```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegressio
from sklearn.model_selection import train_test_s
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error,
```

```
from sklearn.linear_model import LogisticRegress
# Import the datasets module from sklearn
from sklearn import datasets
# Access load_breast_cancer directly from the im
data = datasets.load_breast_cancer()
```

```
data
```

Show hidden output

```
data.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',
       'worst perimeter', 'worst area',
'worst smoothness',
       'worst compactness', 'worst
concavity', 'worst concave points',
       'worst symmetry', 'worst fractal
dimension'], dtype='<U23')
```

```
data.target
```

ValueError NameError •••

Please explain the error:

```
NameError: name 'sklearn
```

Hypothesis

The traceback indicates a `NameError: name 'sklearn' is not defined`. This error occurs when you try to use a variable or module name that hasn't been properly imported or defined in the current scope.

In the provided code, you are trying to access `sklearn.datasets.load_bre` While you have imported `datasets` from `sklearn` using `from sklearn import datasets`, this import statement only makes the `datasets` module available directly. It does not make the top-level `sklearn` package itself available under the name `sklearn`. Therefore, when you try to use `sklearn.datasets`, the name `sklearn` is not recognized.

Suggested Changes

To fix this, you need to import the entire `sklearn` package directly or use the imported `datasets` module correctly. The most

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0,
0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1,
1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       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, 0, 0, 1, 0, 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, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 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,
0, 0, 0, 0, 0, 0, 1])
```

common way to use modules from `sklearn` is to import the specific module or function you need directly, as you have done with `datasets`.

You can either change the line `data= sklearn.datasets.load_brea` to use the imported `datasets` directly, or import `sklearn` as a whole. The former is generally preferred as it keeps the namespace cleaner.

```
from sklearn.linear_mode
# Import the datasets mod
from sklearn import datas
# Access load_breast_canc
data = datasets.load_brea
```

Use code with caution

Rate this answer

```
df=pd.DataFrame(data.data,columns=data.feature_n
```

```
df.head(20)
```

worst exture	worst perimeter	worst area	worst smoothness	wor compactne
17.33	184.60	2019.0	0.1622	0.66
23.41	158.80	1956.0	0.1238	0.18
25.53	152.50	1709.0	0.1444	0.42
26.50	98.87	567.7	0.2098	0.86
16.67	152.20	1575.0	0.1374	0.20
23.75	103.40	741.6	0.1791	0.52
27.66	153.20	1606.0	0.1442	0.25
28.14	110.60	897.0	0.1654	0.36
30.73	106.20	739.3	0.1703	0.54
40.68	97.65	711.4	0.1853	1.05
33.88	123.80	1150.0	0.1181	0.15
27.28	136.50	1299.0	0.1396	0.56
29.94	151.70	1332.0	0.1037	0.39
27.66	112.00	876.5	0.1131	0.19
32.01	108.80	697.7	0.1651	0.77
37.13	124.10	943.2	0.1678	0.65
30.88	123.40	1138.0	0.1464	0.18
31.48	136.80	1315.0	0.1789	0.42
30.88	186.80	2398.0	0.1512	0.31
19.26	99.70	711.2	0.1440	0.17

```
df['target']=data.target
```

```
df.isnull().sum()
```

Show hidden output

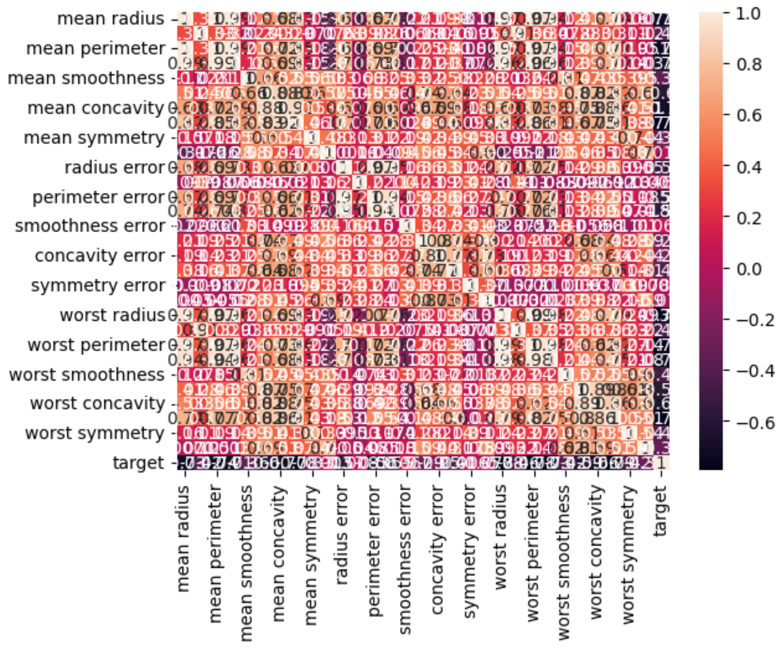
```
df['target'].value_counts()
```

