BREAST CANCER CLASSIFICATION

Importing Libraries and Loading Dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
```

In [2]:
```
df=pd.read_csv("/content/BREAST CANCER PREDICTION.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.10
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11
565	926682	M	20.13	28.25	131.20	1261.0	0.09
566	926954	M	16.60	28.08	108.30	858.1	0.08
567	927241	M	20.60	29.33	140.10	1265.0	0.11
568	92751	B	7.76	24.54	47.92	181.0	0.05

569 rows × 33 columns

In [3]:
```
df.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003

5 rows × 33 columns

In [4]:
```
df.tail()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
564	926424	M	21.56	22.39	142.00	1479.0	0.111(
565	926682	M	20.13	28.25	131.20	1261.0	0.0978
566	926954	M	16.60	28.08	108.30	858.1	0.0845
567	927241	M	20.60	29.33	140.10	1265.0	0.1178
568	92751	B	7.76	24.54	47.92	181.0	0.0526

5 rows × 33 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
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
```

In [7]:
```
df.dtypes
```

Out[7]:
```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

Data Preprocessing

In [8]:
```
#Dropping unnecessaary features

df.drop(['id','Unnamed: 32'],axis=1,inplace=True)

#Dropping duplicates if there is any

df.drop_duplicates()
```

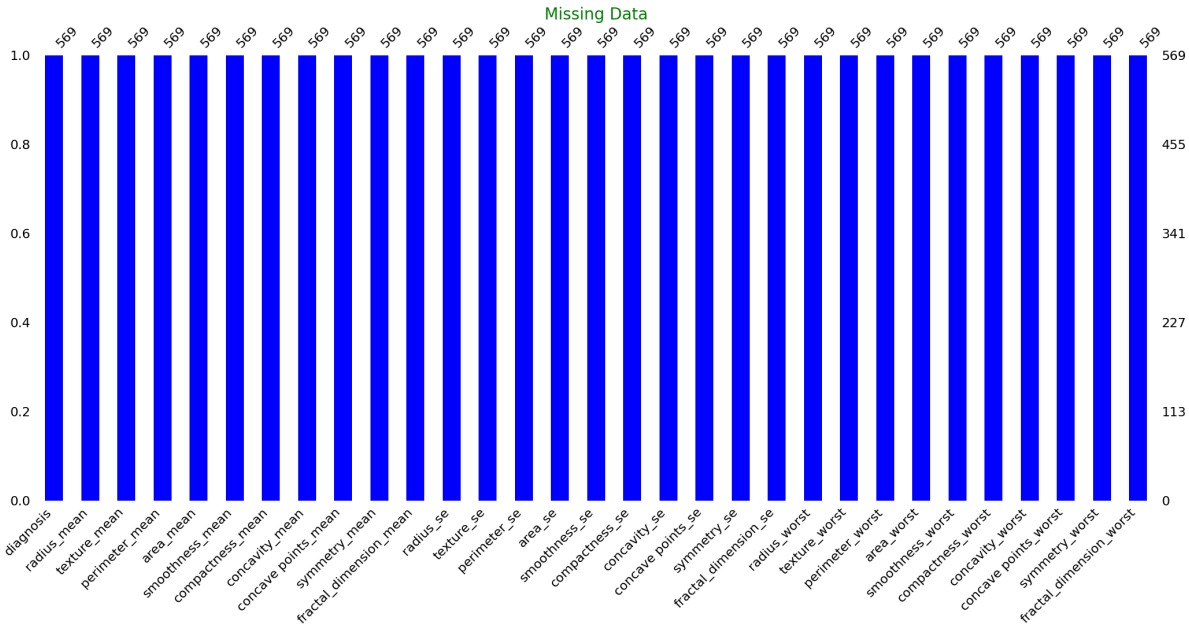
Out[8]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	
565	M	20.13	28.25	131.20	1261.0	0.09780	
566	M	16.60	28.08	108.30	858.1	0.08455	
567	M	20.60	29.33	140.10	1265.0	0.11780	
568	B	7.76	24.54	47.92	181.0	0.05263	

