```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv("/content/1_fiat500_VehicleSelection_Dataset - 1_fiat500_VehicleSelection_Dataset (1).csv")
df
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9	Unnamed: 10
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900	NaN	NaN
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800	NaN	NaN
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200	NaN	NaN
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000	NaN	NaN
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5	NaN	NaN
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice	NaN	NaN
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NO	NaN	NaN
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1	NaN	NaN
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1	NaN	NaN

1549 rows × 11 columns

```
df1=df.drop(df.index[1537:],axis=0)
df1=df1.drop(["Unnamed: 9","Unnamed: 10","model"],axis=1)
df1
```

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900
1	2.0	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800
2	3.0	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200
3	4.0	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000
4	5.0	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700
...	...	...	...	...	...	...	...	...
1532	1533.0	51.0	1917.0	52008.0	1.0	45.548000	11.54946995	9900
1533	1534.0	51.0	3712.0	115280.0	1.0	45.069679	7.704919815	5200
1534	1535.0	74.0	3835.0	112000.0	1.0	45.845692	8.666870117	4600
1535	1536.0	51.0	2223.0	60457.0	1.0	45.481541	9.413479805	7500
1536	1537.0	51.0	2557.0	80750.0	1.0	45.000702	7.68227005	5990

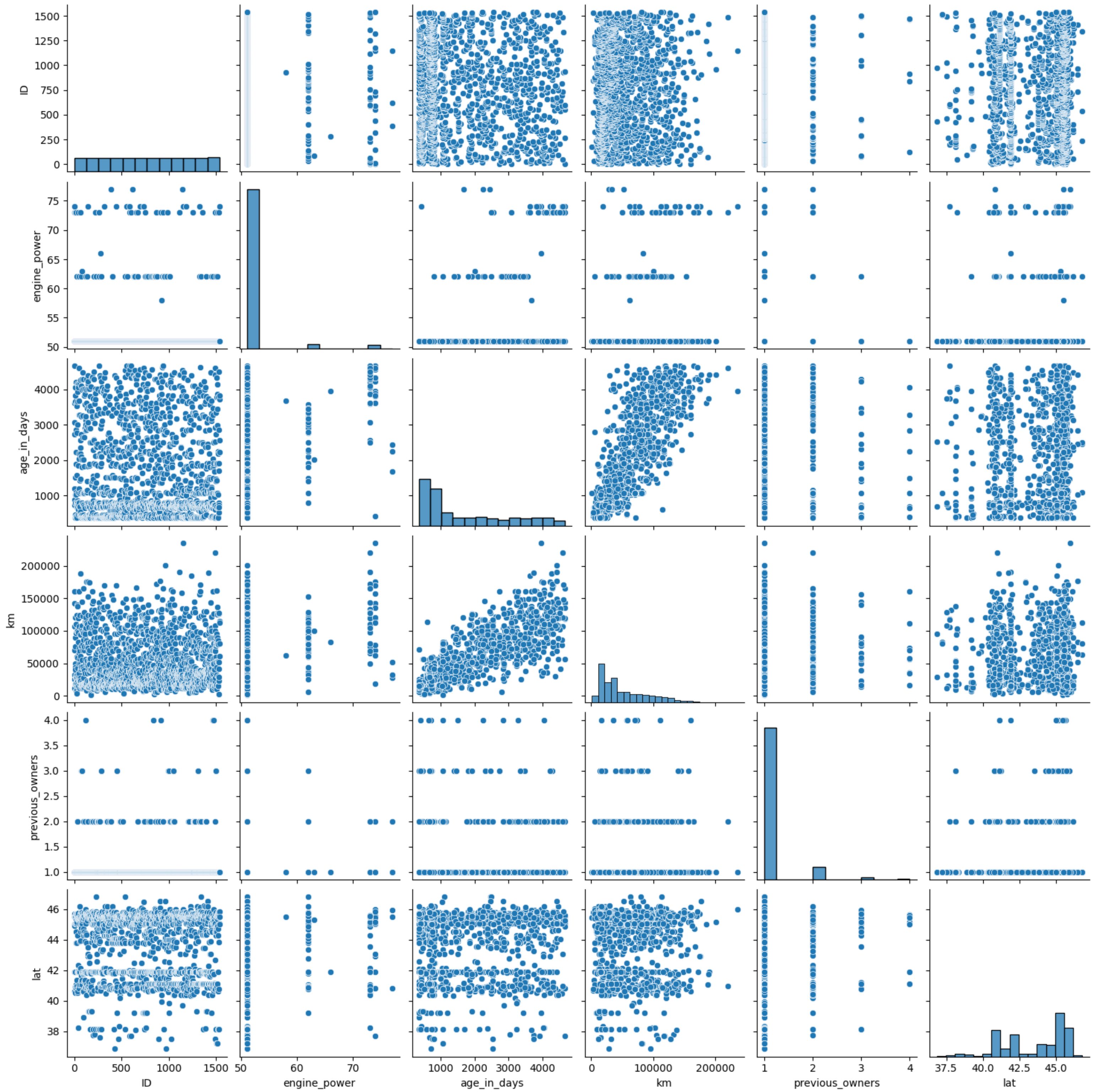
1537 rows × 8 columns

```
df1.describe()
```

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1537.000000	1537.000000	1537.000000	1537.000000	1537.000000	1537.000000
mean	769.000000	51.905010	1650.905660	53395.439167	1.123617	43.543455
std	443.837996	3.989254	1289.938635	40059.858383	0.416546	2.132631
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839
25%	385.000000	51.000000	670.000000	20000.000000	1.000000	41.802990
50%	769.000000	51.000000	1035.000000	39024.000000	1.000000	44.399971
75%	1153.000000	51.000000	2616.000000	79800.000000	1.000000	45.467960
max	1537.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612

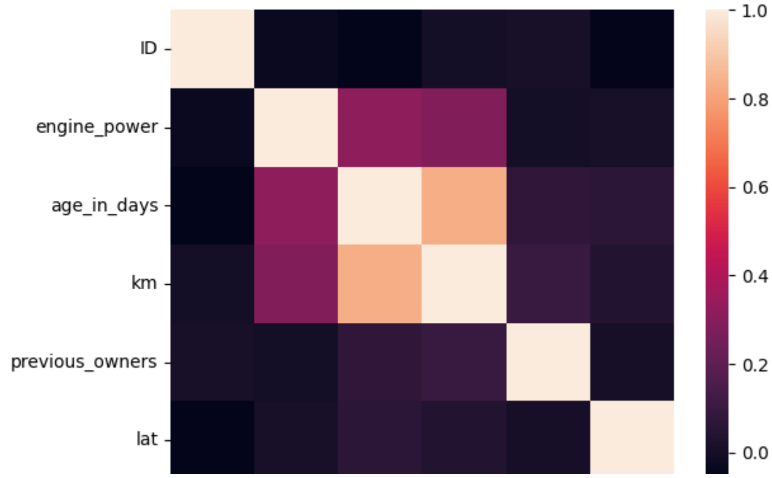
```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x77ff6a8b7760>
```




```
sns.heatmap(df1.corr())
```

```
<ipython-input-169-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to F
  sns.heatmap(df1.corr())
<Axes: >
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge,Lasso
```

```
y=df1['age_in_days']
x=df1.drop(['age_in_days'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

