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

. .

. . .

data=pd.read_csv("/content/cancer_data.csv")
print(data)

								,
→	0		diagnosis			perimeter_mean	_	\
	0	842302	M	17.99	10.38	122.80	1001.0	
	1	842517	M	20.57	17.77	132.90	1326.0	
	2	84300903	М	19.69	21.25	130.00	1203.0	
	3	84348301	М	11.42	20.38	77.58	386.1	
	4	84358402	М	20.29	14.34	135.10	1297.0	
	 564	926424	 M	21.56	 22.39	142.00	 1479.0	
	565	926682	M	20.13	28.25	131.20	1261.0	
	566	926954	M	16.60	28.08	108.30	858.1	
	567	927241	M	20.60	29.33	140.10	1265.0	
	568	92751	В	7.76	24.54	47.92	181.0	
	300	32731	D	7.70	24,54	47.52	101.0	
		smoothnes		•	-	ean concave poi		
	0		0.11840	0.27760	0.300		0.14710	
	1		0.08474	0.07864	0.086		0.07017	
	2	(0.10960	0.15990	0.197	740	0.12790	
	3	(0.14250	0.28390	0.241	140	0.10520	
	4	(0.10030	0.13280	0.198	300	0.10430	
	564	(0.11100	0.11590	0.243	390	0.13890	
	565	(0.09780	0.10340	0.144	100	0.09791	
	566	(0.08455	0.10230	0.092	251	0.05302	
	567	(0.11780	0.27700	0.351	140	0.15200	
	568	(0.05263	0.04362	0.000	000	0.00000	
		tovi	turo worst	perimeter_wors	t area weret	t smoothness_wo	rst \	
	0	tex	ture_worst 17.33	184.6				
		• • •						
	1	• • •	23.41	158.8				
	2	• • •	25.53	152.5				
	3	• • •	26.50	98.8				
	4	• • •	16.67	152.2		0.13		
	 564	• • •	26.40	 166.1) 0.14	100	
	565	• • •	38.25	155.0				
	566	• • •	34.12	126.7				
		• • •	39.42					
	567	• • •		184.6 59.1				
	568	• • •	30.37	59.1	208.0	0.08	1990	
		compactne	ess_worst	concavity_worst	concave poi	ints_worst symm	etry_worst	\
	0	-	0.66560	0.7119		0.2654	0.4601	
	1		0.18660	0.2416		0.1860	0.2750	
	2		0.42450	0.4504		0.2430	0.3613	
	3		0.86630	0.6869		0.2575	0.6638	
	4		0.20500	0.4000		0.1625	0.2364	

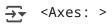
. . .

. . .

564 565 566 567 568	0.21130 0.19220 0.30940 0.86810 0.06444	0.4107 0.3215 0.3403 0.9387 0.0000	0. 0. 0.	2216 1628 1418 2650 0000	0.2060 0.2572 0.2218 0.4087 0.2871
0 1 2 3 4	fractal_dimension_wo 0.11 0.08 0.08 0.17 0.07	890 NaN 902 NaN 758 NaN 300 NaN			
data_fina print(dat	l = data.drop(["id", "l a_final)	Jnnamed: 32","text	ure_se","area_se"	,"compactne	ss_se","concav
0 1 2 3 4	diagnosis radius_mea M 17.9 M 20.5 M 19.6 M 11.4 M 20.2	9 10.38 7 17.77 9 21.25 2 20.38 9 14.34	perimeter_mean 122.80 132.90 130.00 77.58 135.10	1001.0 1326.0 1203.0 386.1 1297.0	\
564 565 566 567 568	M 21.5 M 20.1 M 16.6 M 20.6 B 7.7	6 22.39 3 28.25 0 28.08 0 29.33	142.00 131.20 108.30 140.10 47.92	1479.0 1261.0 858.1 1265.0 181.0	
0 1 2 3 4 564 565 566	0.11840 0.08474 0.10960 0.14250 0.10030 0.11100 0.09780 0.08455	0.27760 0.07864 0.15990 0.28390 0.13280 0.11590 0.10340 0.10230	0.30010 0.08690 0.19740 0.24140 0.19800 0.24390 0.14400 0.09251	concave poi	0.14710 0.07017 0.12790 0.10520 0.10430 0.13890 0.09791 0.05302
567 568 0 1 2 3 4 564 565 566 567 568	0.11780 0.05263 symmetry_mean 0.2419 0.1812 0.2069 0.2597 0.1809 0.1726 0.1752 0.1752 0.1590 0.2397	0.27700 0.04362 radius_se perim 1.0950 0.5435 0.7456 0.4956 0.7572 1.1760 0.7655 0.4564 0.7260 0.3857	3.398 0.4.585 0.3.445 0.5.438 0.7.673 0.5.203 0.3.425 0.5.772 0.5	ness_se col 006399 005225 006150 009110 011490 010300 005769 005903 006522 007189	0.15200 0.00000 ncavity_se 0.05373 0.01860 0.03832 0.05661 0.05688 0.05198 0.03950 0.04730 0.07117 0.00000

