```
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	
0	1.0	lounge	51.0	882.0	25000.0	1.0	4
1	2.0	pop	51.0	1186.0	32500.0	1.0	4
2	3.0	sport	74.0	4658.0	142228.0	1.0	4
3	4.0	lounge	51.0	2739.0	160000.0	1.0	4
4	5.0	pop	73.0	3074.0	106880.0	1.0	4
...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	
1545	NaN	NaN	NaN	NaN	NaN	NaN	
1546	NaN	NaN	NaN	NaN	NaN	NaN	
1547	NaN	NaN	NaN	NaN	NaN	NaN	

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

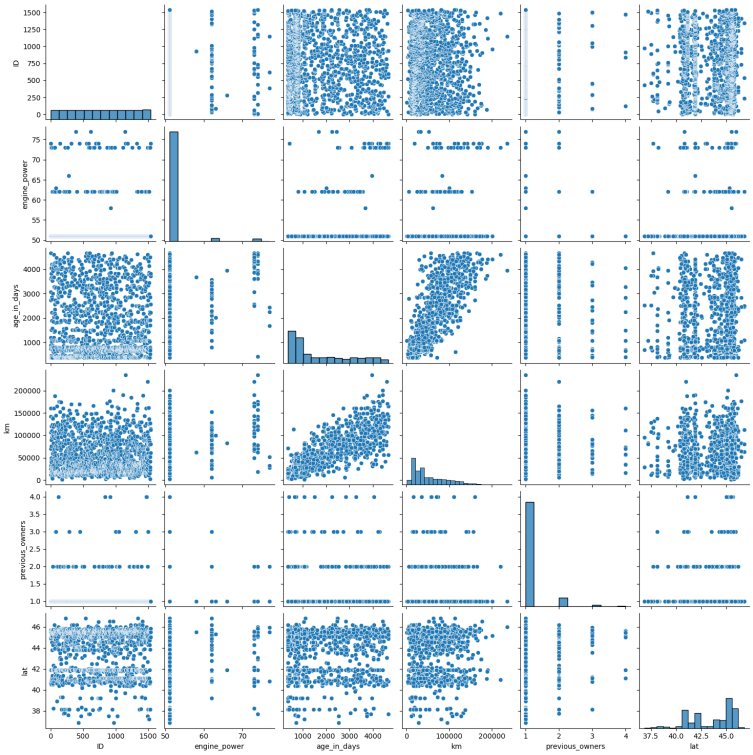
	ID	engine_power	age_in_days	km	previous_owners	l
0	1.0	51.0	882.0	25000.0	1.0	44.9072
1	2.0	51.0	1186.0	32500.0	1.0	45.6663
2	3.0	74.0	4658.0	142228.0	1.0	45.5033
3	4.0	51.0	2739.0	160000.0	1.0	40.6331
4	5.0	73.0	3074.0	106880.0	1.0	41.9032
...	...	...	...	...	...	
1532	1533.0	51.0	1917.0	52008.0	1.0	45.5480
1533	1534.0	51.0	3712.0	115280.0	1.0	45.0696
1534	1535.0	74.0	3835.0	112000.0	1.0	45.8456
1535	1536.0	51.0	2223.0	60457.0	1.0	45.4815

```
df1.describe()
```

	ID	engine_power	age_in_days	km	previous_own
count	1537.000000	1537.000000	1537.000000	1537.000000	1537.000
mean	769.000000	51.905010	1650.905660	53395.439167	1.123
std	443.837996	3.989254	1289.938635	40059.858383	0.416
min	1.000000	51.000000	366.000000	1232.000000	1.000
25%	385.000000	51.000000	670.000000	20000.000000	1.000
50%	769.000000	51.000000	1035.000000	39024.000000	1.000
75%	1153.000000	51.000000	2616.000000	79800.000000	1.000

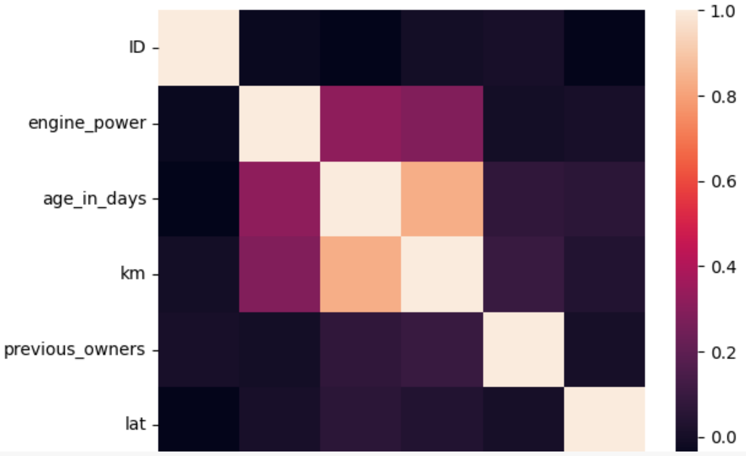
```
sns.pairplot(df1)
```

<seaborn.axisgrid.PairGrid at 0x77ff6a8b7760>




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