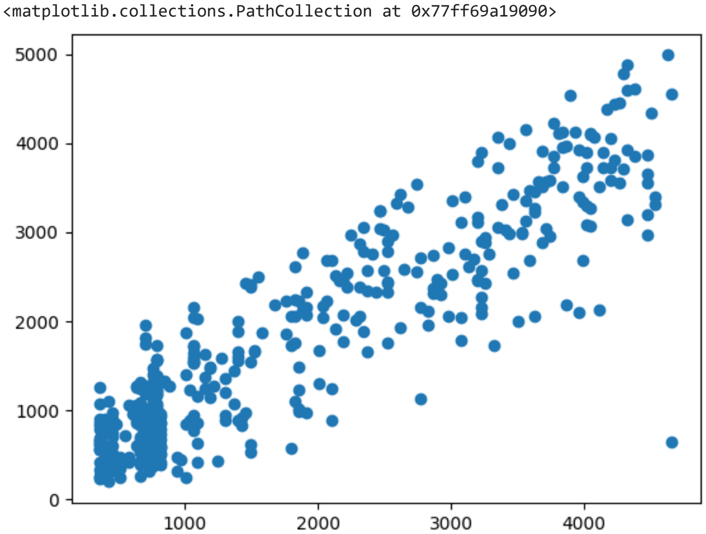
```
3540.607087817586
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
ID	-0.110568
engine_power	22.853431
km	0.007339
previous_owners	15.189556
lat	12.962751
lon	-11.709336
price	-0.447022

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff69a19090>
```



```
model.score(x_test,y_test)
```

```
0.8245985855042313
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.8246038476674138
0.8245854230944103
```

```
daf=pd.read_csv("/content/2_2015.csv")
daf
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...	...	...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0.67042
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858

```
daf1=daf.drop(["Country","Region"],axis=1)
daf1.isna().sum()
```

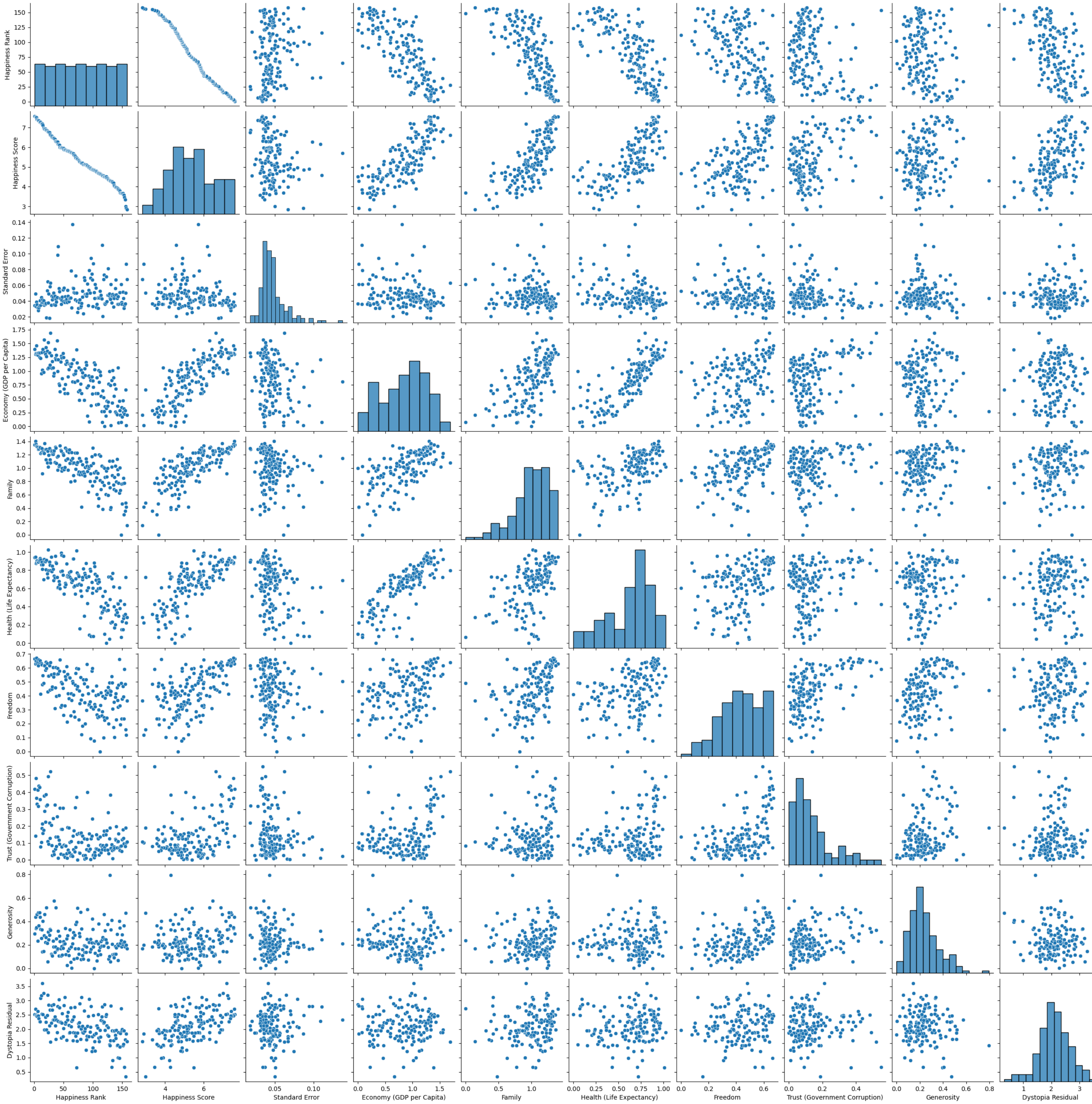
```
Happiness Rank                   0
Happiness Score                  0
Standard Error                   0
Economy (GDP per Capita)         0
Family                           0
Health (Life Expectancy)         0
Freedom                          0
Trust (Government Corruption)    0
Generosity                       0
Dystopia Residual                0
dtype: int64
```

```
daf1.describe()
```

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.098977
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.553550
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.328580
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.759410
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.095415
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.462415
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.602140

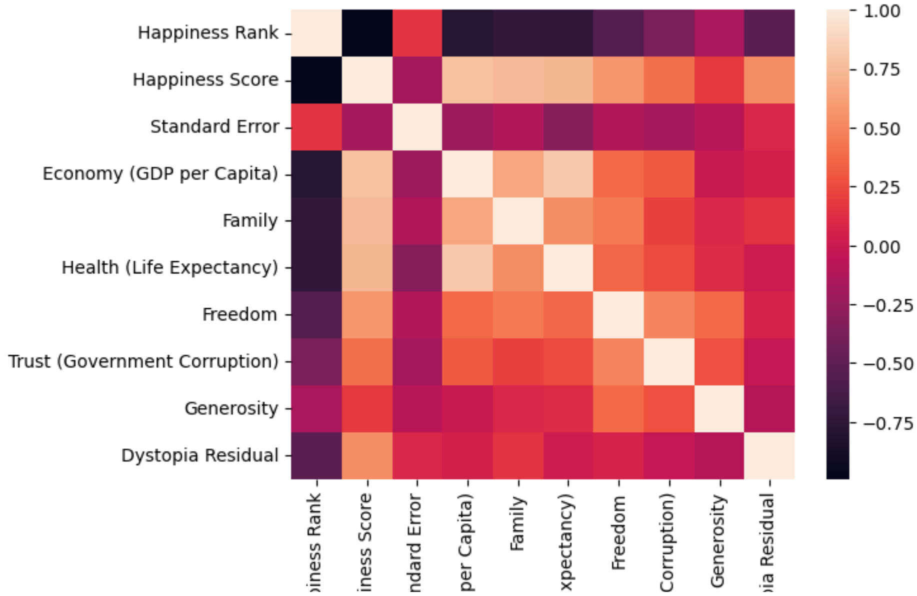
```
sns.pairplot(daf1)
```

