Import the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report, confusion_matri
```

Read the Data

In [2]:
```
df = pd.read_csv('brca.csv')
```

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

Out[3]:

	Unnamed: 0	x.radius_mean	x.texture_mean	x.perimeter_mean	x.area_mean	x.smoothness_mean	x.compactne
0	1	13.540	14.36	87.46	566.3	0.09779	
1	2	13.080	15.71	85.63	520.0	0.10750	
2	3	9.504	12.44	60.34	273.9	0.10240	
3	4	13.030	18.42	82.61	523.8	0.08983	
4	5	8.196	16.84	51.71	201.9	0.08600	

5 rows × 32 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            569 non-null    int64  
 1   x.radius_mean         569 non-null    float64
 2   x.texture_mean        569 non-null    float64
 3   x.perimeter_mean      569 non-null    float64
 4   x.area_mean           569 non-null    float64
 5   x.smoothness_mean     569 non-null    float64
 6   x.compactness_mean    569 non-null    float64
 7   x.concavity_mean      569 non-null    float64
 8   x.concave_pts_mean    569 non-null    float64
 9   x.symmetry_mean       569 non-null    float64
 10  x.fractal_dim_mean    569 non-null    float64
 11  x.radius_se           569 non-null    float64
 12  x.texture_se          569 non-null    float64
 13  x.perimeter_se        569 non-null    float64
 14  x.area_se             569 non-null    float64
 15  x.smoothness_se       569 non-null    float64
 16  x.compactness_se      569 non-null    float64
 17  x.concavity_se        569 non-null    float64
 18  x.concave_pts_se      569 non-null    float64
 19  x.symmetry_se         569 non-null    float64
 20  x.fractal_dim_se      569 non-null    float64
 21  x.radius_worst        569 non-null    float64
 22  x.texture_worst       569 non-null    float64
 23  x.perimeter_worst     569 non-null    float64
 24  x.area_worst          569 non-null    float64
 25  x.smoothness_worst    569 non-null    float64
 26  x.compactness_worst   569 non-null    float64
 27  x.concavity_worst     569 non-null    float64
 28  x.concave_pts_worst   569 non-null    float64
 29  x.symmetry_worst      569 non-null    float64
 30  x.fractal_dim_worst   569 non-null    float64
 31  y                     569 non-null    object 
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

Check if we have missing values

In [5]:
```
df.isna().sum()
```

Out[5]:
```
Unnamed: 0              0
x.radius_mean           0
x.texture_mean          0
x.perimeter_mean        0
x.area_mean             0
x.smoothness_mean       0
x.compactness_mean      0
x.concavity_mean        0
x.concave_pts_mean      0
x.symmetry_mean         0
x.fractal_dim_mean      0
x.radius_se             0
x.texture_se            0
x.perimeter_se          0
x.area_se               0
x.smoothness_se         0
x.compactness_se        0
