```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
uploaded =files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving data.csv to data.csv

```
df=pd.read_csv("data.csv")
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0

5 rows × 32 columns

```
df.head()
```


	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0

5 rows × 32 columns


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave_points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave_points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave_points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
```

```
memory usage: 142.4+ KB
```

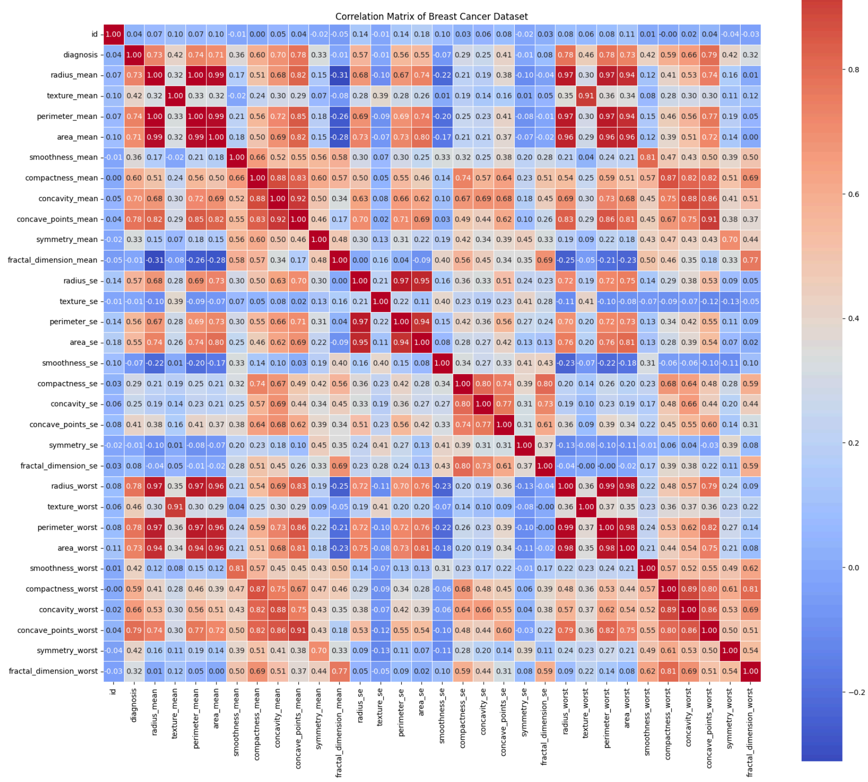
```
df['diagnosis'].unique()
```

```
array(['M', 'B'], dtype=object)
```

```
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
```

```
correlation_matrix = df.corr()
```

```
plt.figure(figsize=(20, 20))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, linewidths=0.5)
plt.title('Correlation Matrix of Breast Cancer Dataset')
plt.show()
```

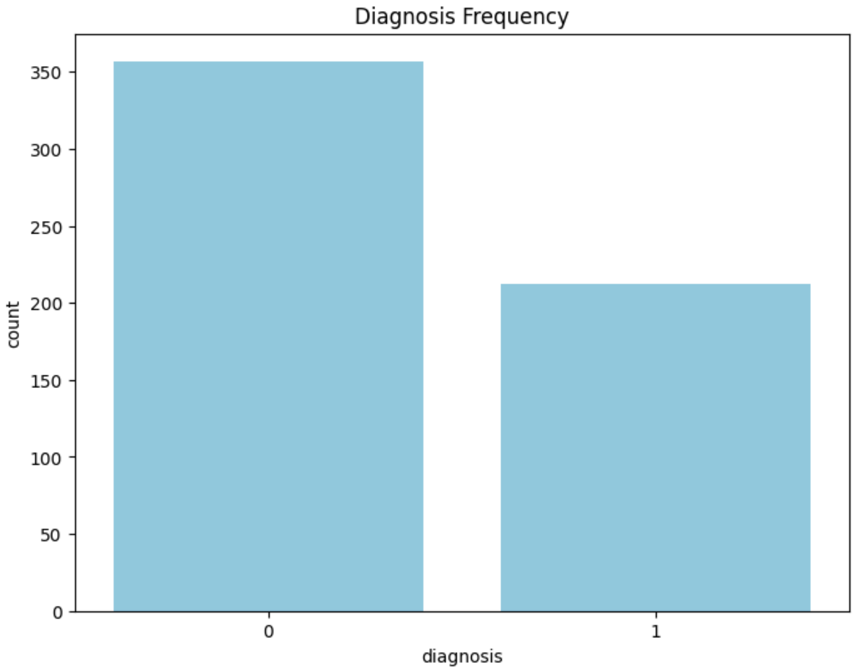

Correlation Matrix of Breast Cancer Dataset
id
diagnosis
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave_points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave_points_worst
symmetry_worst
fractal_dimension_worst
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

The heatmap shows how features in the Breast Cancer dataset are related to each other. Correlation values range from -1 to 1, where -1 means a perfect negative relationship, 0 means no relationship, and 1 means a perfect positive relationship.

High correlation: Some features are almost perfectly related, like "mean radius" and "mean perimeter" (0.997). You can choose just one for analysis to avoid redundancy. Low correlation: Some features are almost unrelated, like "mean symmetry" and "mean fractal dimension" (0.008). These independent features can be useful for building accurate models.

```
# Distribution Plots
sns.pairplot(df, hue='diagnosis', vars=['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean'])
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.countplot(x='diagnosis', data=df, color='skyblue')
plt.title('Diagnosis Frequency')
plt.show()
```




```
threshold = 0.5
high_corr_features = correlation_matrix.index[abs(correlation_matrix['diagnosis']) > threshold].tolist()
high_corr_features.remove('diagnosis')

len(high_corr_features)
high_corr_features
```

```
['radius_mean',
 'perimeter_mean',
 'area_mean',
 'compactness_mean',
 'concavity_mean',
 'concave_points_mean',
 'radius_se',
 'perimeter_se',
 'area_se',
 'radius_worst',
 'perimeter_worst',
 'area_worst',
 'compactness_worst',
 'concavity_worst',
 'concave_points_worst']
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
X = df[high_corr_features]
y = df['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

```
logistecReegression_y_pred = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
accuracy = accuracy_score(y_test, logistecReegression_y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.9824561403508771
```

```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
```

```
knn_y_pred = knn_model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, knn_y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.9298245614035088
```

```
numerical_cols = [col for col in numerical_cols if 'id' not in col]
id_ = "id"
if id_ in numerical_cols:
```