```
<ipython-input-169-3ed1a1a51dc0>:1: FutureWarning: The default value of nu
  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_
```

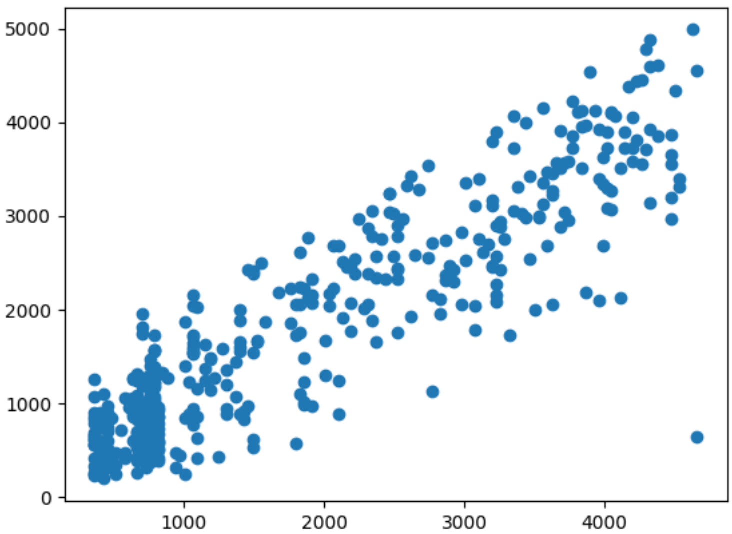
```
3540.607087817586
```

```
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
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261
...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan	154	3.465	0.03464	0.22208	0.77370

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

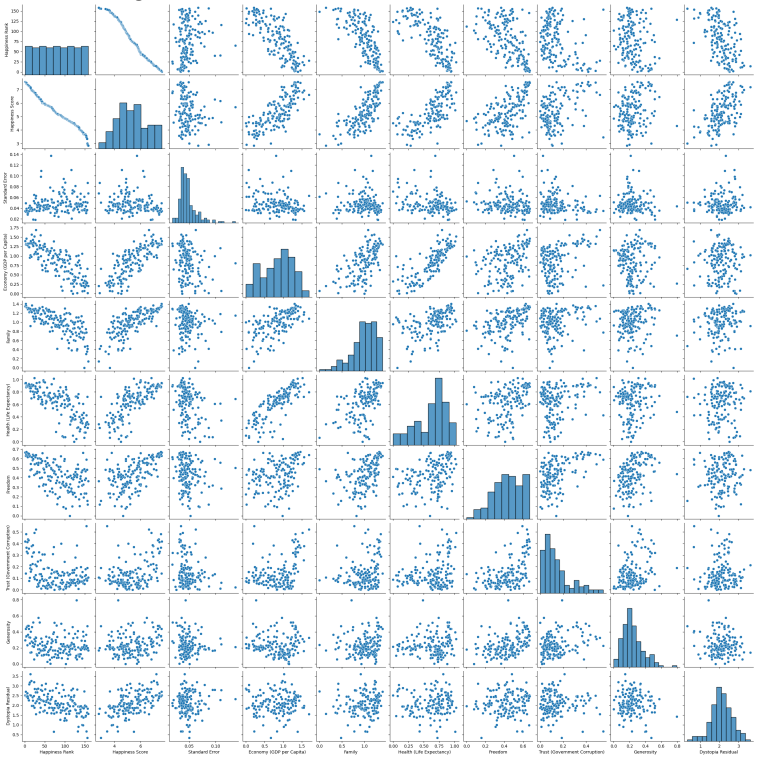
```
Happiness Rank                   0
Happiness Score                  0
Standard Error                   0
Economy (GDP per Capita)         0
Family                           0
Health (Life Expectancy)         0
Freedom                          0
Trust (Government Corruption)    0
Generosity      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

```
daf1.describe()
```

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Heal (Li Expectanc
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.0000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.6302
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.2470
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.0000
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.4391
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.6967
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.8110

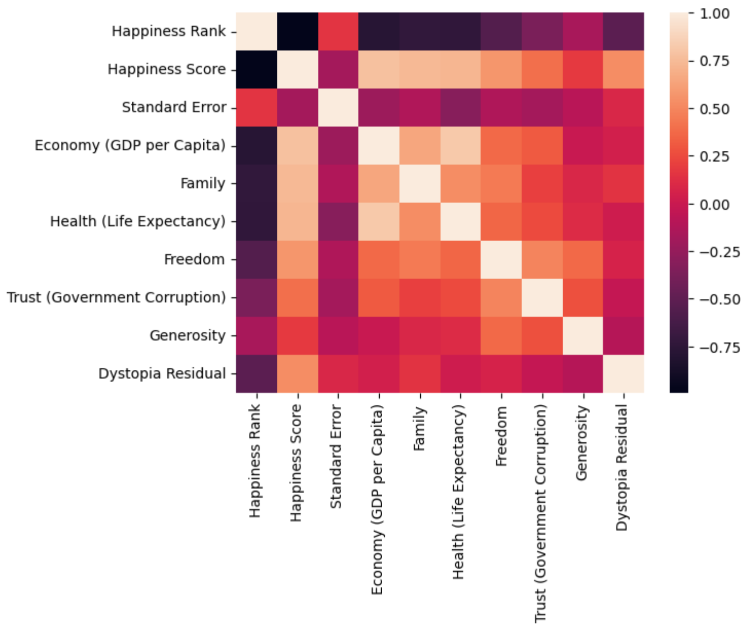
```
sns.pairplot(daf1)
```

<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_
```