x.concavity_se          0
x.concave_pts_se        0
x.symmetry_se           0
x.fractal_dim_se        0
x.radius_worst          0
x.texture_worst         0
x.perimeter_worst       0
x.area_worst            0
x.smoothness_worst      0
x.compactness_worst     0
x.concavity_worst       0
x.concave_pts_worst     0
x.symmetry_worst        0
x.fractal_dim_worst     0
y                       0
dtype: int64
```

Clean and Prepare the Data

In [6]:
```
df.shape
```

Out[6]:
```
(569, 32)
```

In [7]:
```
df.drop('Unnamed: 0', axis=1, inplace=True)
```

In [8]:
```
print(df.columns)
```

```
Index(['x.radius_mean', 'x.texture_mean', 'x.perimeter_mean', 'x.area_mean',
       'x.smoothness_mean', 'x.compactness_mean', 'x.concavity_mean',
       'x.concave_pts_mean', 'x.symmetry_mean', 'x.fractal_dim_mean',
       'x.radius_se', 'x.texture_se', 'x.perimeter_se', 'x.area_se',
       'x.smoothness_se', 'x.compactness_se', 'x.concavity_se',
       'x.concave_pts_se', 'x.symmetry_se', 'x.fractal_dim_se',
       'x.radius_worst', 'x.texture_worst', 'x.perimeter_worst',
       'x.area_worst', 'x.smoothness_worst', 'x.compactness_worst',
       'x.concavity_worst', 'x.concave_pts_worst', 'x.symmetry_worst',
       'x.fractal_dim_worst', 'y'],
      dtype='object')
```

In [9]:
```
df.rename(columns={'y': 'cancer_type'}, inplace= True)
```

In [10]:
```
df.cancer_type.unique()
```

Out[10]:
```
array(['B', 'M'], dtype=object)
```

In [11]:
```
df['cancer_type'], _ = pd.factorize(df['cancer_type'])
```

In [12]:
```
df["cancer_type"].value_counts()
```

Out[12]:
```
0    357
1    212
Name: cancer_type, dtype: int64
```

Explore the Data

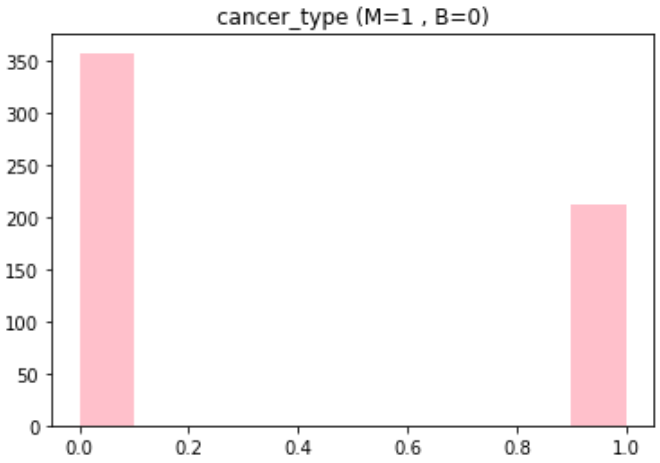
In [13]:
```
df.describe()
```

Out[13]:

	x.radius_mean	x.texture_mean	x.perimeter_mean	x.area_mean	x.smoothness_mean	x.compactness_mea
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.10434
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.05281
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.01938
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.06492
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.09263
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.13040
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.34540

8 rows × 31 columns

In [14]:
```
hist_color='pink'
df.describe()
plt.hist(df['cancer_type'], color=hist_color)
plt.title('cancer_type (M=1 , B=0)')
plt.show()
```



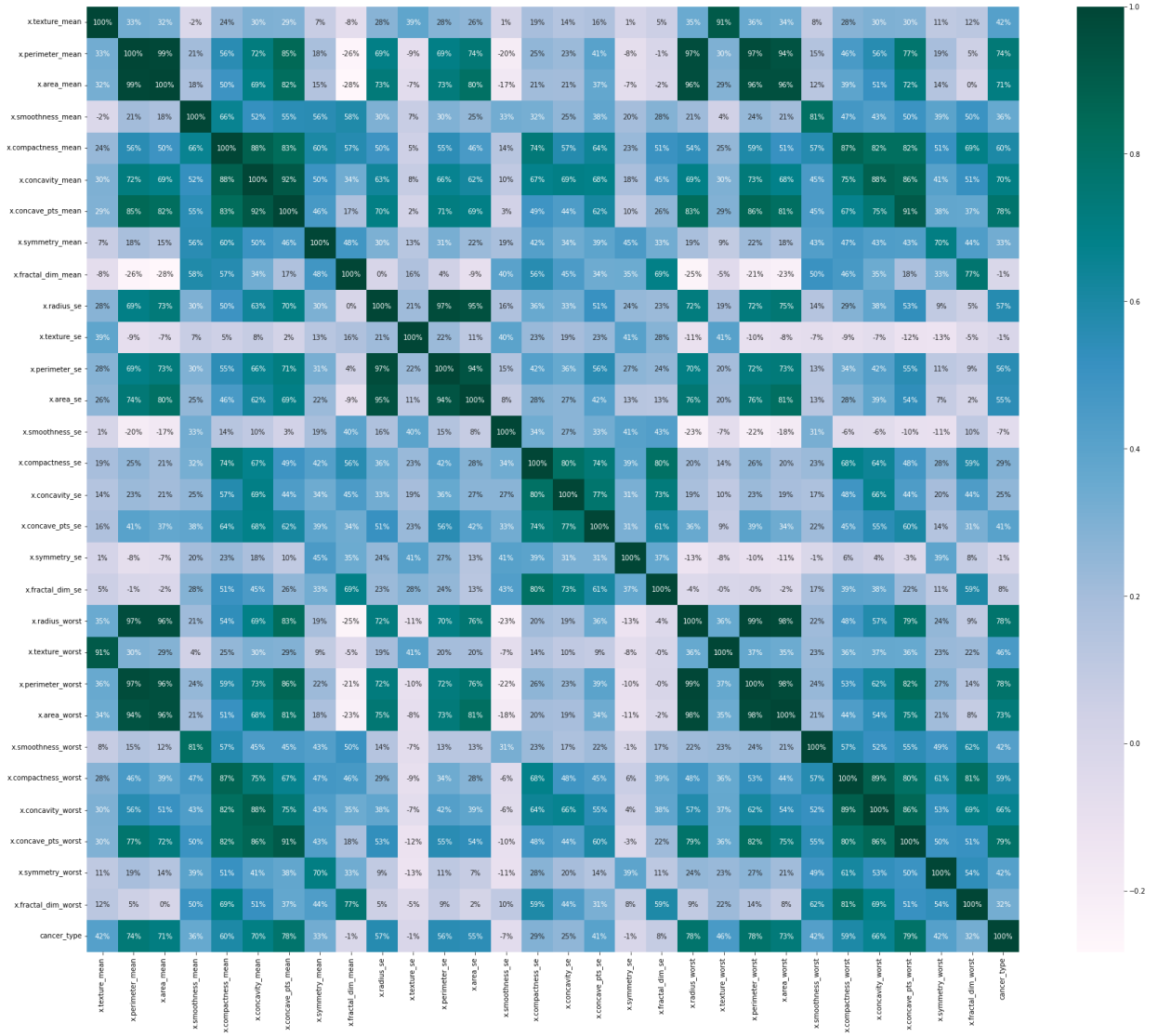
In [15]:
```
df.iloc[:,1:].corr()
```

Out[15]:

	x.texture_mean	x.perimeter_mean	x.area_mean	x.smoothness_mean	x.compactness_mean
x.texture_mean	1.000000	0.329533	0.321086	-0.023389	0.236702
x.perimeter_mean	0.329533	1.000000	0.986507	0.207278	0.556936
x.area_mean	0.321086	0.986507	1.000000	0.177028	0.498502