0	0.03003	25.380	184.60	0.16220
1	0.01389	24.990	158.80	0.12380
2	0.02250	23.570	152.50	0.14440
3	0.05963	14.910	98.87	0.20980
4	0.01756	22.540	152.20	0.13740
564	0.01114	25.450	166.10	0.14100
565	0.01898	23.690	155.00	0.11660
566	0.01318	18.980	126.70	0.11390
567	0.02324	25.740	184.60	0.16500
568	0.02676	9.456	59.16	0.08996
0 1 2 3	concavity_worst 0.7119 0.2416 0.4504 0.6869 n_4nnn	symmetry_worst 0.4601 0.2750 0.3613 0.6638 0.2364		

sns.heatmap(data_final.isnull())

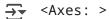


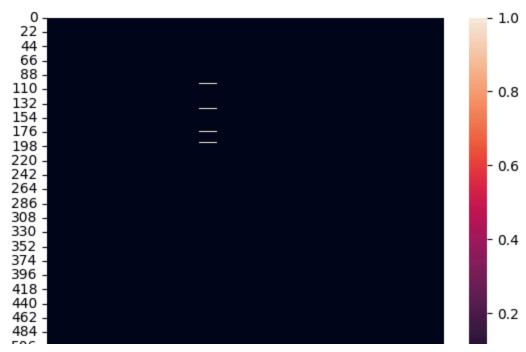


→	0	diagnosis rad M	ius_mean 17.99		_mean 10.38	perimet	er_mean 122.80	area_me		
	0	M					132.90			
	1		20.57		17.77					
	2	M	19.69		21.25		130.00			
	3	M	11.42		20.38		77.58			
	4	M	20.29	ĺ	14.34		135.10	1297	.0	
	 564	 M	21.56	-	22.39		142.00			
	565	M	20.13		28.25		131.20			
	566	M	16.60		28.08		108.30			
	567	М	20.60		29.33		140.10			
	568	В	7.76	4	24.54		47.92	181	.0	
		smoothness_me	an compa	ctness me	ean c	oncavity	/ mean	concave p	oints mean	\
	0	0.118		$0.\overline{277}$			30010	'	0.14710	
	1	0.084		0.078			08690		0.07017	
	2	0.109		0.159			19740		0.12790	
	3	0.142		0.283			24140		0.10520	
	4	0.100		0.132			19800		0.10320	
	4					0.			0.10430	
	564	0.111		0.115		0.	24390		0.13890	
	565	0.097		0.103			14400		0.09791	
	566	0.084		0.102			09251		0.05302	
	567	0.117		0.277			35140		0.15200	
	568	0.052		0.043			00000		NaN	
	300	0.032	.03	0.045	302	0.	.00000		Ivaiv	
		symmetry_mean		dius_se	perim	eter_se			concavity_s	
	0	0.2419		1.0950		8.589		.006399	0.0537	
	1	0.1812		0.5435		3.398	0	.005225	0.0186	0
	2	0.2069		0.7456		4.585	0	.006150	0.0383	2
	3	0.2597	• • •	0.4956		3.445	0	.009110	0.0566	1
	4	0.1809		0.7572		5.438	0	.011490	0.0568	8
	 564	0.1726		 1.1760		7.673	0	.010300	0.0519	
	565	0.1752		0.7655		5.203		.005769	0.0395	
	566	0.1590		0.4564		3.425		.005903	0.0473	
	567	0.2397		0.7260		5.772		.006522	0.0711	
	568	0.1587	• • •	0.3857		2.548	U	.007189	0.0000	U
							· ·			
		symmetrv se	radius wo	rst peri	imeter				\	
	0	-	_	•		_worst		ess_worst		
	0	0.03003	25.	380		_worst 184.60		ess_worst 0.16220		
	1	0.03003 0.01389	25. 24.	380 990		_worst 184.60 158.80		ess_worst 0.16220 0.12380		