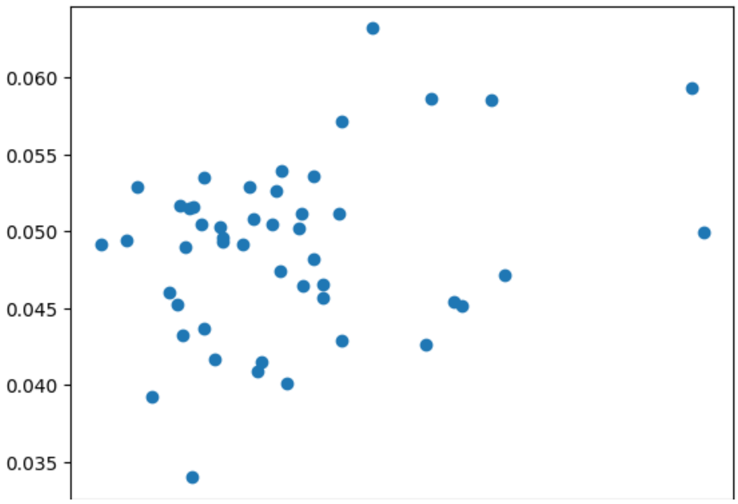
```
0.23535376084260795
```

```
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

```
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

```
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
```

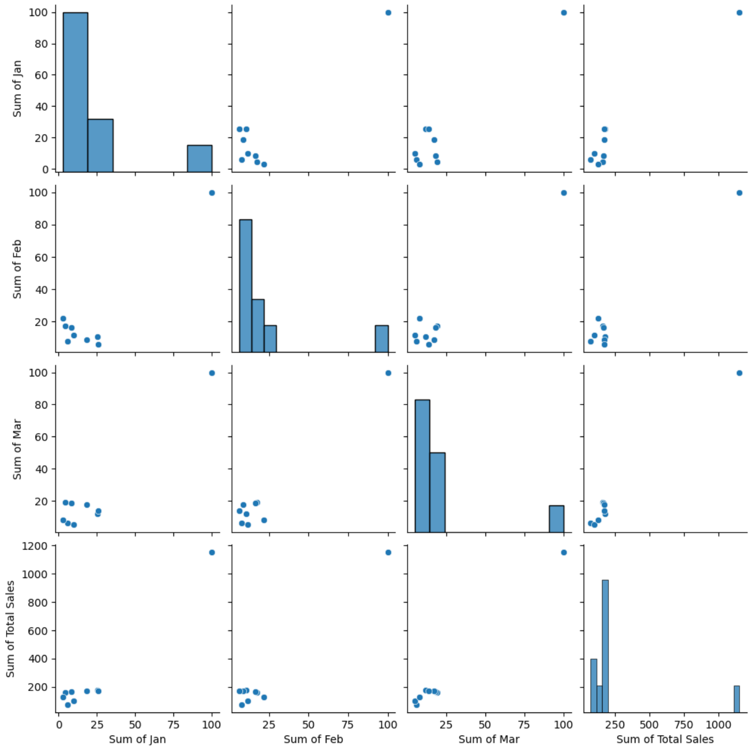
	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	5.62	7.73	6.16	75
1	4.21	17.27	19.21	160
2	9.83	11.60	5.17	101
3	2.81	21.91	7.88	127
4	25.28	10.57	11.82	179
5	8.15	16.24	18.47	167
6	18.54	8.76	17.49	171
7	25.56	5.93	13.79	170

```
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
25%	5.620000	8.760000	7.880000	127.000000
50%	9.830000	11.600000	13.790000	167.000000
75%	25.280000	17.270000	18.470000	171.000000

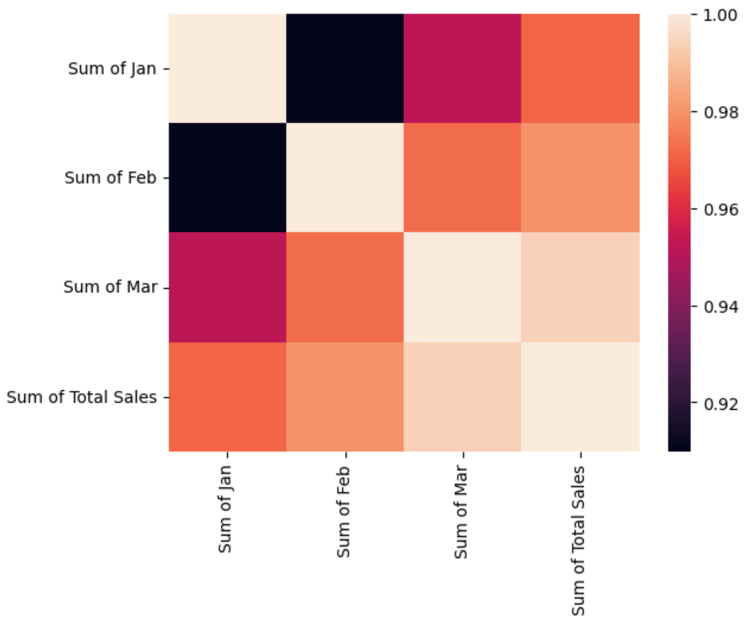
```
sns.pairplot(df)
```

<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_
```

