In [2]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input dire

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you 
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/breast-cancer-wisconsin-data/data.csv
```

In [3]:

```
df = pd.read_csv("/kaggle/input/breast-cancer-wisconsin-data/data.csv")
df.head(10)
```

```
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
```

Out [3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	co points_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.1471
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.0701
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.1279
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.1052
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.1043
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.0808
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.0740
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.0598
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.0935
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.0854

10 rows × 33 columns

In [4]:

```
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 [5]:

```
df.drop(columns=["Unnamed: 32","id"],inplace = True)
```

In [6]:

```
df.duplicated().sum()
```

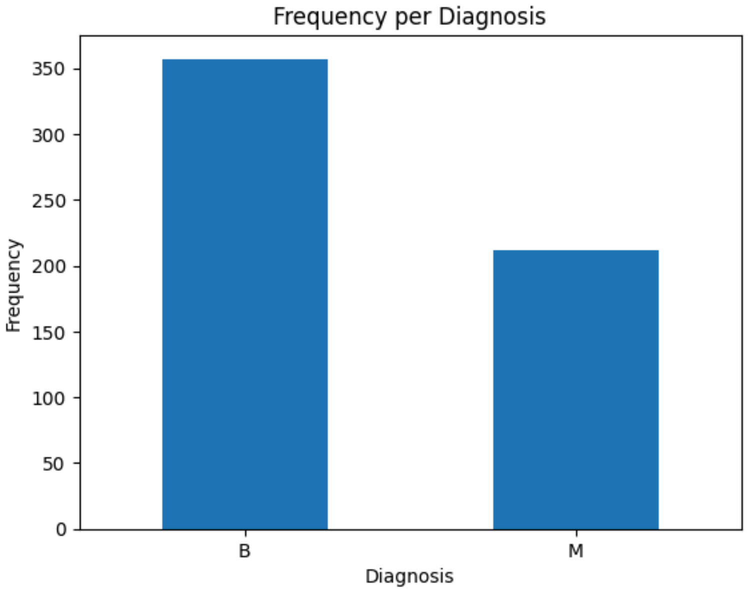
Out [6]:

```
0
```

clean data and ready to be visualized

In [7]:

```
import matplotlib.pyplot as plt

df["diagnosis"].value_counts().plot(kind="bar")
plt.xlabel("Diagnosis")
plt.ylabel("Frequency")
plt.title("Frequency per Diagnosis")
plt.xticks(rotation=0)
plt.show()
df["diagnosis"].value_counts()
```



