperature"].mean()
print("theta1="+str(theta1))
print("theta0="+str(theta0))
```

theta1=−2.6560725497955553

theta0=12.747022317796091

```
plt.plot(useful_temperature_info["Temperature"],theta1*useful_temperature_info["Temperature"]+theta0)
plt.scatter(useful_temperature_info["Temperature"],useful_temperature_info["Extent"])
plt.title("Least Squares Estimation")
plt.xlabel("Temperature")
plt.ylabel("Extent")
plt.savefig("Least Squares Estimation.png")
plt.show()
```

```
# Method 3: Gradient Descent
iters=100000 #iterating time
alpha=0.001    #Learning Rate
theta0=-1
theta1=10
loss_value=[]
for i in range(iters):

error=theta1*useful_temperature_info["Temperature"]+theta0-useful_temperature_info["
Extent"]
    cost=np.power(error,2).sum()/m
    loss_value.append(cost)
    theta0=theta0-(alpha*error.sum()/m)
    theta1=theta1-alpha*(((error*useful_temperature_info["Temperature"]).sum()/m))
print("theta1="+str(theta1))
print("theta0="+str(theta0))
```

theta1=−2.5351624908979553

theta0=12.685077241529278

```
plt.figure(1)
plt.plot(useful_temperature_info["Temperature"],theta1*useful_temperature_info["Temper
ature"]+theta0)
plt.scatter(useful_temperature_info["Temperature"],useful_temperature_info["Extent"])
plt.title("Gradient Descent")
plt.xlabel("Temperature")
plt.ylabel("Extent")
plt.savefig("Gradient Descent.png")
plt.figure(2)
plt.plot(range(iters),loss_value)
plt.title("Loss Value")
plt.xlabel("Iteration Time")
plt.ylabel("Loss Value")
plt.savefig("Loss Value.png")
plt.show()
```

3. CO_2 Concentration-Extent Code

```
# Making a new table including year, CO2 concentration and extent
import pandas as pd
import numpy as np
co2_concentration_info=pd.read_csv("../../documents/Machine Learning/Project/CO2 Concentration Data/Yearly/co2_annmean_mlo.txt",sep='\s+')
useful_co2_concentration_info=co2_concentration_info[["year","mean"]][19:60]
useful_co2_concentration_info=useful_co2_concentration_info.reset_index(drop=True)

monthly_co2_concentration_info=pd.read_csv("../../documents/Machine Learning/Project/CO2 Concentration Data/Monthly/co2_mm_mlo.txt",sep='\s+')
monthly_co2_concentration_info.head(1000)
monthly_co2_concentration_info_2019=monthly_co2_concentration_info.loc[monthly_co2_concentration_info["year"]==2019]
co2_concentration_annual_mean_2019=monthly_co2_concentration_info_2019["average"].mean()
ind=len(useful_co2_concentration_info)
useful_co2_concentration_info.loc[ind]=[2019,co2_concentration_annual_mean_2019]

yearly_ice_extent_info=pd.read_csv("../../documents/Machine Learning/Project/yearly_ice_extent_info.csv")
extent=yearly_ice_extent_info[" extent"]
useful_co2_concentration_info["extent"]=extent
useful_co2_concentration_info.rename(columns={"mean":"CO2 concentration"},inplace=True)
year=useful_co2_concentration_info["year"].astype("int")
useful_co2_concentration_info["year"]=year
useful_co2_concentration_info.head(50)
```

	year	CO2 concentration	extent
0	1978	335.40	12.660000
1	1979	336.84	12.350000
2	1980	338.75	12.348333
3	1981	340.11	12.146667
4	1982	341.45	12.467500
5	1983	343.05	12.353333
6	1984	344.65	11.920000
7	1985	346.12	12.015833
8	1986	347.42	12.224167
9	1987	349.19	12.162727
10	1988	351.57	11.938182
11	1989	353.12	11.986667
12	1990	354.39	11.717500
13	1991	355.61	11.770000
14	1992	356.45	12.121667
15	1993	357.10	11.946667
16	1994	358.83	12.032500
17	1995	360.82	11.438333
18	1996	362.61	11.848333
19	1997	363.73	11.691667
20	1998	366.70	11.781667
21	1999	368.38	11.713333
22	2000	369.55	11.519167
23	2001	371.14	11.622500
24	2002	373.28	11.385000
25	2003	375.80	11.419167
26	2004	377.52	11.250833
27	2005	379.80	10.927500
28	2006	381.90	10.794167

29	2007	383.79	10.498333
30	2008	385.60	10.990000
31	2009	387.43	10.955833
32	2010	389.90	10.734167
33	2011	391.65	10.505833
34	2012	393.85	10.420000
35	2013	396.52	10.919167
36	2014	398.65	10.812500
37	2015	400.83	10.589167
38	2016	404.24	10.175833
39	2017	406.55	10.415000
40	2018	408.52	10.378333
41	2019	411.41	10.052727

