BREAST CANCER DATASET

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

```
from sklearn.datasets import load_breast_cancer
df = load_breast_cancer()
df
```


```
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n--------------------------------------------
\n\n**Data Set Characteristics:**\n\n:Number of Instances: 569\n\n:Number of Attributes: 30 numeric, predictive attributes and the
class\n\n:Attribute Information:\n    - radius (mean of distances from center to points on the perimeter)\n    - texture (standard
deviation of gray-scale values)\n    - perimeter\n    - area\n    - smoothness (local variation in radius lengths)\n    - compactness
(perimeter^2 / area - 1.0)\n    - concavity (severity of concave portions of the contour)\n    - concave points (number of concave
portions of the contour)\n    - symmetry\n    - fractal dimension ("coastline approximation" - 1)\n\n    The mean, standard error,
and "worst" or largest (mean of the three\n    worst/largest values) of these features were computed for each image,\n    resulting
in 30 features.  For instance, field 0 is Mean Radius, field\n    10 is Radius SE, field 20 is Worst Radius.\n\n    - class:\n
- WDBC-Malignant\n            - WDBC-Benign\n\n:Summary Statistics:\n\n===================================== ====== ======\n
Min    Max\n===================================== ====== ======\nradius (mean):                       6.981  28.11\ntexture (mean):
9.71   39.28\nperimeter (mean):                    43.79  188.5\narea (mean):                          143.5  2501.0\nsmoothness
(mean):                   0.053  0.163\ncompactness (mean):                 0.019  0.345\nconcavity (mean):
0.0    0.427\nconcave points (mean):               0.0    0.201\nsymmetry (mean):                      0.106  0.304\nfractal
dimension (mean):             0.05   0.097\nradius (standard error):              0.112  2.873\ntexture (standard error):
0.36   4.885\nperimeter (standard error):          0.757  21.98\narea (standard error):               6.802  542.2\nsmoothness
(standard error):          0.002  0.031\ncompactness (standard error):        0.002  0.135\nconcavity (standard error):
0.0    0.396\nconcave points (standard error):      0.0    0.053\nsymmetry (standard error):            0.008  0.079\nfractal
dimension (standard error):   0.001  0.03\nradius (worst):                       7.93   36.04\ntexture (worst):
12.02  49.54\nperimeter (worst):                   50.41  251.2\narea (worst):                         185.2  4254.0\nsmoothness
(worst):                   0.071  0.223\ncompactness (worst):                 0.027  1.058\nconcavity (worst):
0.0    1.252\nconcave points (worst):              0.0    0.291\nsymmetry (worst):                     0.156  0.664\nfractal
dimension (worst):            0.055  0.208\n===================================== ====== ======\n\n:Missing Attribute Values:
None\n\n:Class Distribution: 212 - Malignant, 357 - Benign\n\n:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\n:Donor: Nick Street\n\n:Date: November, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass.
They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtained
using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of the
4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which uses
linear\nprogramming to construct a decision tree.  Relevant features\nwere selected using an exhaustive search in the space of 1-
4\nfeatures and 1-3 separating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional space
is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable
Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:\n\nftp
ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. dropdown:: References\n\n  - W.N. Street, W.H. Wolberg and O.L.
Mangasarian. Nuclear feature extraction\n    for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on\n    Electronic
Imaging: Science and Technology, volume 1905, pages 861-870,\n    San Jose, CA, 1993.\n  - O.L. Mangasarian, W.N. Street and W.H.
Wolberg. Breast cancer diagnosis and\n    prognosis via linear programming. Operations Research, 43(4), pages 570-577,\n    July-
August 1995.\n  - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n    to diagnose breast cancer from
fine-needle aspirates. Cancer Letters 77 (1994)\n    163-171.\n',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
```


The dataset which is loaded from sklearn is in the form of dictionary. It cant be accessed as normal normal dataset.

```
features = pd.DataFrame(df.data, columns=df.feature_names)
features
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16

569 rows × 30 columns