x.smoothness_mean	-0.023389	0.207278	0.177028	1.000000	0.659123
x.compactness_mean	0.236702	0.556936	0.498502	0.659123	1.000000
x.concavity_mean	0.302418	0.716136	0.685983	0.521984	0.883121
x.concave_pts_mean	0.293464	0.850977	0.823269	0.553695	0.831135
x.symmetry_mean	0.071401	0.183027	0.151293	0.557775	0.602641
x.fractal_dim_mean	-0.076437	-0.261477	-0.283110	0.584792	0.565369
x.radius_se	0.275869	0.691765	0.732562	0.301467	0.497473
x.texture_se	0.386358	-0.086761	-0.066280	0.068406	0.046205
x.perimeter_se	0.281673	0.693135	0.726628	0.296092	0.548905
x.area_se	0.259845	0.744983	0.800086	0.246552	0.455653
x.smoothness_se	0.006614	-0.202694	-0.166777	0.332375	0.135299
x.compactness_se	0.191975	0.250744	0.212583	0.318943	0.738722
x.concavity_se	0.143293	0.228082	0.207660	0.248396	0.570517
x.concave_pts_se	0.163851	0.407217	0.372320	0.380676	0.642262
x.symmetry_se	0.009127	-0.081629	-0.072497	0.200774	0.229977
x.fractal_dim_se	0.054458	-0.005523	-0.019887	0.283607	0.507318
x.radius_worst	0.352573	0.969476	0.962746	0.213120	0.535315
x.texture_worst	0.912045	0.303038	0.287489	0.036072	0.248133
x.perimeter_worst	0.358040	0.970387	0.959120	0.238853	0.590210
x.area_worst	0.343546	0.941550	0.959213	0.206718	0.509604
x.smoothness_worst	0.077503	0.150549	0.123523	0.805324	0.565541
x.compactness_worst	0.277830	0.455774	0.390410	0.472468	0.865809
x.concavity_worst	0.301025	0.563879	0.512606	0.434926	0.816275
x.concave_pts_worst	0.295316	0.771241	0.722017	0.503053	0.815573
x.symmetry_worst	0.105008	0.189115	0.143570	0.394309	0.510223
x.fractal_dim_worst	0.119205	0.051019	0.003738	0.499316	0.687382
cancer_type	0.415185	0.742636	0.708984	0.358560	0.596534

30 rows × 30 columns

In [16]:
```
plt.subplots(figsize=(30,25))
sns.heatmap(df.iloc[:,1:].corr(),annot=True,fmt=".0%", cmap='PuBuGn');
```



Splitting Data to Features and Labels

In [17]:
```
X=df.drop(["cancer_type"],axis=1)
Y=df["cancer_type"]
```

Creating Test and Training dataset

In [18]:
```
traindf, testdf = train_test_split(df, test_size = 0.3)
```

Model Classification

In [19]:
```
def classification_model(model, data, predictors, outcome):
    model.fit(data[predictors], data[outcome])
    predictions = model.predict(data[predictors])
    accuracy = metrics.accuracy_score(predictions, data[outcome])
    print("Accuracy : %s" % "{0:.3%}".format(accuracy))
    kf = KFold(n_splits=5)
    error = []
    for train, test in kf.split(data):
        train_predictors = (data[predictors].iloc[train, :])
        train_target = data[outcome].iloc[train]
        model.fit(train_predictors, train_target)
        error.append(model.score(data[predictors].iloc[test, :], data[outcome].ilo
        print("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))