569 rows × 31 columns

In [9]:

```
msno.bar(df,color='blue')
plt.title("Missing Data",color='green',fontsize=20)
plt.show()
```



In [10]:

```
df['diagnosis']=df['diagnosis'].map({"M":"Malignant","B":"Bengin"})
df
```

Out[10]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	Malignant	17.99	10.38	122.80	1001.0	0.11840	
1	Malignant	20.57	17.77	132.90	1326.0	0.08474	
2	Malignant	19.69	21.25	130.00	1203.0	0.10960	
3	Malignant	11.42	20.38	77.58	386.1	0.14250	
4	Malignant	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	
564	Malignant	21.56	22.39	142.00	1479.0	0.11100	
565	Malignant	20.13	28.25	131.20	1261.0	0.09780	
566	Malignant	16.60	28.08	108.30	858.1	0.08455	
567	Malignant	20.60	29.33	140.10	1265.0	0.11780	
568	Bengin	7.76	24.54	47.92	181.0	0.05263	

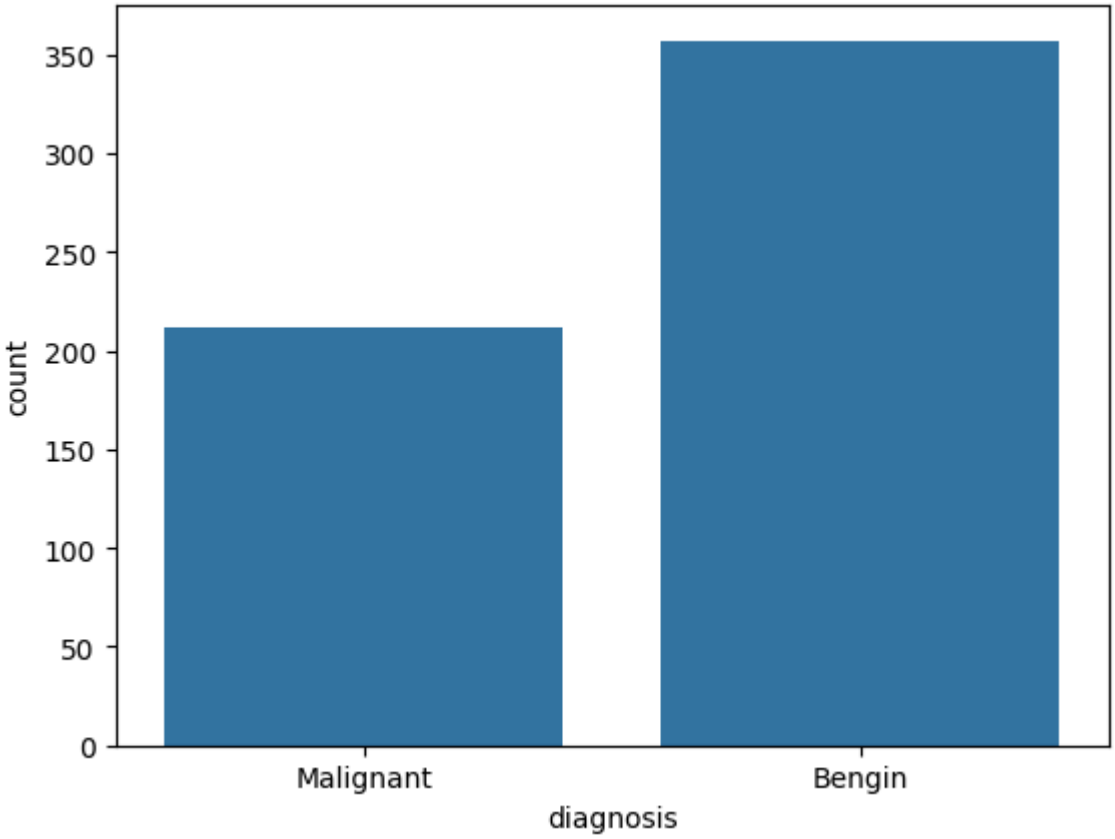
569 rows × 31 columns

In [11]:
```
df['diagnosis'].value_counts()
```

Out[11]:
```
diagnosis
Bengin       357
Malignant    212
Name: count, dtype: int64
```