```
target = pd.DataFrame(df.target, columns=['Target'])
target
```

	Target
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows × 1 columns

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

```
features.isnull().sum()
```

	0
mean radius	0
mean texture	0
mean perimeter	0
mean area	0
mean smoothness	0
mean compactness	0
mean concavity	0
mean concave points	0
mean symmetry	0
mean fractal dimension	0
radius error	0
texture error	0
perimeter error	0
area error	0
smoothness error	0
compactness error	0
concavity error	0
concave points error	0
symmetry error	0
fractal dimension error	0
worst radius	0
worst texture	0
worst perimeter	0
worst area	0
worst smoothness	0
worst compactness	0
worst concavity	0
worst concave points	0
worst symmetry	0
worst fractal dimension	0

dtype: int64

```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
```

```
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

In dataset breast cancer from sklearn there no missing values.

```
features.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	w ra
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.26
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.83
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.93
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.01
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.97
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.79
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.04

8 rows × 30 columns

```
features.shape
```

```
(569, 30)
```

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
features = sc.fit_transform(features)
```

To ensure the quality of the breast cancer dataset, preprocessing steps are essential. Models like k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM) rely on the distances between data points for predictions. Therefore, feature scaling is applied to make sure all features are on the same scale, allowing the models to perform effectively.

In this dataset, there are no missing values. However, if there were missing values that went unchecked, it could lead to errors and result in inaccurate predictions. Thus, it's important to always check for missing values to maintain data integrity and ensure reliable results.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

LOGISTIC REGRESSION -

Logistic regression is a method used to predict two possible outcomes, like whether a tumor is cancerous or not. It uses a special formula to give a probability for each class, making it good for binary problems like cancer diagnosis.

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
```

LogisticRegression

```
LogisticRegression()
```

DECISION TREE -

A decision tree is a tool that helps make decisions by splitting data into branches based on different features. It looks like a tree, where each branch represents a choice, leading to a final decision.

```
from sklearn.tree   import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

DecisionTreeClassifier

```
DecisionTreeClassifier()
```

RANDOM FOREST -

Random forests use many decision trees together to improve accuracy. Instead of just one tree, they combine the results of several trees to make better predictions.

Double-click (or enter) to edit

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array was ex
  return fit_method(estimator, *args, **kwargs)
```

RandomForestClassifier

```
RandomForestClassifier()
```

SUPPORT VECTOR MACHINE (SVM) -

Support vector machinw is a method that finds the best line to separate two classes. It tries to create a clear gap between different groups in the data.

```
from sklearn.svm import SVC

svm = SVC()
svm.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
```

SVC

```
SVC()
```

K-NEAREST NEIGHBOURS (KNN) -

KNN is a simple algorithm that classifies a point by looking at the nearest points around it. It checks the majority class of its closest neighbors to decide what class it belongs to.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:238: DataConversionWarning: A column-vector y was passed wh
  return self._fit(X, y)
```

KNeighborsClassifier

```
KNeighborsClassifier()
```

```
from sklearn.metrics import accuracy_score , confusion_matrix , classification_report

models ={}

models['Logistic Regression'] = lr
models['Decision Tree'] = dt
models['Random Forest'] = rf
models['KNN'] = knn
models['SVM'] = svm

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(name)
    print(accuracy_score(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
    print()
```

```
Decision Tree
0.9298245614035088
[[39  4]
 [ 4 67]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        43
           1       0.94      0.94      0.94        71

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114


Random Forest
0.9649122807017544
[[40  3]
 [ 1 70]]
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

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


KNN
0.9473684210526315
[[40  3]
 [ 3 68]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

```
        macro avg       0.97      0.97      0.97       114
     weighted avg       0.97      0.97      0.97       114
```

```
best_model = max(models)
print("Best Model:", best_model)

worst_model = min(models)
print("Worst Model:", worst_model)
```