```
# Method 1: using seaborn library to plot the linear regression model
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.regplot(x="CO2 concentration",y="extent",data=useful_co2_concentration_info)
plt.title("Global CO2 concentration-Arctic Ice Extent")
plt.savefig("Global CO2 concentration-Arctic Ice Extent.png")
plt.show()

# Plotting 3-dimensional figure for year-CO2 Concentration-extent(not shown in the paper)
from mpl_toolkits import mplot3d
ax = plt.axes(projection='3d')
ax.plot3D(useful_co2_concentration_info["year"],useful_co2_concentration_info["CO2 concentration"],useful_co2_concentration_info["extent"])
ax.set_xlabel("Year")
ax.set_ylabel("CO2 concentration")
plt.yticks(rotation=-20)
ax.set_zlabel("Extent")
ax.set_title("Year-CO2 concentration-Extent")
plt.savefig("Year-CO2 concentration-Extent.png")
```

```
plt.show()
# Method 2: Least Squares Estimation
m=len(range(1978,2020))
theta1=(m*(useful_co2_concentration_info["CO2
concentration"]*useful_co2_concentration_info["extent"]).sum()-useful_co2_concentratio
n_info["CO2
concentration"].sum()*useful_co2_concentration_info["extent"].sum())/(m*(useful_co2_co
ncentration_info["CO2      concentration"]**2).sum()-(useful_co2_concentration_info["CO2
concentration"].sum())**2)
theta0=useful_co2_concentration_info["extent"].mean()-theta1*useful_co2_concentration
_info["CO2 concentration"].mean()
print("theta1="+str(theta1))
print("theta0="+str(theta0))
```

theta1=−0.03098790234076758

theta0=22.903318207084066

```
plt.plot(useful_co2_concentration_info["CO2
concentration"],theta1*useful_co2_concentration_info["CO2 concentration"]+theta0)
plt.scatter(useful_co2_concentration_info["CO2
concentration"],useful_co2_concentration_info["extent"])
plt.title("Least Squares Estimation")
plt.xlabel("CO2 concentration")
plt.ylabel("Extent")
plt.savefig("Least Squares Estimation.png")
plt.show()

# Method 3: Gradient Descent
iters=10000 #iterating time
alpha=0.01    #Learning Rate
theta0=1
theta1=1
loss_value=[]
def normalization(X):
    minVal=X.min()
    maxVal=X.max()
    diff=maxVal-minVal
    if diff != 0:
        X = (X-minVal)/diff
    else:
        X=0
    return X,diff,minVal
[X,diffx,minValx]=normalization(useful_co2_concentration_info["CO2 concentration"])
```

```
[Y,diffy,minValy]=normalization(useful_co2_concentration_info["extent"])
for i in range(iters):
        error=theta1*X+theta0-Y
        cost=np.power(error,2).sum()/m
        loss_value.append(cost)
        theta0=theta0-(alpha*error.sum()/m)
        theta1=theta1-alpha*(((error*X).sum()/m))

plt.figure(1)
plt.plot(X,theta1*X+theta0)
plt.scatter(X,Y)
plt.figure(2)
plt.plot(range(iters),loss_value)
plt.title("Loss Value")
plt.xlabel("Iteration Time")
plt.ylabel("Loss Value")
plt.savefig("Loss Value.png")
plt.show()

# Revertion of parameters and plotting figures of linear regression model.
theta1=theta1*diffy/diffx
theta0=theta0*diffy+minValy-theta1*minValx
print("theta1="+str(theta1))
print("theta0="+str(theta0))
plt.plot(useful_co2_concentration_info["CO2
concentration"],theta1*useful_co2_concentration_info["CO2 concentration"]+theta0)
plt.scatter(useful_co2_concentration_info["CO2
concentration"],useful_co2_concentration_info["extent"])
plt.title("Gradient Descent")
plt.xlabel("CO2 concentration")
plt.ylabel("Extent")
plt.savefig("Gradient Descent.png")
plt.show()
```

theta1=−0.030933992859181542

theta0=22.883261295455938