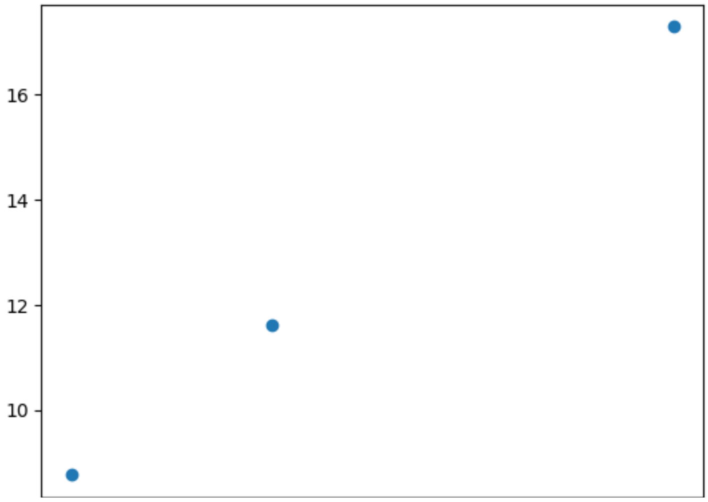
0.0014324800799876414

```
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
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables
...	...	...	...	...	...	...	...
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout

```
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
```

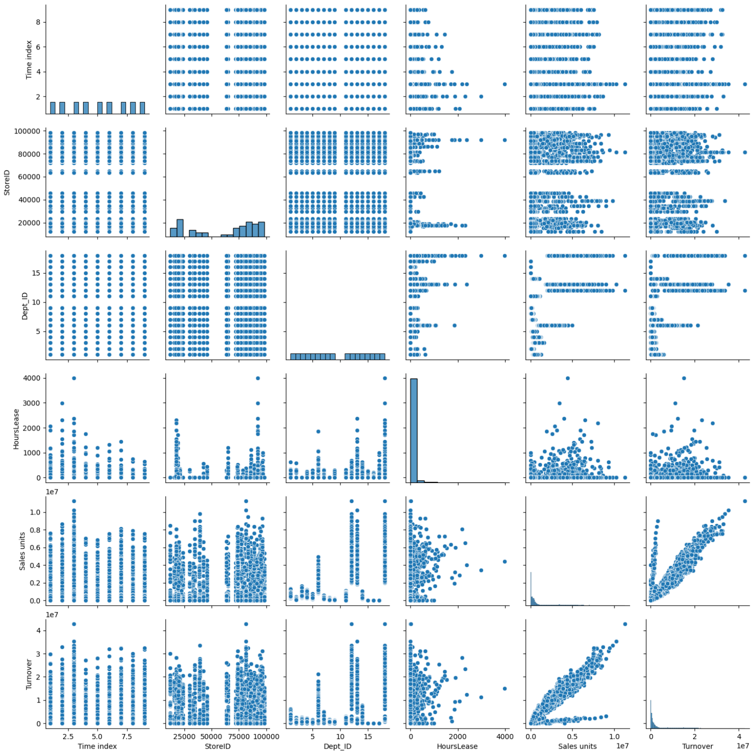
	MonthYear	Time index	StoreID	Dept_ID	HoursOwn	HoursLease	Sales units
0	10.2016	1.0	88253.0	1.0	3184.764	0.0	398560.0
1	10.2016	1.0	88253.0	2.0	1582.941	0.0	82725.0
2	10.2016	1.0	88253.0	3.0	47.205	0.0	438400.0
3	10.2016	1.0	88253.0	4.0	1623.852	0.0	309425.0
4	10.2016	1.0	88253.0	5.0	1759.173	0.0	165515.0
...	...	...	...	...	...	...	...
7653	06.2017	9.0	29650.0	12.0	6322.323	0.0	3886530.0
7654	06.2017	9.0	29650.0	16.0	4270.479	0.0	245.0
7655	06.2017	9.0	29650.0	11.0	0	0.0	0.0
7656	06.2017	9.0	29650.0	17.0	2224.929	0.0	245.0

```
df1.describe()
```

	Time index	StoreID	Dept_ID	HoursLease	Sales units	
count	7648.000000	7648.000000	7648.000000	7648.000000	7.648000e+03	7.
mean	4.999869	61999.574268	9.472019	22.041841	1.076492e+06	3.
std	2.582369	29923.753974	5.337296	133.316467	1.728290e+06	6.
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.
25%	3.000000	29650.000000	5.000000	0.000000	5.455375e+04	2.
50%	5.000000	76852.000000	9.000000	0.000000	2.932300e+05	9.
75%	7.000000	87703.000000	14.000000	0.000000	9.164325e+05	3.

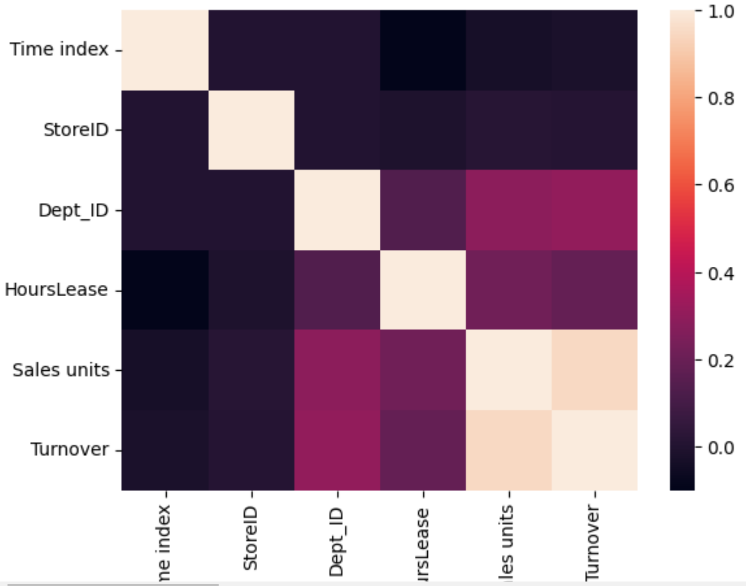
```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x77ff63402dd0>
```



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