```
<seaborn.axisgrid.PairGrid at 0x77ff699b58a0>
```




```
sns.heatmap(daf1.corr())
```

```
<Axes: >
```




```
y=daf1['Standard Error']
x=daf1.drop(['Standard Error'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()

model.fit(x_train,y_train)

model.intercept_
```

```
0.23535376084260795
```

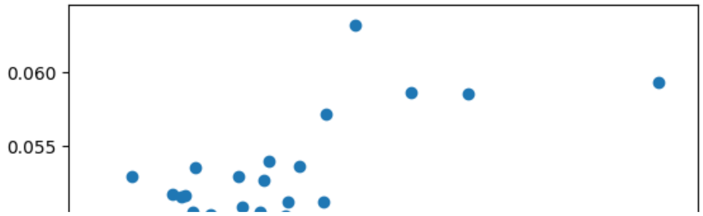
```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
Happiness Rank	-0.000610
Happiness Score	-2.950489
Economy (GDP per Capita)	2.936649
Family	2.922085
Health (Life Expectancy)	2.895794
Freedom	2.934559
Trust (Government Corruption)	2.905042
Generosity	2.929666
Dystopia Residual	2.928397

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

<matplotlib.collections.PathCollection at 0x77ff655ad870>




```
model.score(x_test,y_test)
```

0.08208352621456139

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

0.08191027311878352
-0.0030562607358459726

```
df=pd.read_csv("/content/3_Fitness-1.csv")
df
```

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

```
df=df.drop(['Row Labels'],axis=1)
```

```
df["Sum of Jan"]=df["Sum of Jan"].replace("%","",regex=True).astype(float)
df["Sum of Feb"]=df["Sum of Feb"].replace("%","",regex=True).astype(float)
df["Sum of Mar"]=df["Sum of Mar"].replace("%","",regex=True).astype(float)
df
```

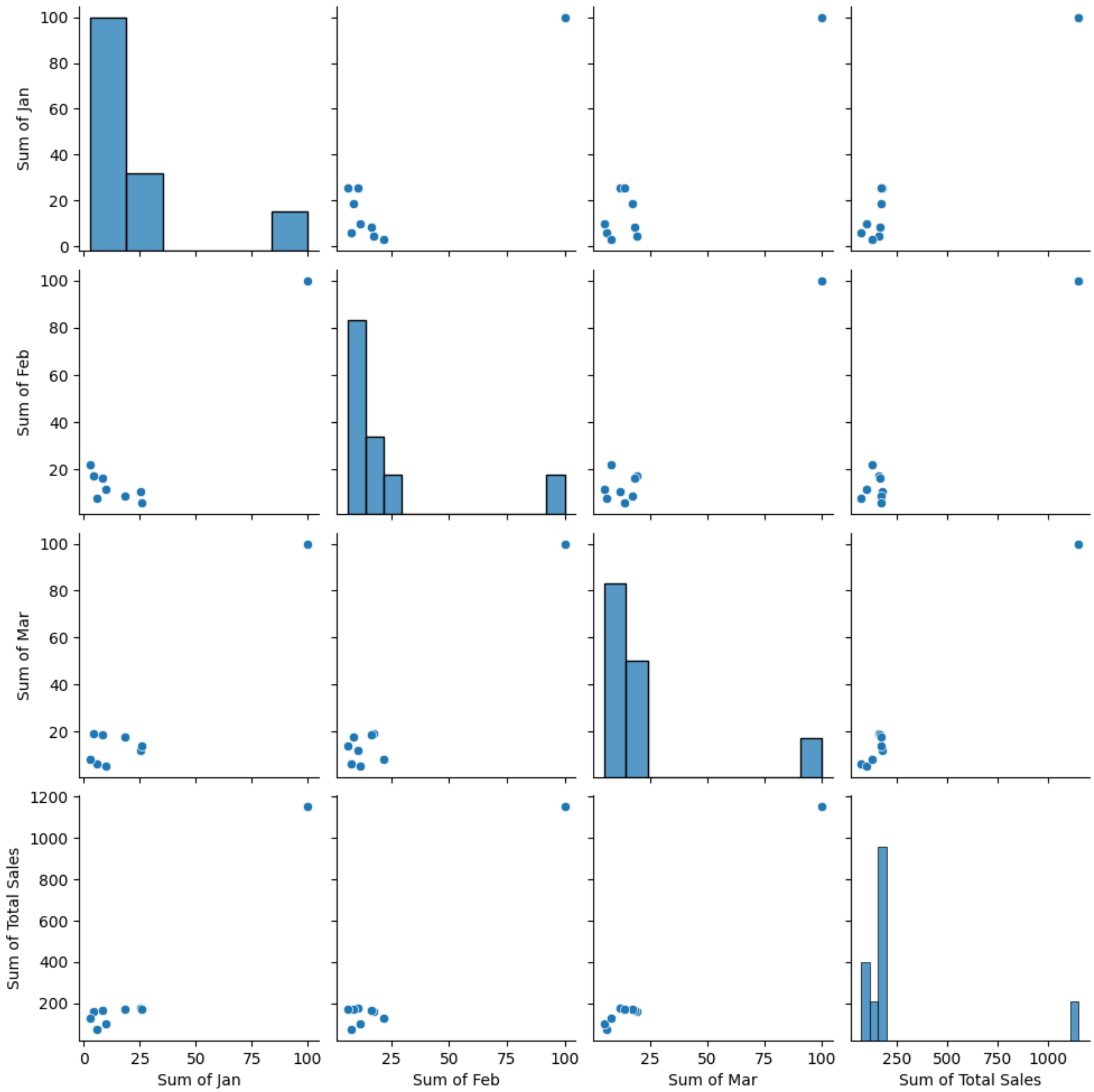
	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	5.62	7.73	6.16	75
1	4.21	17.27	19.21	160
2	9.83	11.60	5.17	101
3	2.81	21.91	7.88	127
4	25.28	10.57	11.82	179
5	8.15	16.24	18.47	167
6	18.54	8.76	17.49	171
7	25.56	5.93	13.79	170
8	100.00	100.00	100.00	1150

```
df.describe()
```

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	9.000000	9.000000	9.000000	9.000000
mean	22.222222	22.223333	22.221111	255.555556
std	30.438329	29.612265	29.640999	337.332963
min	2.810000	5.930000	5.170000	75.000000

