```
%%capture
!pip install scikit-learn numpy pandas
```

```
import os
```

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.7.4)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.5)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kagg
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggl
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

```
!mkdir -p ~/.kaggle
!mv /content/kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
mv: cannot stat '/content/kaggle.json': No such file or directory
```

```
filename = 'breast-cancer-wisconsin-data.zip'
os.makedirs("classification", exist_ok=True)
for root, dirs, file in os.walk("./", topdown=True):
    if filename in file:
        break
else:
    !kaggle datasets download -d uciml/breast-cancer-wisconsin-data
    !mv breast-cancer-wisconsin-data.zip 'classification/'
```

```
import zipfile
with zipfile.ZipFile('classification/breast-cancer-wisconsin-data.zip', 'r') as zip_ref:
    zip_ref.extractall('classification/')
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
dataset=pd.read_csv('classification/data.csv')
dataset.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns

```
dataset.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoo
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	

8 rows × 32 columns

```
dataset.isnull().sum()
```

	0
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

```
df=dataset.drop(columns=["id","Unnamed: 32"],axis="1")
```

```
X=df[df.columns[1:]]
Y=df["diagnosis"]
X.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 30 columns

```
Y=Y.replace({"M":1,"B":0})
Y.tail()
```

	diagnosis
564	1
565	1
566	1
567	1
568	0

dtype: int64

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

Classification

```
X_train,X_temp,Y_train,Y_temp=train_test_split(X,Y,test_size=0.3,random_state=42)
X_val,X_test,Y_val,Y_test=train_test_split(X_temp,Y_temp,test_size=0.7,random_state=42)
lr=LogisticRegression()
X_train.shape,X_val.shape,X_test.shape
```

```
((398, 30), (51, 30), (120, 30))
```

```
log_reg=LogisticRegression(max_iter=10000)
log_reg.fit(X_train,Y_train)
Y_pred=log_reg.predict(X_test)
accuracy_score(Y_test,Y_pred)
```

```
0.9833333333333333
```

SVC

```
svm_clf=SVC(random_state=42)
svm_clf.fit(X_train,Y_train)
Y_pred=svm_clf.predict(X_test)
accuracy_score(Y_test,Y_pred)
```

```
0.9416666666666667
```

Decision Trees

```
tree_clf=DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train,Y_train)
```

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
Y_prediction=tree_clf.predict(X_test)
accuracy_score(Y_test,Y_prediction)
```

```
0.9416666666666667
```

```
# Summary of performance metrics
metrics = {
    'Model': ['Logistic Regression', 'Decision Tree', 'SVM'],
    'Accuracy': [accuracy_score(Y_val, log_reg.predict(X_val)),
                 accuracy_score(Y_val, tree_clf.predict(X_val)),
                 accuracy_score(Y_val, svm_clf.predict(X_val))],
    'Precision': [precision_score(Y_val, log_reg.predict(X_val)),
                  precision_score(Y_val, tree_clf.predict(X_val)),
                  precision_score(Y_val, svm_clf.predict(X_val))],
    'Recall': [recall_score(Y_val, log_reg.predict(X_val)),
               recall_score(Y_val, tree_clf.predict(X_val)),
               recall_score(Y_val, svm_clf.predict(X_val))],
    'F1-Score': [f1_score(Y_val, log_reg.predict(X_val)),
                 f1_score(Y_val, tree_clf.predict(X_val)),
                 f1_score(Y_val, svm_clf.predict(X_val))]
}

metrics_df = pd.DataFrame(metrics)
metrics_df
```

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.960784	1.000000	0.904762	0.950000
1	Decision Tree	0.941176	0.909091	0.952381	0.930233
2	SVM	0.921569	1.000000	0.809524	0.894737

Next steps: Generate code with `metrics_df` | View recommended plots | New interactive sheet