```
<ipython-input-208-3ed1a1a51dc0>:1: FutureWarning: The default value of nu
  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_
```

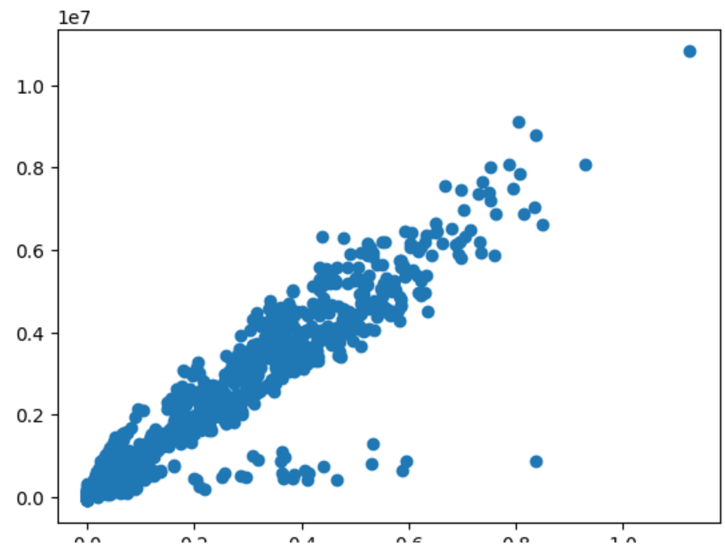
```
82599.27591178799
```

```
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_
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	
...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	

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

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude
0	7.5	-73.999817	40.738354	-73.999512
1	7.7	-73.994355	40.728225	-73.994710
2	12.9	-74.005043	40.740770	-73.962565
3	5.3	-73.976124	40.790844	-73.965316
4	16.0	-73.925023	40.744085	-73.973082
...	...	...	...	...
199995	3.0	-73.987042	40.739367	-73.986525
199996	7.5	-73.984722	40.736837	-74.006672
199997	30.9	-73.986017	40.756487	-73.858957
199998	14.5	-73.997124	40.725452	-73.983215

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

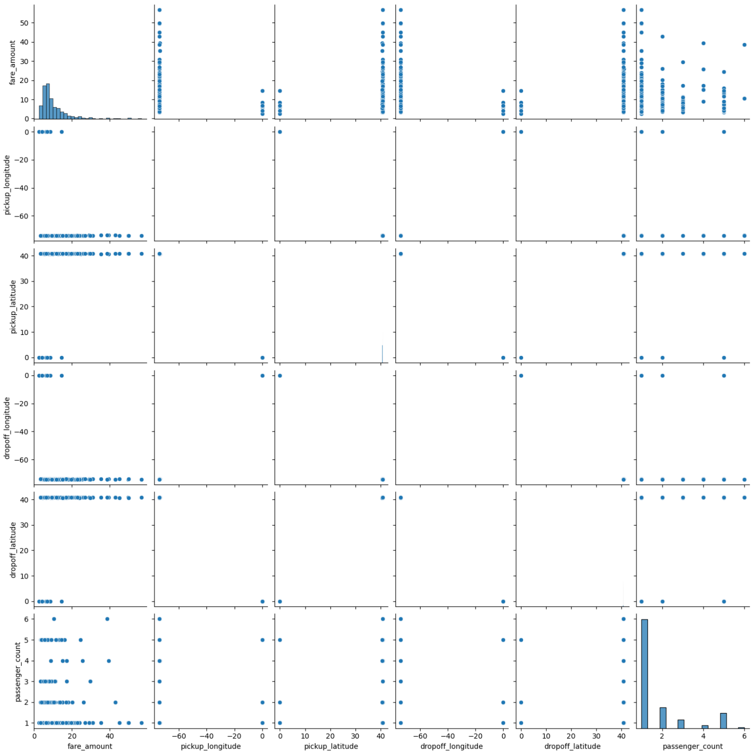
```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

```
df1.describe()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude
count	199999.000000	199999.000000	199999.000000	199999.000000
mean	11.359892	-72.527631	39.935881	-72.525292
std	9.901760	11.437815	7.720558	13.117408
min	-52.000000	-1340.648410	-74.015515	-3356.666300
25%	6.000000	-73.992065	40.734796	-73.991407
50%	8.500000	-73.981823	40.752592	-73.980093
75%	12.500000	-73.967154	40.767158	-73.963658