    model.fit(data[predictors], data[outcome])
```

Logistic Regression Model

In [20]:
```
predictor_var = ['x.radius_mean','x.perimeter_mean','x.area_mean','x.compactness_m
outcome_var='cancer_type'
model=LogisticRegression()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 90.704%
Cross-Validation Score : 91.250%
Cross-Validation Score : 89.375%
Cross-Validation Score : 89.167%
Cross-Validation Score : 89.027%
Cross-Validation Score : 89.703%
```

In [21]:
```
predictor_var = ['x.radius_mean']
model=LogisticRegression()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 86.935%
Cross-Validation Score : 87.500%
Cross-Validation Score : 87.500%
Cross-Validation Score : 86.667%
Cross-Validation Score : 86.203%
Cross-Validation Score : 86.937%
```

Decision Tree Model

In [22]:
```
predictor_var = ['x.radius_mean','x.perimeter_mean','x.area_mean','x.compactness_m
model = DecisionTreeClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 100.000%
Cross-Validation Score : 90.000%
Cross-Validation Score : 86.875%
Cross-Validation Score : 89.583%
Cross-Validation Score : 88.074%
Cross-Validation Score : 88.180%
```

In [23]:
```
predictor_var = ['x.radius_mean']
model = DecisionTreeClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 96.482%
Cross-Validation Score : 81.250%
Cross-Validation Score : 79.375%
Cross-Validation Score : 79.167%
Cross-Validation Score : 79.628%
Cross-Validation Score : 80.665%
```

Random Forest Model

In [24]:
```
predictor_var = ['x.radius_mean','x.perimeter_mean','x.area_mean','x.compactness_m
model = RandomForestClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 100.000%
Cross-Validation Score : 92.500%
Cross-Validation Score : 89.375%
Cross-Validation Score : 90.417%
Cross-Validation Score : 90.281%
Cross-Validation Score : 89.946%
```

In [25]:
```
predictor_var = ['x.radius_mean']
model = RandomForestClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 96.482%
Cross-Validation Score : 81.250%
Cross-Validation Score : 80.000%
Cross-Validation Score : 80.000%
Cross-Validation Score : 80.253%
Cross-Validation Score : 81.165%
```

In [26]:
```
features_mean=list(df.columns[1:11])
```

In [27]:
```
featimp = pd.Series(model.feature_importances_, index=predictor_var).sort_values(a
print(featimp)
```

```
x.radius_mean    1.0
dtype: float64
```

Evaluate the model on Test Dataset

In [28]:
```
predictor_var = features_mean
model = RandomForestClassifier(n_estimators=100,min_samples_split=25, max_depth=7,
classification_model(model, testdf,predictor_var,outcome_var)
```

```
Accuracy : 95.322%
Cross-Validation Score : 80.000%
Cross-Validation Score : 84.118%
Cross-Validation Score : 86.471%
Cross-Validation Score : 89.118%
Cross-Validation Score : 88.941%
```