	1 2	0.03003 0.01389 0.02250	25. 24. 23.	380 990 570		_worst 184.60 158.80 152.50		0.16220 0.12380 0.14440		
	1 2 3	0.03003 0.01389 0.02250 0.05963	25. 24. 23. 14.	380 990 570 910		_worst 184.60 158.80 152.50 98.87		0.16220 0.12380 0.14440 0.20980		
	1 2 3 4	0.03003 0.01389 0.02250 0.05963 0.01756	25. 24. 23. 14. 22.	380 990 570 910 540		_worst 184.60 158.80 152.50 98.87 152.20		0.16220 0.12380 0.124440 0.20980 0.13740		
	1 2 3 4	0.03003 0.01389 0.02250 0.05963 0.01756	25. 24. 23. 14. 22.	380 990 570 910 540		_worst 184.60 158.80 152.50 98.87 152.20		0.16220 0.12380 0.14440 0.20980 0.13740		
	1 2 3 4 564	0.03003 0.01389 0.02250 0.05963 0.01756 	25. 24. 23. 14. 22.	380 990 570 910 540 		_worst 184.60 158.80 152.50 98.87 152.20 		0.16220 0.12380 0.14440 0.20980 0.13740 		
	1 2 3 4 564 565	0.03003 0.01389 0.02250 0.05963 0.01756 0.01114 0.01898	25. 24. 23. 14. 22.	380 990 570 910 540 450 690		_worst 184.60 158.80 152.50 98.87 152.20 166.10 155.00		0.16220 0.12380 0.14440 0.20980 0.13740 0.14100 0.11660		
	1 2 3 4 564 565 566	0.03003 0.01389 0.02250 0.05963 0.01756 0.01114 0.01898 0.01318	25. 24. 23. 14. 22. 25. 23. 18.	380 990 570 910 540 450 690 980		_worst 184.60 158.80 152.50 98.87 152.20 166.10 155.00 126.70		0.16220 0.12380 0.14440 0.20980 0.13740 0.14100 0.11660 0.11390		
	1 2 3 4 564 565	0.03003 0.01389 0.02250 0.05963 0.01756 0.01114 0.01898	25. 24. 23. 14. 22. 25. 23. 18. 25.	380 990 570 910 540 450 690		_worst 184.60 158.80 152.50 98.87 152.20 166.10 155.00		0.16220 0.12380 0.14440 0.20980 0.13740 0.14100 0.11660		

0	0.7119	0.4601
1	0.2416	0.2750
2	0.4504	0.3613
3	0.6869	0.6638
4	0.4000	0.2364

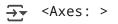
sns.heatmap(data_final.isnull())





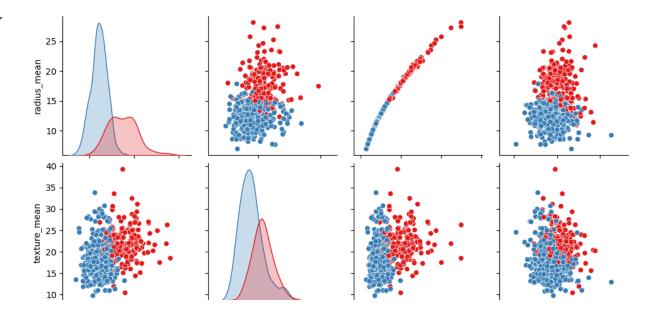
```
data_final.fillna({'texture_mean':data_final['texture_mean'].mean()},inplace=True)
data_final.fillna({'area_mean':data_final['area_mean'].mean()},inplace=True)
data_final.fillna({'compactness_mean':data_final['compactness_mean'].mean()},inplace=True)
data_final.fillna({'concave points_mean':data_final['concave points_mean'].mean()},inplace=
data_final.fillna({'fractal_dimension_mean':data_final['fractal_dimension_mean'].mean()},inplace=
```

sns.heatmap(data_final.isnull())