```
        count
target
     1    357
     0    212

dtype: int64
```

```
sns.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```



```
df.shape
```

```
(569, 31)
```

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

```
x
```

Show hidden output

```
y
```

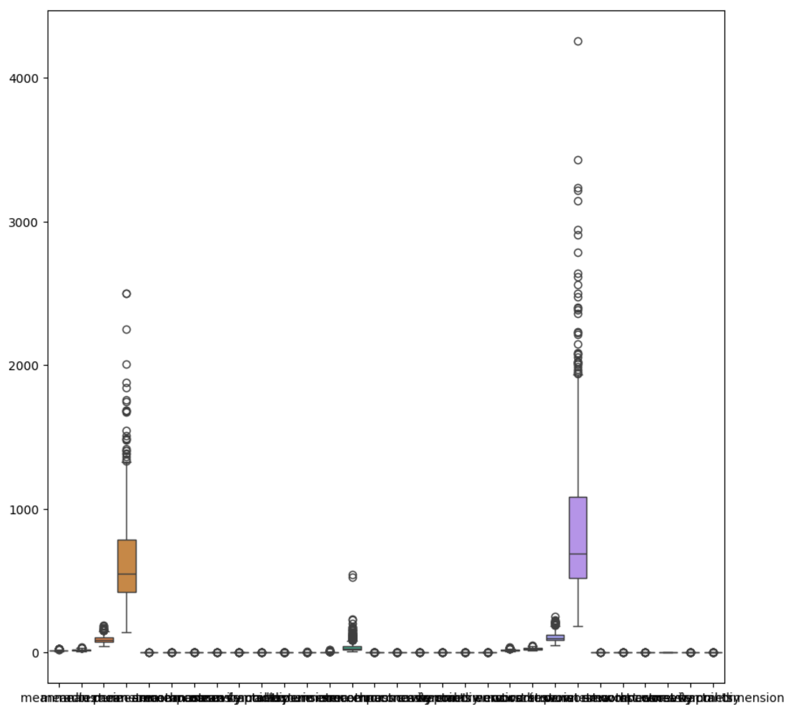
	target
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows × 1 columns

dtype: int64

```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(data=x, ax=ax)
```

```
<Axes: >
```