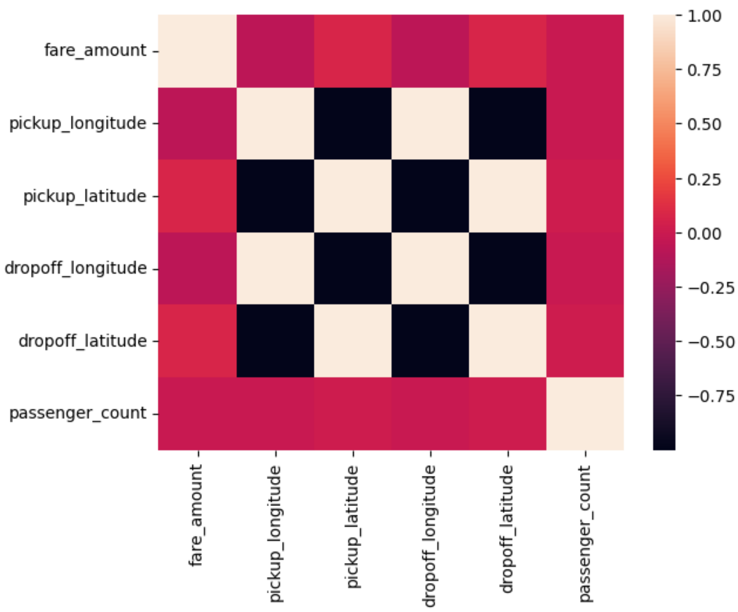
```
sns.pairplot(df1.iloc[0:300,:])
```

<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_
```

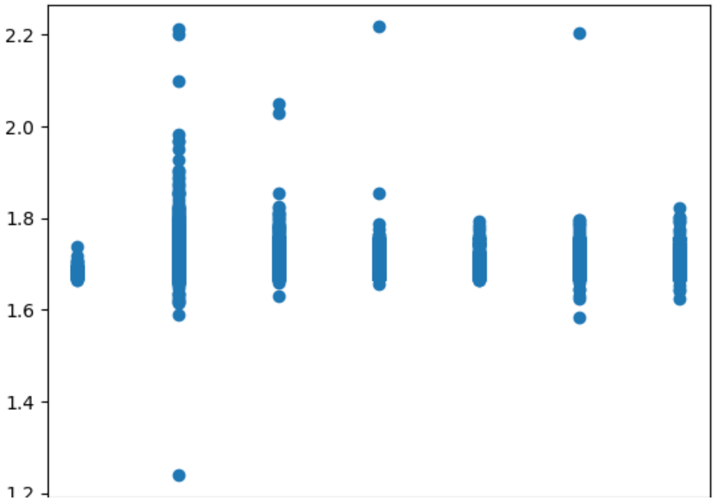
```
1.6616864300378735
```

```
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_
0	842302	M	17.99	10.38	122.80	1(
1	842517	M	20.57	17.77	132.90	1:
2	84300903	M	19.69	21.25	130.00	1:
3	84348301	M	11.42	20.38	77.58	:
4	84358402	M	20.29	14.34	135.10	1:
...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1.

```
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_
0	842302	1	17.99	10.38	122.80	1(
1	842517	1	20.57	17.77	132.90	1:
2	84300903	1	19.69	21.25	130.00	1:
3	84348301	1	11.42	20.38	77.58	:
4	84358402	1	20.29	14.34	135.10	1:
...	...	...	...	...	...	
564	926424	1	21.56	22.39	142.00	1.
565	926682	1	20.13	28.25	131.20	1:
566	926954	1	16.60	28.08	108.30	;
567	927241	1	20.60	29.33	140.10	1:
568	92751	0	7.76	24.54	47.92	

```
df1.describe()
```

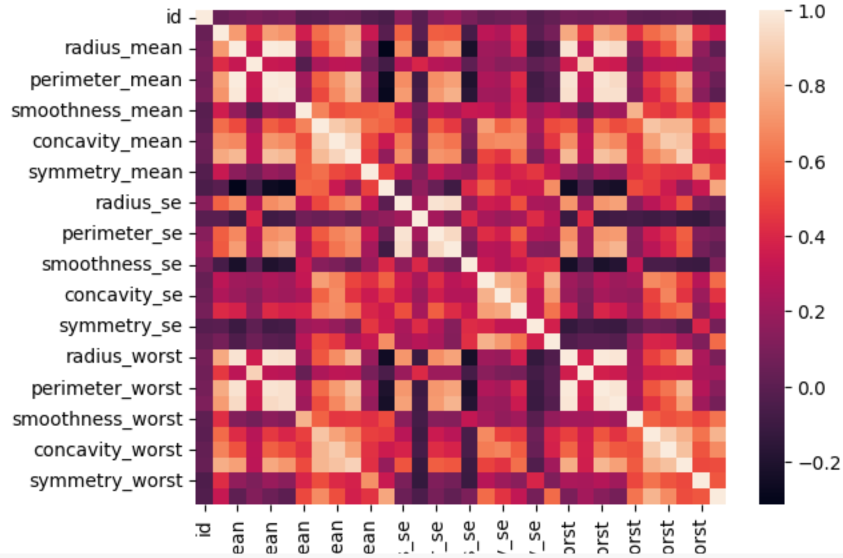
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038
min	8.670000e+03	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000
25%	8.692180e+05	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020
50%	9.060240e+05	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033
75%	8.813129e+06	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074
max	9.113205e+08	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201

8 rows × 32 columns

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