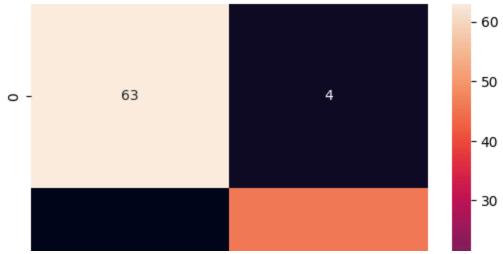




selected_features = ['radius_mean', 'texture_mean', 'area_mean', 'smoothness_mean', 'diagno
sns.pairplot(data[selected_features], hue='diagnosis', palette='Set1')
plt.show()



```
x=data_final.drop('diagnosis', axis=1).values
y=data_final['diagnosis'].values
print(data_final.shape)
→ (569, 31)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=0)
from sklearn.linear_model import LogisticRegression
logm=LogisticRegression()
logm.fit(x_train,y_train)
   /usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: C
    STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.
    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear model.html#logistic-regressio
      n_iter_i = _check_optimize result(
     ▼ LogisticRegression ① ??
    LogisticRegression()
from sklearn.metrics import accuracy score, confusion matrix, classification report
ac=accuracy_score(y_test,logm.predict(x_test))
print(ac)
→ 0.956140350877193
sns.heatmap(confusion matrix(y test,logm.predict(x test)),annot=True)
→ <Axes: >
```



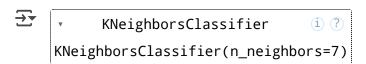
logm_cr=classification_report(y_test,logm.predict(x_test))
print(logm_cr)

₹		precision	recall	f1-score	support
	B	0.98	0.94	0.96	67
	M	0.92	0.98	0.95	47
	accuracy			0.96	114
	macro avg	0.95	0.96	0.96	114
	weighted avg	0.96	0.96	0.96	114

logm_cr=classification_report(y_test,logm.predict(x_test))
print(logm_cr)

₹		precision	recall	f1-score	support
	В	0.98	0.94	0.96	67
	M	0.92	0.98	0.95	47
	accuracy			0.96	114
	macro avg	0.95	0.96	0.96	114
	weighted avg	0.96	0.96	0.96	114

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)

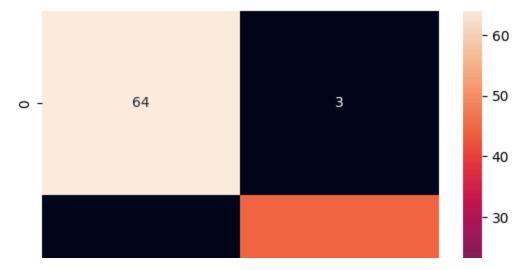


acknn=accuracy_score(y_test,knn.predict(x_test))
print(acknn)

→ 0.9473684210526315

sns.heatmap(confusion_matrix(y_test,knn.predict(x_test)),annot=True)

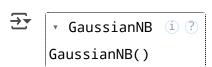
→ <Axes: >



crknn=classification_report(y_test,knn.predict(x_test))
print(crknn)

→	precision	recall	f1-score	support
B M	0.96 0.94	0.96 0.94	0.96 0.94	67 47
accuracy macro avg weighted avg	0.95 0.95	0.95 0.95	0.95 0.95 0.95	114 114 114

from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(x_train,y_train)

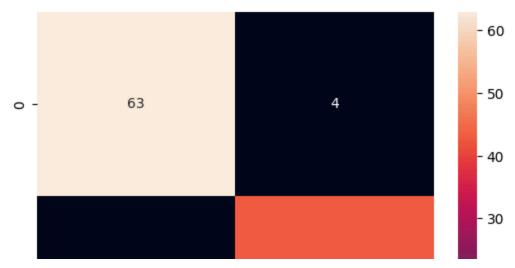


acnb=accuracy_score(y_test,nb.predict(x_test))
print(acnb)