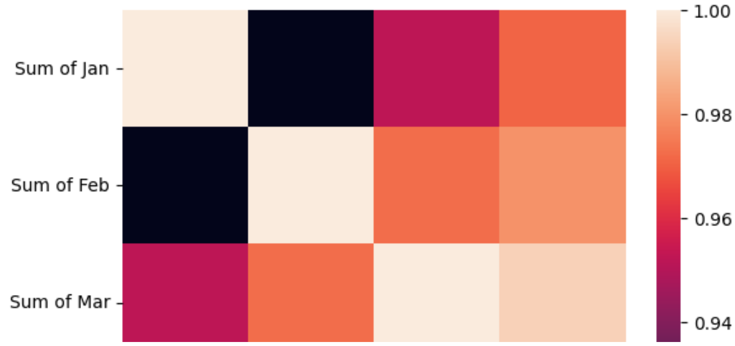
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x77ff6a8d8670>
```




```
sns.heatmap(df.corr())
```

```
<Axes: >
```




```
y=df['Sum of Feb']
x=df.drop(['Sum of Feb'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

```
0.0014324800799876414
```

o

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
```

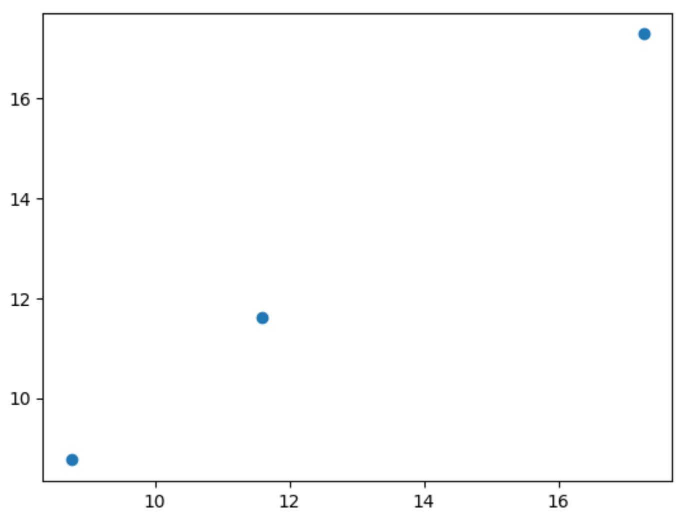
```
coeff
```

	Coefficient
Sum of Jan	-0.917752
Sum of Mar	-1.046221
Sum of Total Sales	0.257736

```
prediction=model.predict(x_test)
```

```
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff633d5780>
```




```
model.score(x_test,y_test)
```

```
0.9999983656425949
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.9173143100864749
0.02592453617404189
```

```
df=pd.read_csv("/content/6_Salesworkload1.csv")
df
```

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units	Turnover	Customer	Area (m2)	Opening hours
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0	1226244.0	NaN	953.04	Type A
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0	387810.0	NaN	720.48	Type A
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0	654657.0	NaN	966.72	Type A
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0	499434.0	NaN	1053.36	Type A
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0	329397.0	NaN	1053.36	Type A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.0	3886530.0	14538825.0	NaN	#NV	Type A
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.0	245.0	0.0	NaN	#NV	Type A
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.0	0.0	0.0	NaN	#NV	Type A
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.0	245.0	0.0	NaN	#NV	Type A

```
df.isna().sum()
```

```
MonthYear           0
Time index          8
Country             8
StoreID             8
City                8
Dept_ID             8
Dept. Name          8
HoursOwn            8
HoursLease          8
Sales units         8
Turnover            8
Customer         7658
Area (m2)           8
Opening hours       8
dtype: int64
```

```
df1=df.drop(["Customer","Country","Dept. Name","Opening hours","City"],axis=1)
df1=df1.dropna()
```

```
val=df["HoursOwn"]=="?"
print(df.index[val])
```

```
Int64Index([2966, 5889], dtype='int64')
```

```
val=["#NV"]
df1["Area (m2)"].isin(val).sum()
df1=df1.drop([2966,5889],axis=0)
```

```
df1=df1.drop(["Area (m2)"],axis=1)
df1
```

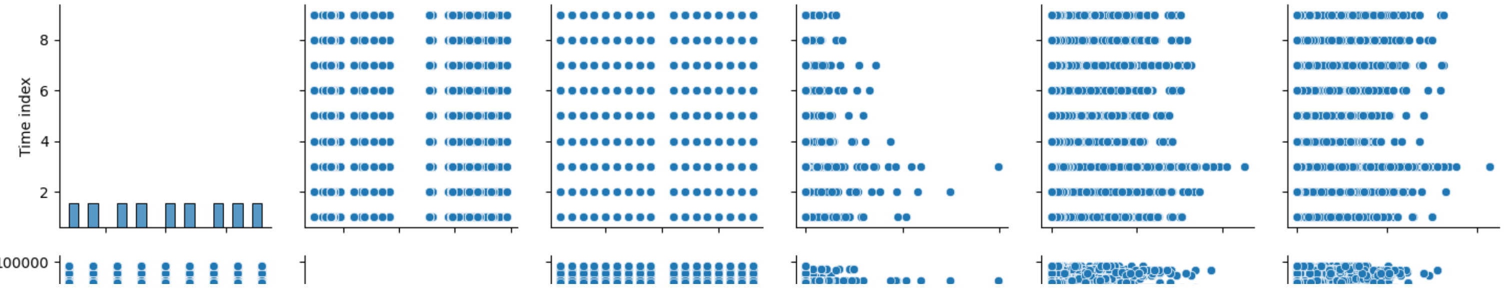
	MonthYear	Time index	StoreID	Dept_ID	HoursOwn	HoursLease	Sales units	Turnover

```
df1.describe()
```

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover
count	7648.000000	7648.000000	7648.000000	7648.000000	7.648000e+03	7.648000e+03
mean	4.999869	61999.574268	9.472019	22.041841	1.076492e+06	3.721465e+06
std	2.582369	29923.753974	5.337296	133.316467	1.728290e+06	6.004067e+06
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00
25%	3.000000	29650.000000	5.000000	0.000000	5.455375e+04	2.724480e+05
50%	5.000000	76852.000000	9.000000	0.000000	2.932300e+05	9.315390e+05
75%	7.000000	87703.000000	14.000000	0.000000	9.164325e+05	3.259014e+06
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07

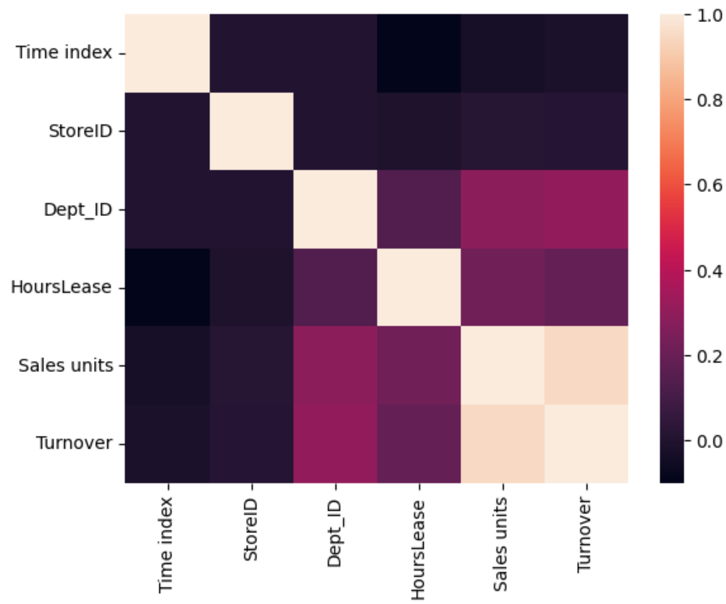
```
sns.pairplot(df1)
```

<seaborn.axisgrid.PairGrid at 0x77ff63402dd0>