Out [7]:

```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

mmm i smell a Class imbalance

In [9]:

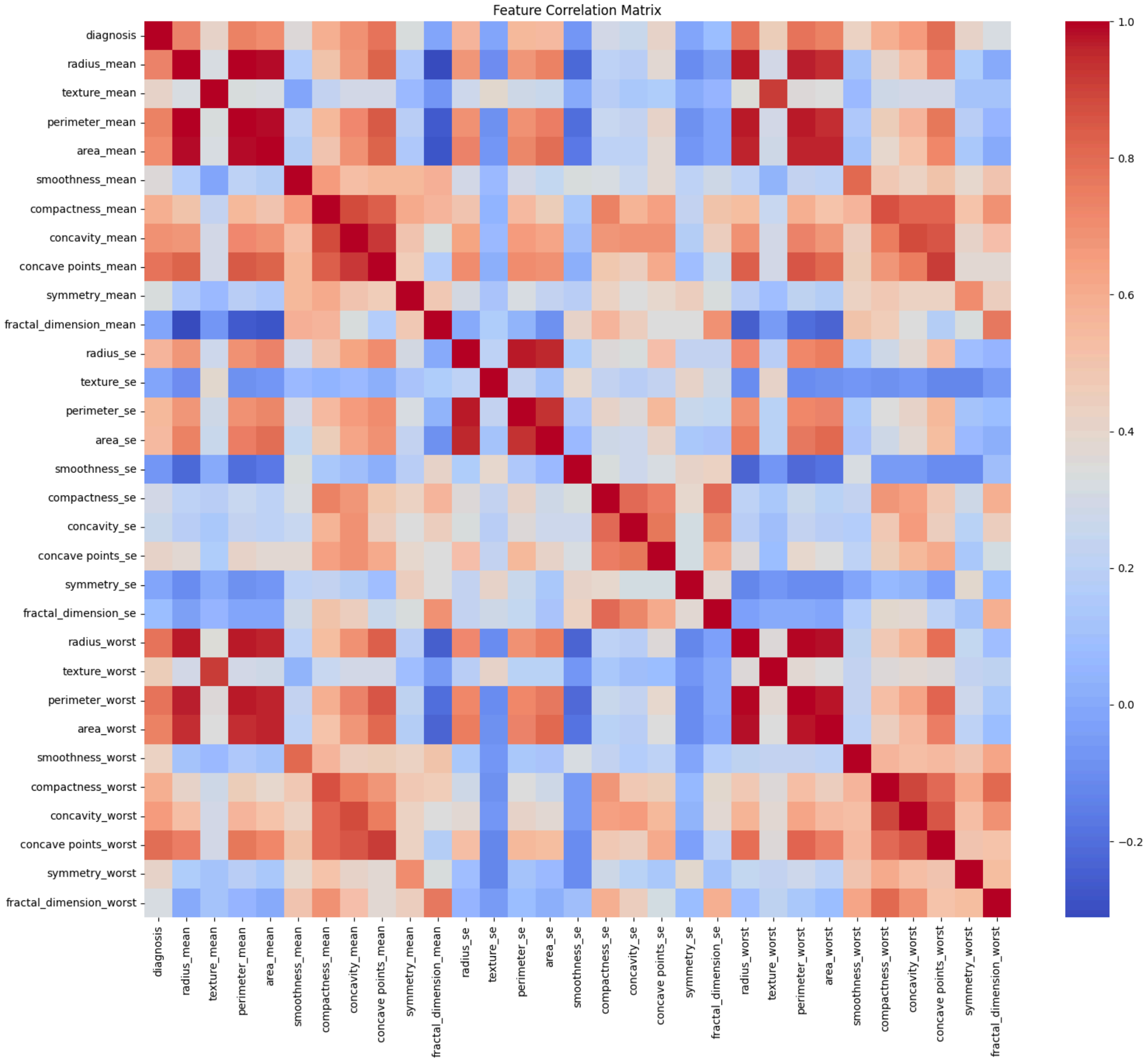
```
## load & encode target
from sklearn.preprocessing import LabelEncoder, StandardScaler

df["diagnosis"] = LabelEncoder().fit_transform(df["diagnosis"])
df["diagnosis"]
```

Out [9]:

```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

In [10]:

```
## Correlation heatmap
import seaborn as sns
plt.figure(figsize=(18, 15))
sns.heatmap(df.corr(), annot=False, cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()
```



In [12]:

```
## Preprocessing
from sklearn.model_selection import train_test_split, cross_val_score

x = df.drop(columns="diagnosis")
y = df["diagnosis"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state = 42)
```

In [15]:

```
## Model Training
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score

models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(probability=True),
    "KNN": KNeighborsClassifier()
}

for name, model in models.items():
    model.fit(x_train, y_train)
    preds = model.predict(x_test)
    print(f"\n{name}")
    print("Accuracy:", accuracy_score(y_test, preds))
    print("ROC AUC:", roc_auc_score(y_test, model.predict_proba(x_test)[:, 1]))
    print(confusion_matrix(y_test, preds))
    print(classification_report(y_test, preds))
```

```
Logistic Regression
```

```
Accuracy: 0.9736842105263158
ROC AUC: 0.99737962659679
[[70  1]
 [ 2 41]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

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


Random Forest
Accuracy: 0.9649122807017544
ROC AUC: 0.9941041598427777
[[70  1]
 [ 3 40]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        71
           1       0.98      0.93      0.95        43

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


SVM
Accuracy: 0.9736842105263158
ROC AUC: 0.99737962659679
[[70  1]
 [ 2 41]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

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


KNN
Accuracy: 0.9473684210526315
ROC AUC: 0.9816573861775302
[[68  3]
 [ 3 40]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        71
           1       0.93      0.93      0.93        43

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

In [16]:

```python
## Cross-Validation
for name, model in models.items():
    scores = cross_val_score(model, X_scaled, y, cv=5, scoring='accuracy')
    print(f"{name} - CV Accuracy: {scores.mean():.3f}")
```

```
Logistic Regression - CV Accuracy: 0.981
Random Forest - CV Accuracy: 0.961
SVM - CV Accuracy: 0.974
KNN - CV Accuracy: 0.965
```

as we saw all models scored good but the Logistic Regression scored the best out of all of them