0.9298245614035088

sns.heatmap(confusion_matrix(y_test,nb.predict(x_test)),annot=True)

→ <Axes: >

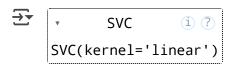


crnb=classification_report(y_test,nb.predict(x_test))
print(crnb)

→	precision	recall	f1-score	support
B M	0.94 0.91	0.94 0.91	0.94 0.91	67 47
accuracy macro avg weighted avg	0.93 0.93	0.93 0.93	0.93 0.93 0.93	114 114 114

SVM

from sklearn.svm import SVC
svmmodel=SVC(kernel="linear")
svmmodel.fit(x_train,y_train)

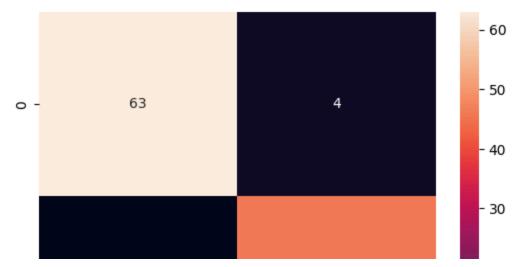


acsvm=accuracy_score(y_test,svmmodel.predict(x_test))
print(acsvm)

→ 0.956140350877193

sns.heatmap(confusion_matrix(y_test,svmmodel.predict(x_test)),annot=True)

→ <Axes: >



crsvm=classification_report(y_test,svmmodel.predict(x_test))
print(crsvm)

→		precision	recall	f1-score	support
	B M	0.98 0.92	0.94	0.96 0.95	67 47
accura		0.32	0.30	0.96	114
macro a weighted a	avg	0.95 0.96	0.96 0.96	0.96 0.96	114 114

< DT

dt.fit(x_train,y_train)

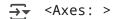


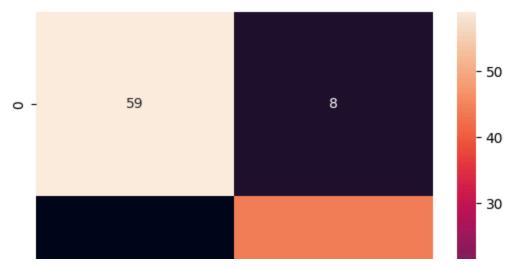
DecisionTreeClassifier ① ??
DecisionTreeClassifier()

acdt=accuracy_score(y_test,dt.predict(x_test))
print(acdt)

→ 0.9035087719298246

sns.heatmap(confusion_matrix(y_test,dt.predict(x_test)),annot=True)





crdt=classification_report(y_test,dt.predict(x_test))
print(crdt)

→		precision	recall	f1-score	support
	В	0.95	0.88	0.91	67
	M	0.85	0.94	0.89	47
	accuracy			0.90	114
	macro avg	0.90	0.91	0.90	114
	weighted avg	0.91	0.90	0.90	114

Random Forest

from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(x_train,y_train)



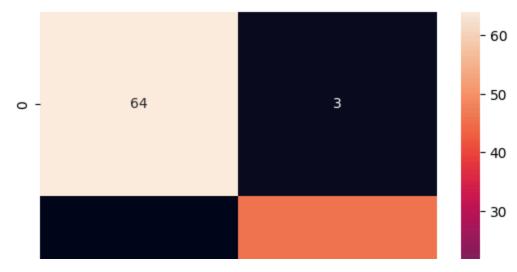
r RandomForestClassifier ① ?
RandomForestClassifier()

acrf=accuracy_score(y_test,rf.predict(x_test))
print(acrf)

0.9649122807017544

sns.heatmap(confusion_matrix(y_test,rf.predict(x_test)),annot=True)

→ <Axes: >



crrf=classification_report(y_test,rf.predict(x_test))
print(crrf)

→	р	recision	recall	f1-score	support
	B	0.98	0.96	0.97	67
	M	0.94	0.98	0.96	47

accuracy			0.96	114
macro avg	0.96	0.97	0.96	114
weighted avg	0.97	0.96	0.97	114