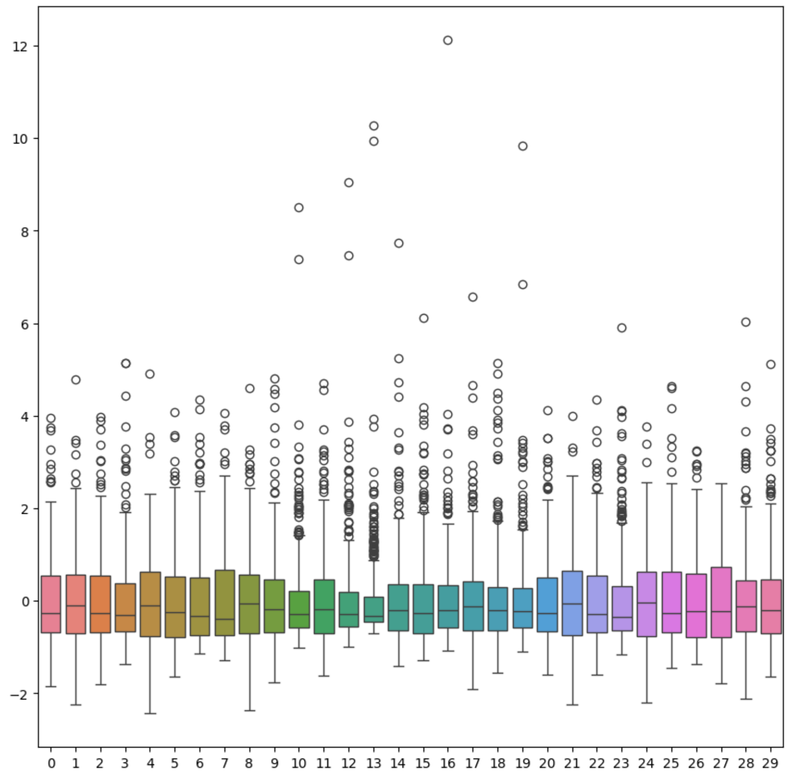


```
x_train, x_test, y_train, y_test = train_test_sp
```

```
scaler = StandardScaler()
x_train_norm = scaler.fit_transform(x_train)
x_test_norm = scaler.transform(x_test)
```

```
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(data=x_train_norm, ax=ax)
```

```
<Axes: >
```

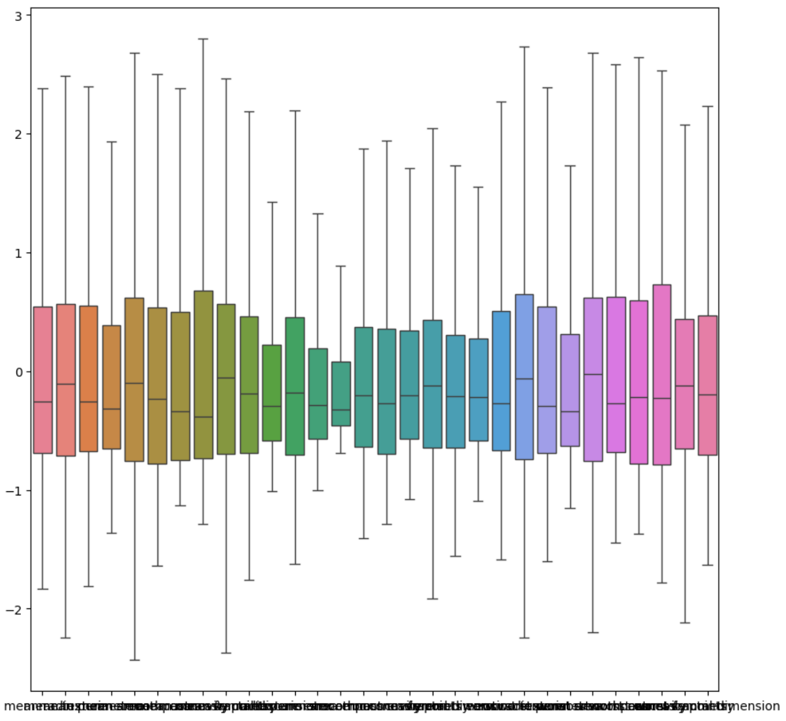


```
x_train_df = pd.DataFrame(x_train_norm, columns=

# Use IQR method to cap outliers across all colu
for col in x_train_df.columns:
    Q1 = x_train_df[col].quantile(0.25)
    Q3 = x_train_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    x_train_df[col] = np.where(x_train_df[col] >
                               np.where(x_train_


fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(data=x_train_df, ax=ax)
```

```
<Axes: >
```



```
from sklearn.linear_model import LogisticRegress
classification = LogisticRegression()
classification.fit(x_train_df, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_
# Check resampled class distribution
print("\nResampled class distribution:")
print(pd.Series(y_train_resampled).value_counts()
# Check resampled class distribution
```

```

Resampled class distribution:
target
1    249
0    249
Name: count, dtype: int64
```

```
y_pred = classification.predict(x_test_norm)
```

```
/usr/local/lib/python3.11/dist-packages/skle
  warnings.warn(
```

```
from sklearn.metrics import accuracy_score
accuracy_score (y_test,y_pred)*100
```

```
98.24561403508771
```

Enter a prompt here

0 / 2000