```
sns.heatmap(df1.corr())
```

```
<ipython-input-208-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to F
  sns.heatmap(df1.corr())
<Axes: >
```




```
y=df1['Sales units']
x=df1.drop(['Sales units'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
```

```
model.fit(x_train,y_train)
```

```
model.intercept_
```

```
82599.27591178799
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
```

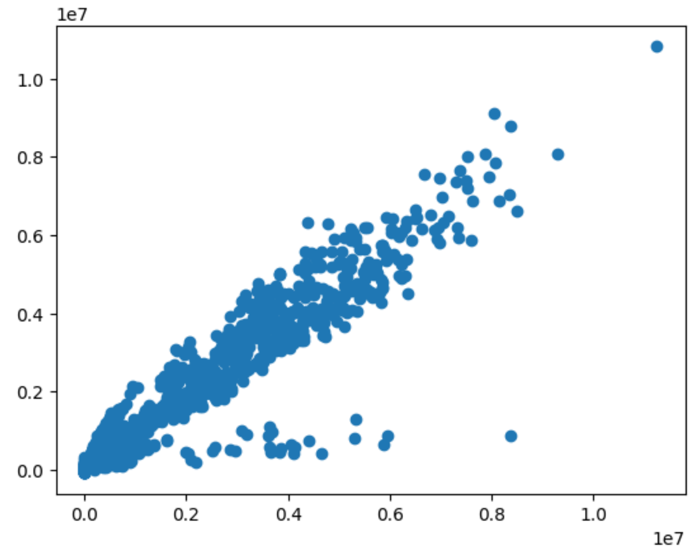
```
coeff
```

	Coefficient
MonthYear	5317.605518
Time index	-3298.987826
StoreID	0.128203
Dept_ID	-9522.316339
HoursOwn	17.086125
HoursLease	472.492932
Turnover	0.246672

```
prediction=model.predict(x_test)
```

```
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff62171bd0>
```




```
model.score(x_test,y_test)
```

```
0.9052719859905412
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.9052719888316975
0.9052719872820221
```

```
df=pd.read_csv("/content/7_uber.csv")
df
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986525	40.740297	1
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006672	40.739620	1
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858957	40.692588	2

```
df1=df.drop(["Unnamed: 0","key","pickup_datetime"],axis=1)
df1
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...
199995	3.0	-73.987042	40.739367	-73.986525	40.740297	1
199996	7.5	-73.984722	40.736837	-74.006672	40.739620	1

```
df1=df1.dropna()
df1.isna().sum()
```

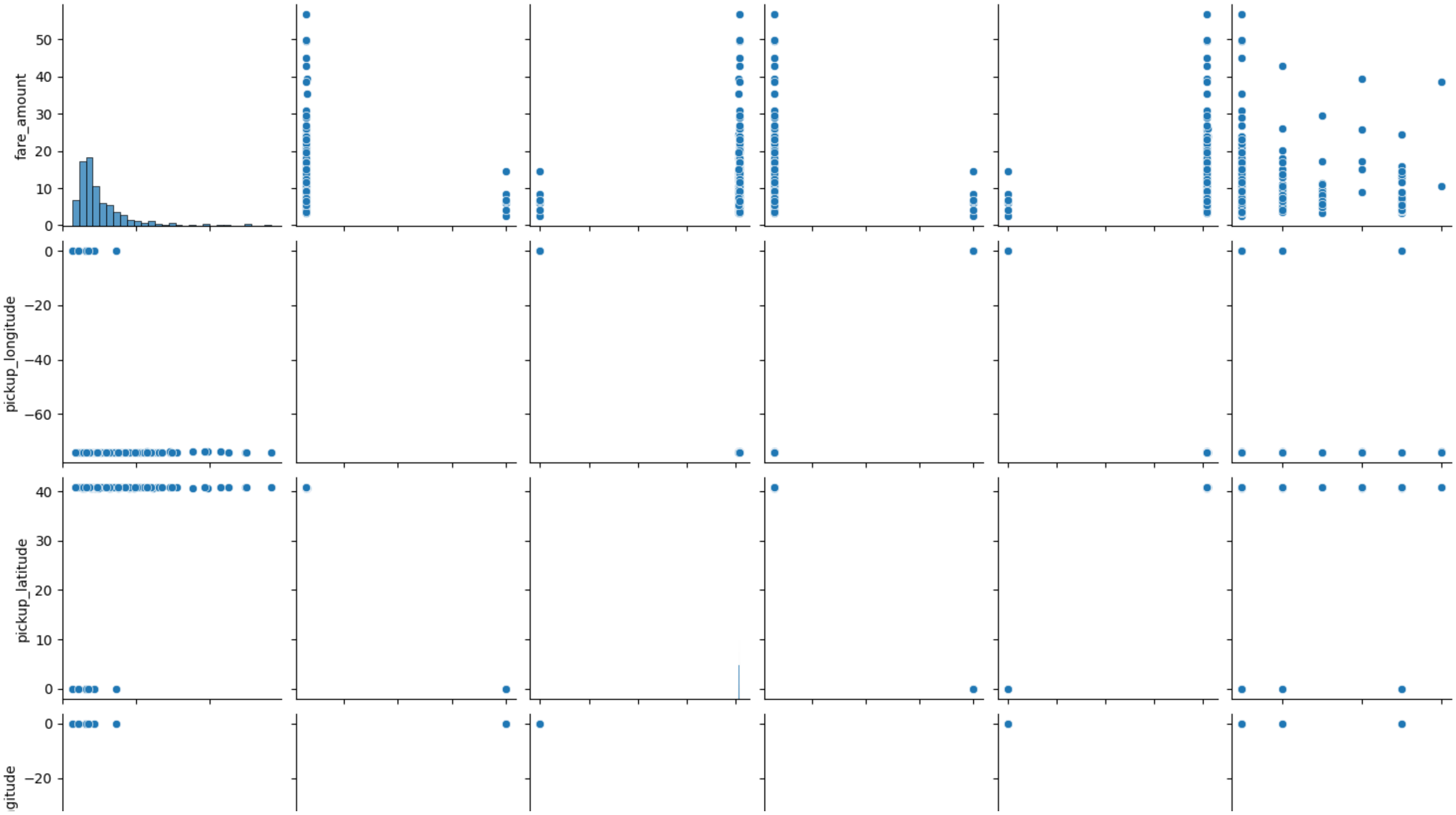
```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

```
df1.describe()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	199999.000000	199999.000000	199999.000000	199999.000000	199999.000000	199999.000000
mean	11.359892	-72.527631	39.935881	-72.525292	39.923890	1.684543
std	9.901760	11.437815	7.720558	13.117408	6.794829	1.385995
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992065	40.734796	-73.991407	40.733823	1.000000
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	12.500000	-73.967154	40.767158	-73.963658	40.768001	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

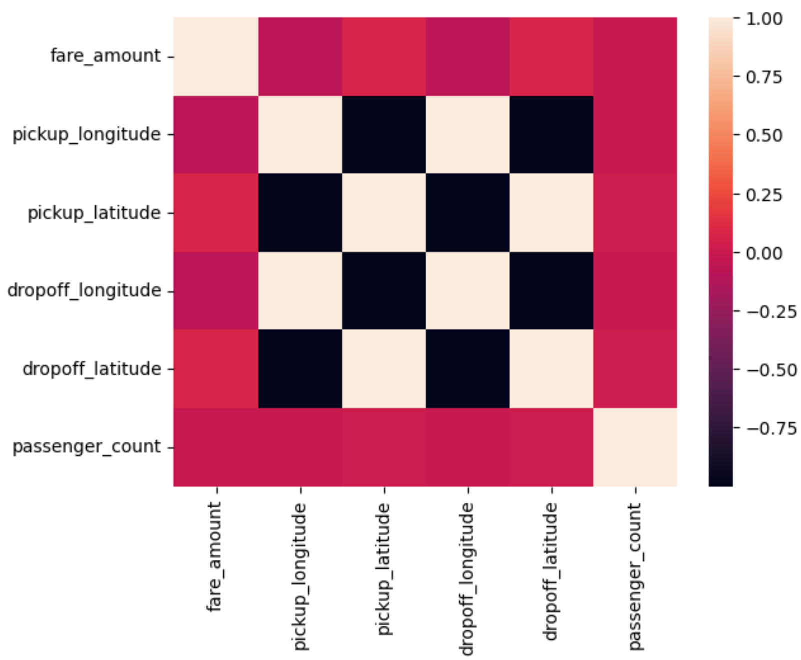
```
sns.pairplot(df1.iloc[0:300,:])
```

<seaborn.axisgrid.PairGrid at 0x77ff6201d5a0>