```
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_
```

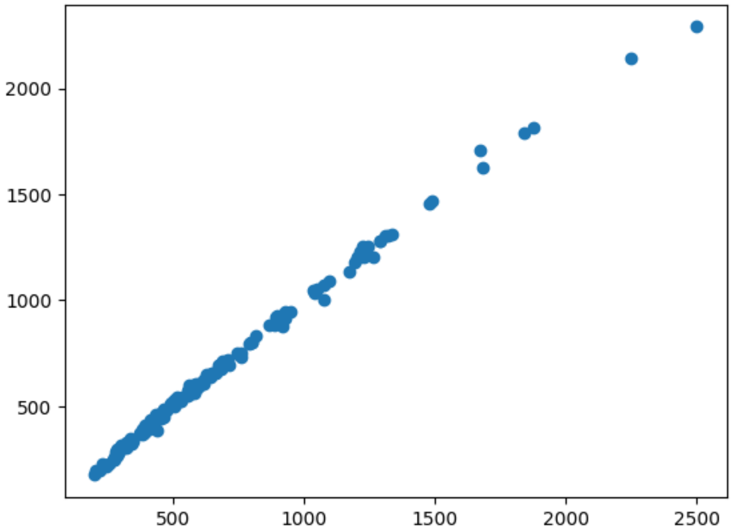
    -303.20237369576057

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

	Coefficient
id	1.534436e-08
diagnosis	5.474975e+00

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

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



concave points_se -2.821298e+03

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

```
0.9973315510664574
```

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

```
0.9955370544009499
```

area_worst 4.976328e-01

```
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
  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
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

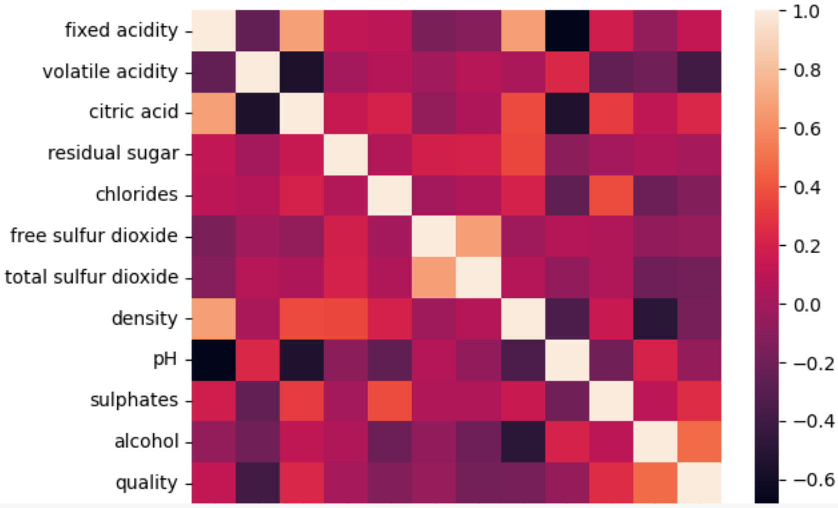
```
df.info()
```

```
<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,:])
```

```
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_
```