In [12]:
```
sns.countplot(x='diagnosis',data=df)
```

Out[12]:
```
<Axes: xlabel='diagnosis', ylabel='count'>
```



In [13]:
```
df1=df.drop(['diagnosis'],axis=1)
df1.corr()
```

Out[13]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
radius_mean	1.000000	0.323782	0.997855	0.987357	0.17058
texture_mean	0.323782	1.000000	0.329533	0.321086	-0.02338
perimeter_mean	0.997855	0.329533	1.000000	0.986507	0.20727
area_mean	0.987357	0.321086	0.986507	1.000000	0.17702
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1.00000
compactness_mean	0.506124	0.236702	0.556936	0.498502	0.65912
concavity_mean	0.676764	0.302418	0.716136	0.685983	0.52198
concave points_mean	0.822529	0.293464	0.850977	0.823269	0.55369
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.55777
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	0.58479
radius_se	0.679090	0.275869	0.691765	0.732562	0.30146
texture_se	-0.097317	0.386358	-0.086761	-0.066280	0.06840
perimeter_se	0.674172	0.281673	0.693135	0.726628	0.29609
area_se	0.735864	0.259845	0.744983	0.800086	0.24655
smoothness_se	-0.222600	0.006614	-0.202694	-0.166777	0.33237
compactness_se	0.206000	0.191975	0.250744	0.212583	0.31894
concavity_se	0.194204	0.143293	0.228082	0.207660	0.24839
concave points_se	0.376169	0.163851	0.407217	0.372320	0.38067
symmetry_se	-0.104321	0.009127	-0.081629	-0.072497	0.20077
fractal_dimension_se	-0.042641	0.054458	-0.005523	-0.019887	0.28360
radius_worst	0.969539	0.352573	0.969476	0.962746	0.21312
texture_worst	0.297008	0.912045	0.303038	0.287489	0.03607
perimeter_worst	0.965137	0.358040	0.970387	0.959120	0.23885
area_worst	0.941082	0.343546	0.941550	0.959213	0.20671
smoothness_worst	0.119616	0.077503	0.150549	0.123523	0.80532
compactness_worst	0.413463	0.277830	0.455774	0.390410	0.47246
concavity_worst	0.526911	0.301025	0.563879	0.512606	0.43492
concave points_worst	0.744214	0.295316	0.771241	0.722017	0.50305
symmetry_worst	0.163953	0.105008	0.189115	0.143570	0.39430
fractal_dimension_worst	0.007066	0.119205	0.051019	0.003738	0.49931

30 rows × 30 columns

In [ ]:

```
plt.figure(figsize=(20,20))
sns.heatmap(df1.corr(),annot=True)
plt.show()
```

In [ ]:
```
x=df.drop('diagnosis',axis=1).values
x
```

In [ ]:
```
y=df['diagnosis'].values
y
```

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train
```

In [ ]:
```
x_test
```

In [ ]:
```
y_train
```

In [ ]:
```
y_test
```

Normalization

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

Model Creation and Performance Evaluation

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report
knn=KNeighborsClassifier(n_neighbors=7)
dectree=DecisionTreeClassifier(random_state=42)
ranfor=RandomForestClassifier(n_estimators=100,random_state=42)
logreg=LogisticRegression()
lst=[knn,dectree,ranfor,logreg]
```

In [ ]:
```
for i in lst:
 i.fit(x_train,y_train)
 y_pred=i.predict(x_test)
 print("R2_score of",i,"model is",accuracy_score(y_test,y_pred))
 print(classification_report(y_test,y_pred))
```