```
sns.heatmap(df1.iloc[0:300,:].corr())
```

<Axes: >




```
y=df1['passenger_count']
x=df1.drop(['passenger_count'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

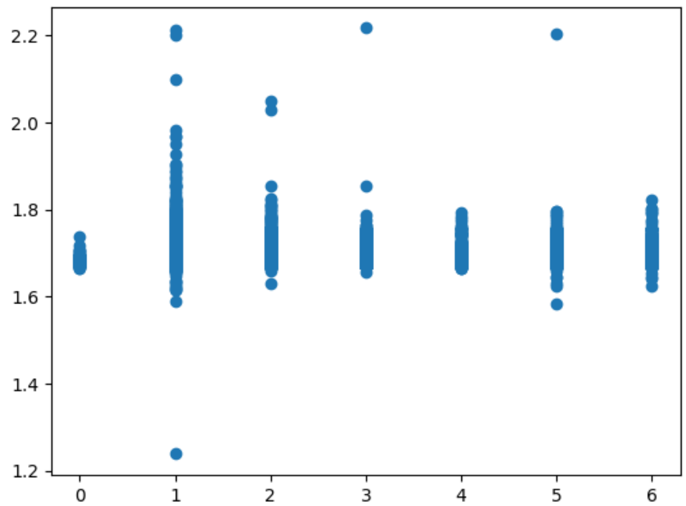
1.6616864300378735

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
fare_amount	0.001586
pickup_longitude	-0.000802
pickup_latitude	-0.001208
dropoff_longitude	-0.000543
dropoff_latitude	-0.001153

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff506cbe20>
```




```
print(model.score(x_test,y_test))
print(model.score(x_train,y_train))
```

```
6.113243523797607e-05
0.0001366165686994547
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
6.113223065817852e-05
-1.613536212707878e-05
```

```
df=pd.read_csv("/content/8_BreastCancerPrediction (1).csv")
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_wor
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	17
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	23
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	25
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	26
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	16
...	...	...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	26
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	38
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	34
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	39
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	30

569 rows × 33 columns

```
df.isna().sum()
```

```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

```
df1=df.drop(["Unnamed: 32"],axis=1)
df1["diagnosis"]= df1["diagnosis"].replace("M",1,regex=True)
df1["diagnosis"]= df1["diagnosis"].replace("B",0,regex=True)
df1
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_wors
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.38
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.9
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.57
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.91
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.5
...	...	...	...	...	...	...	...	...	...	...	...	
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.4
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.6
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.98
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.7
568	92751	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.45

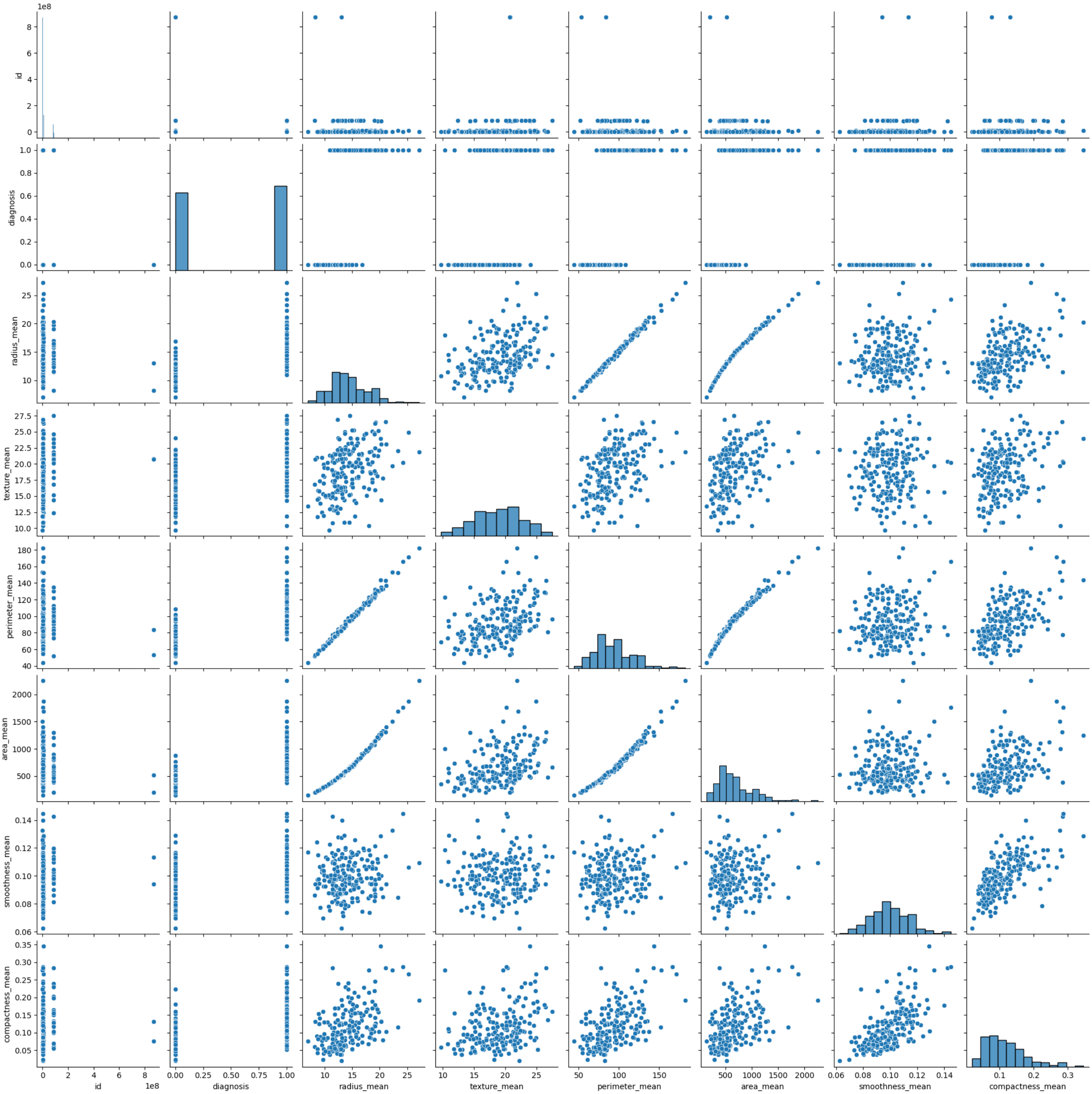
569 rows × 32 columns

```
df1.describe()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	rad
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	5
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	...	
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	...	

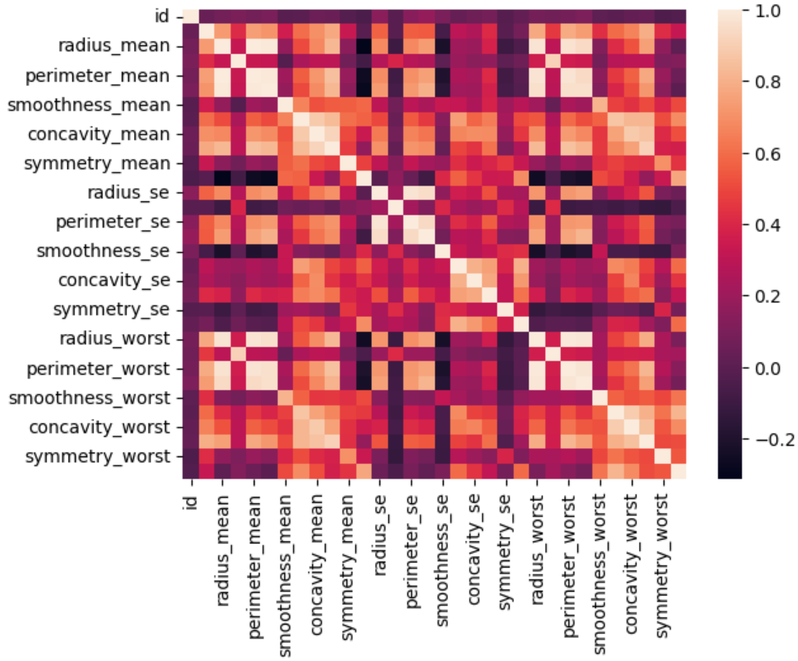
```
sns.pairplot(df1.iloc[:20,:6])
```