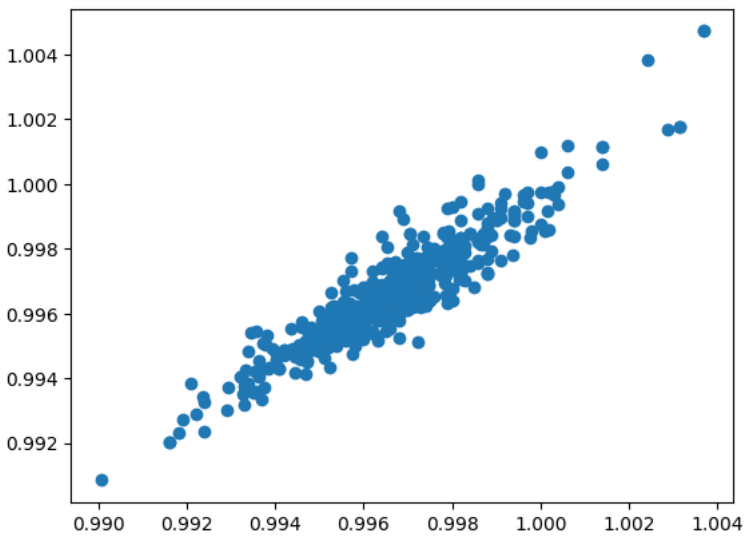
0.9787608689936755

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

	Coefficient
fixed acidity	0.000929
volatile acidity	0.000881
citric acid	0.000249
residual sugar	0.000417
chlorides	0.001950
free sulfur dioxide	-0.000007
total sulfur dioxide	0.000002
pH	0.005167
sulphates	0.001353
alcohol	-0.000908
quality	-0.000001

```
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

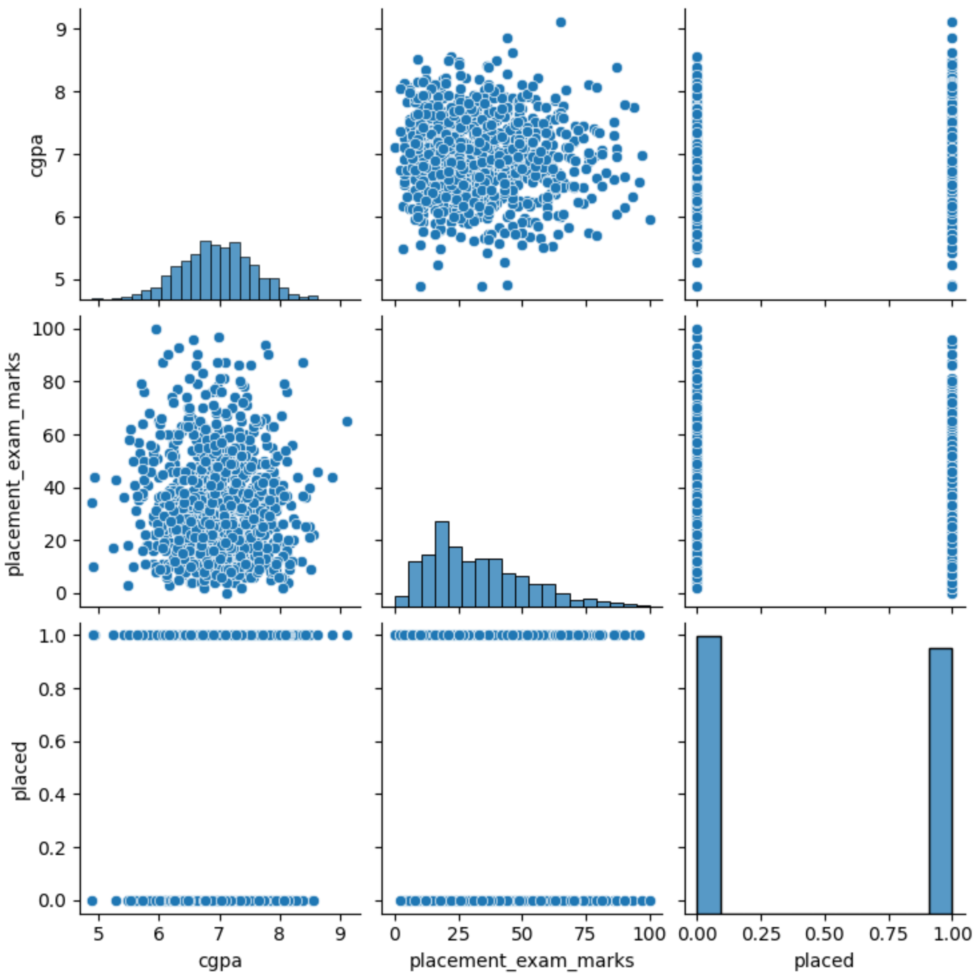
1000 rows × 3 columns

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

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

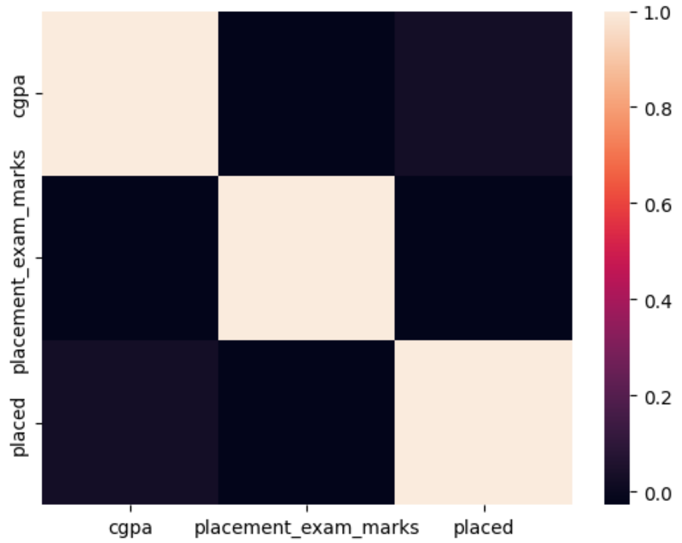
```
sns.pairplot(df)
```

```
<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_
```

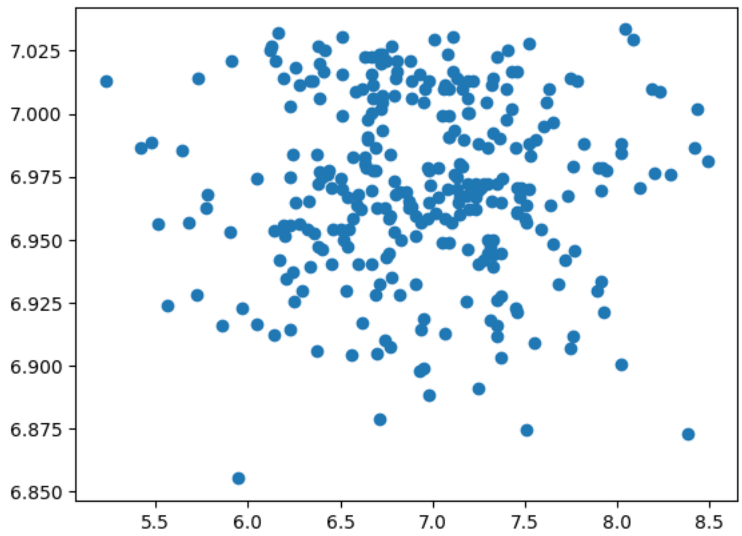
6.992731140418286

```
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)
```

<matplotlib.collections.PathCollection at 0x77ff45c99de0>




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

-0.005696268814112004
0.002959362405151489

```
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)
```

```
-0.005568922940949239
-0.0018765133764020447
```