```
<seaborn.axisgrid.PairGrid at 0x77ff5073d030>
```



```
sns.heatmap(df1.corr())
```

```
<Axes: >
```




```
y=df1['area_mean']
x=df1.drop(['area_mean'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

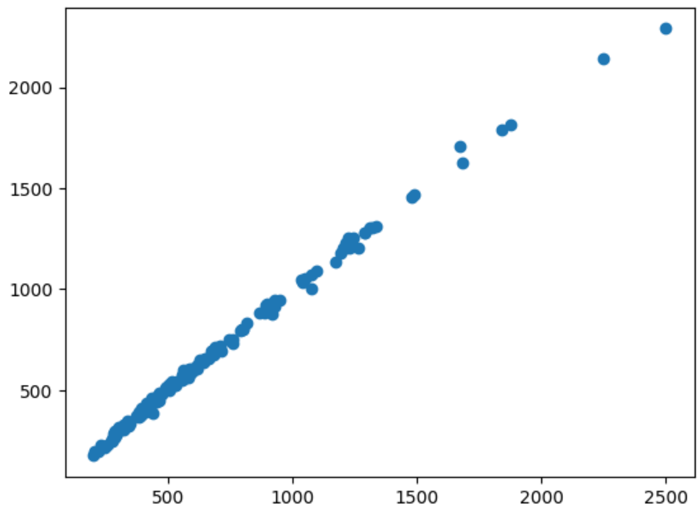
```
-303.20237369576057
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
id	1.534436e-08
diagnosis	5.474975e+00
radius_mean	7.119492e+01
texture_mean	-2.149570e-01
perimeter_mean	6.664949e+00
smoothness_mean	-4.479525e+02
compactness_mean	-4.893977e+02
concavity_mean	2.458095e+01
concave points_mean	1.864207e+02

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff4e608eb0>
```



smoothness_worst 3.552571e+02

```
print(model.score(x_train,y_train))
```

```
0.9973315510664574
```

```
model.score(x_test,y_test)
```

```
0.9955370544009499
```

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
la=Lasso(alpha=10)
la.fit(x_train,y_train)
print(rr.score(x_test,y_test))
la.score(x_test,y_test)
```

```
0.9927343626688578
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: Ill-conditioned matrix (rcond=1.65851e-18): result may not be accura
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
0.9899364925652687
```

```
df=pd.read_csv("/content/11_winequality-red.csv")
df
```

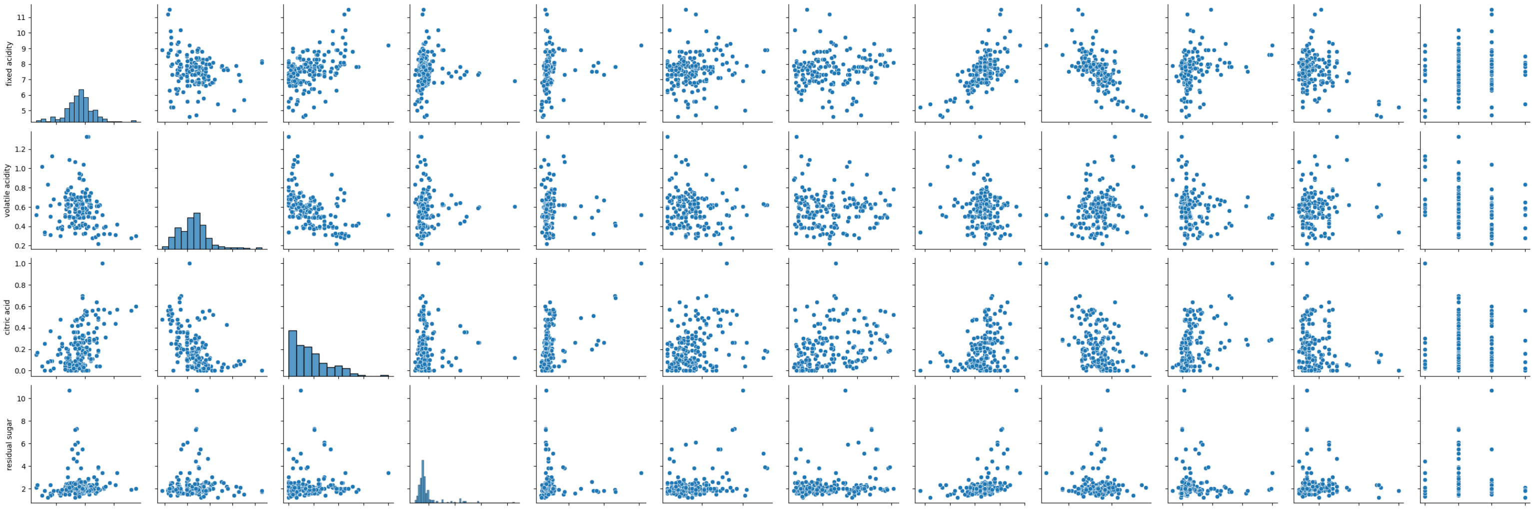
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

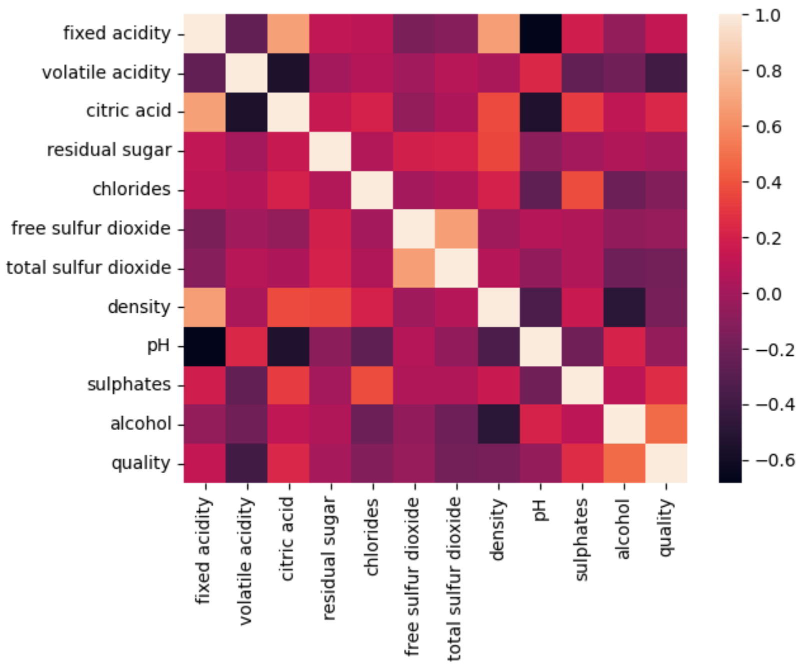
```
sns.pairplot(df.iloc[:200,:])
```

<seaborn.axisgrid.PairGrid at 0x77ff4e64b460>




```
sns.heatmap(df.corr())
```

<Axes: >




```
y=df['density']
x=df.drop(['density'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

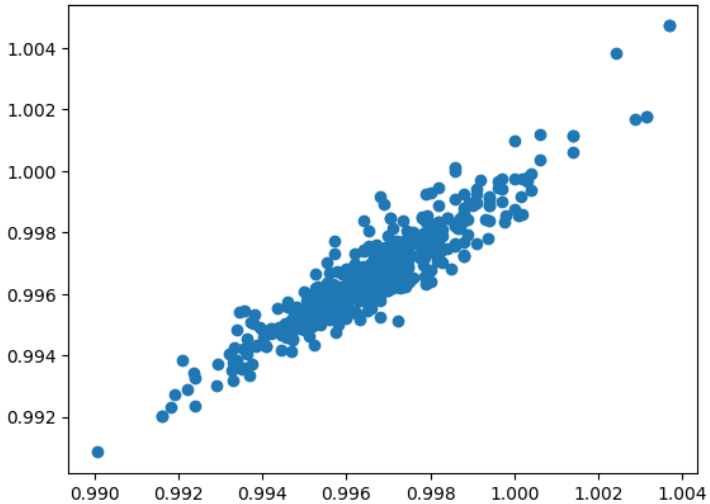
0.9787608689936755

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
fixed acidity	0.000929
volatile acidity	0.000881
citric acid	0.000249

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x77ff46275540>
```




```
model.score(x_test,y_test)
```

```
0.8669526751569081
```

```
df=pd.read_csv("/content/13_placement.csv")
df
```

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

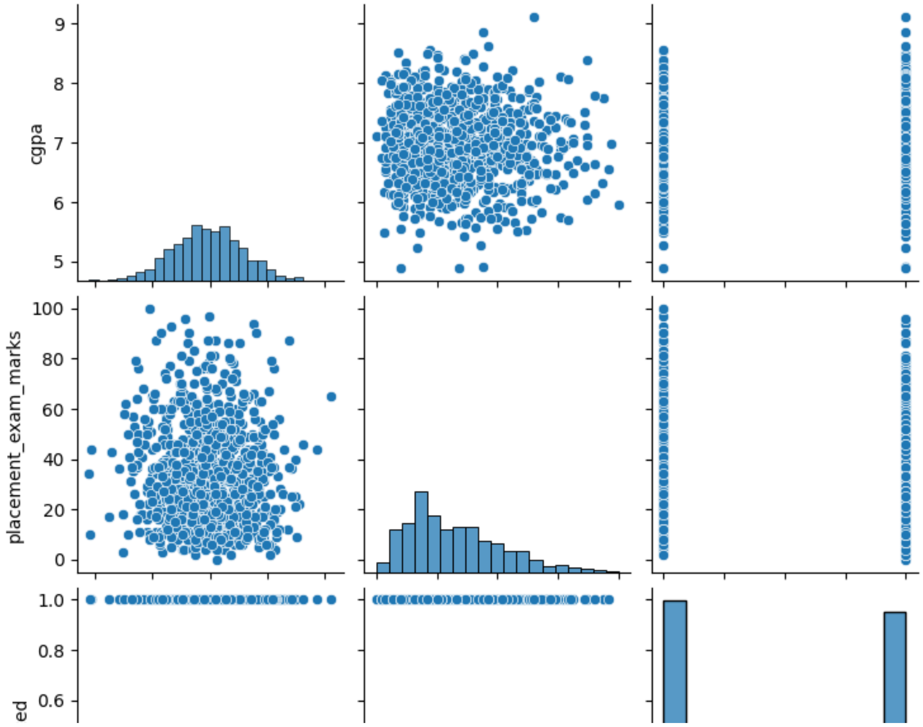
1000 rows × 3 columns

```
df.isna().sum()
```

```
cgpa                    0
placement_exam_marks    0
placed                  0
dtype: int64
```

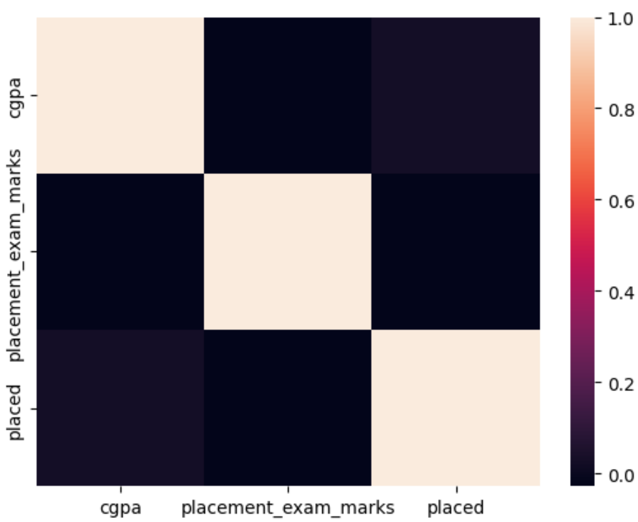
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x77ff462bc250>
```




```
sns.heatmap(df.corr())
```

```
<Axes: >
```




```
y=df['cgpa']
x=df.drop(['cgpa'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

```
6.992731140418286
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
placement_exam_marks	-0.001376
placed	0.043491

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```