In [1]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
df = pd.read_csv('C:/Users/Dell/Desktop/Breast Cancer Prediction/data.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.1

5 rows × 33 columns

In [5]:

```
df.columns
```

Out[5]:

```
Index(['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_s
e',
       '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', 'Unnamed: 32'],
      dtype='object')
```

In [6]:

```
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 [7]:

```
df['Unnamed: 32']
```

Out[7]:

```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
      ..
564   NaN
565   NaN
566   NaN
567   NaN
568   NaN
Name: Unnamed: 32, Length: 569, dtype: float64
```

In [8]:

```
df = df.drop("Unnamed: 32", axis=1)
```

In [9]:

```
df.head()
```

Out[9]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.1

5 rows × 32 columns

In [10]:

```
df.columns
```

Out[10]:

```
Index(['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_s
e',
       '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'],
      dtype='object')
```

In [11]:

```
df.drop('id', axis=1, inplace=True)
# df = df.drop('id', axis=1)
```

In [12]:

```
df.columns
```

Out[12]:

```
Index(['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_s
e',
       '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'],
      dtype='object')
```

In [13]:

```
type(df.columns)
```

Out[13]:

```
pandas.core.indexes.base.Index
```

In [14]:

```
l = list(df.columns)
print(l)
```

```
['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mea
n', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'p
erimeter_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', 'compa
ctness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst',
'fractal_dimension_worst']
```

In [15]:

```
features_mean = l[1:11]

features_se = l[11:21]

features_worst = l[21:]
```

In [16]:

```
print(features_mean)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_m
ean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry
_mean', 'fractal_dimension_mean']
```

In [17]:

```
print(features_se)
```

```
['radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'com
pactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_d
imension_se']
```

In [18]:

```
print(features_worst)
```

```
['radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothne
ss_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst',
'symmetry_worst', 'fractal_dimension_worst']
```

In [19]:

```
df.head(2)
```

Out[19]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
0	M	17.99	10.38	122.8	1001.0	0.11840	
1	M	20.57	17.77	132.9	1326.0	0.08474	

2 rows × 31 columns

In [20]:

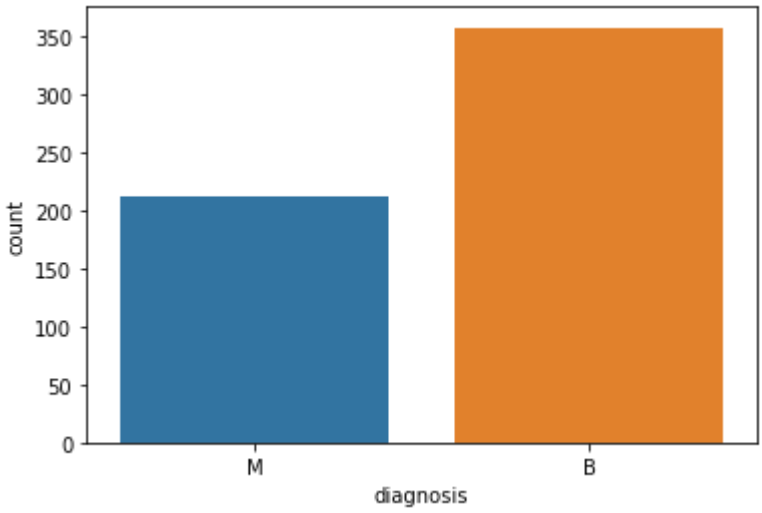
```
df['diagnosis'].unique()
# M= Malignant, B= Benign
```

Out[20]:

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

In [21]:

```
sns.countplot(df['diagnosis'], label="Count",);
```



In [22]:

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

Out[22]:

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

In [23]:

```
df.shape
```

Out[23]:

```
(569, 31)
```

EXPLORE THE DATA

In [24]:

```
df.describe()
# summary of all the numeric columns
```

Out[24]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
count	569.000000	569.000000	569.000000	569.000000	569.000000	5(
mean	14.127292	19.289649	91.969033	654.889104	0.096360	
std	3.524049	4.301036	24.298981	351.914129	0.014064	
min	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 30 columns

In [25]:

```
len(df.columns)
```

Out[25]:

```
31
```

In [26]:

```
# Correlation Plot
corr = df.corr()
corr
```

Out[26]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
radius_mean	1.000000	0.323782	0.997855	0.987357	0.17
texture_mean	0.323782	1.000000	0.329533	0.321086	-0.02
perimeter_mean	0.997855	0.329533	1.000000	0.986507	0.20
area_mean	0.987357	0.321086	0.986507	1.000000	0.17
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1.00
compactness_mean	0.506124	0.236702	0.556936	0.498502	0.65
concavity_mean	0.676764	0.302418	0.716136	0.685983	0.52
concave points_mean	0.822529	0.293464	0.850977	0.823269	0.55
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.55
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	0.58
radius_se	0.679090	0.275869	0.691765	0.732562	0.30
texture_se	-0.097317	0.386358	-0.086761	-0.066280	0.06
perimeter_se	0.674172	0.281673	0.693135	0.726628	0.29
area_se	0.735864	0.259845	0.744983	0.800086	0.24
smoothness_se	-0.222600	0.006614	-0.202694	-0.166777	0.33
compactness_se	0.206000	0.191975	0.250744	0.212583	0.31
concavity_se	0.194204	0.143293	0.228082	0.207660	0.24
concave points_se	0.376169	0.163851	0.407217	0.372320	0.38
symmetry_se	-0.104321	0.009127	-0.081629	-0.072497	0.20
fractal_dimension_se	-0.042641	0.054458	-0.005523	-0.019887	0.28
radius_worst	0.969539	0.352573	0.969476	0.962746	0.21
texture_worst	0.297008	0.912045	0.303038	0.287489	0.03
perimeter_worst	0.965137	0.358040	0.970387	0.959120	0.23
area_worst	0.941082	0.343546	0.941550	0.959213	0.20
smoothness_worst	0.119616	0.077503	0.150549	0.123523	0.80
compactness_worst	0.413463	0.277830	0.455774	0.390410	0.47
concavity_worst	0.526911	0.301025	0.563879	0.512606	0.43
concave points_worst	0.744214	0.295316	0.771241	0.722017	0.50
symmetry_worst	0.163953	0.105008	0.189115	0.143570	0.39
fractal_dimension_worst	0.007066	0.119205	0.051019	0.003738	0.49

30 rows × 30 columns

In [27]:

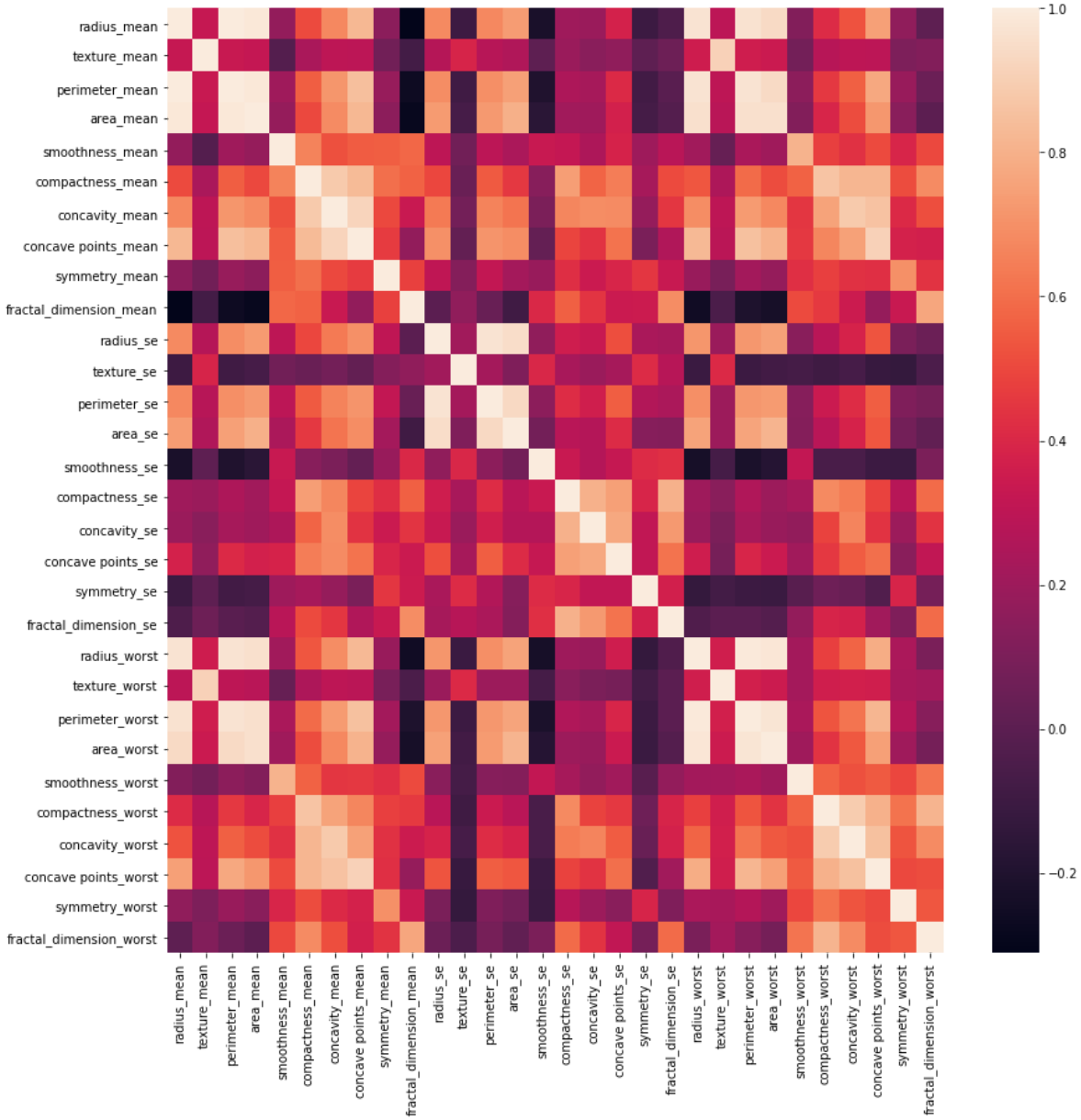
```
corr.shape
```

Out[27]:

```
(30, 30)
```

In [28]:

```
plt.figure(figsize=(14,14))
sns.heatmap(corr);
```



In [29]:

```
df.head()
```

Out[29]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
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	

5 rows × 31 columns

In [30]:

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

In [31]:

```
df.head()
```

Out[31]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

In [32]:

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

Out[32]:

```
array([1, 0], dtype=int64)
```

In [33]:

```
X = df.drop('diagnosis', axis=1)
X.head()
```

Out[33]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
0	17.99	10.38	122.80	1001.0	0.11840	0.27
1	20.57	17.77	132.90	1326.0	0.08474	0.07
2	19.69	21.25	130.00	1203.0	0.10960	0.15
3	11.42	20.38	77.58	386.1	0.14250	0.28
4	20.29	14.34	135.10	1297.0	0.10030	0.13

5 rows × 30 columns

In [34]:

```
y = df['diagnosis']
y.head()
```

Out[34]:

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

In [35]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [36]:

```
df.shape
```

Out[36]:

```
(569, 31)
```

In [37]:

```
X_train.shape
```

Out[37]:

```
(398, 30)
```

In [38]:

```
X_test.shape
```

Out[38]:

```
(171, 30)
```

In [39]:

```
y_train.shape
```

Out[39]:

```
(398,)
```

In [40]:

```
y_test.shape
```

Out[40]:

```
(171,)
```

In [41]:

```
X_train.head(1)
```

Out[41]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
516	18.31	20.58	120.8	1052.0	0.1068	

1 rows × 30 columns

In [42]:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

In [43]:

```
X_train
```

Out[43]:

```
array([[ 1.10532633,  0.24629838,  1.10508062, ...,  0.51420559,
         0.28092604, -0.28946014],
       [-0.34031124, -0.21371493, -0.3183473 , ...,  0.13684347,
         0.34326213, -0.04666068],
       [-0.79104234, -1.0620812 , -0.77913774, ..., -0.80416587,
        -1.73780421, -0.52263318],
       ...,
       [-0.13143585,  0.97908342, -0.06044815, ...,  1.26743236,
         2.45150062,  1.31333717],
       [ 1.86937051,  1.63096158,  1.97803137, ...,  2.60017952,
         1.91764796,  2.3722423 ],
       [-0.53544482,  1.6979987 , -0.47539871, ...,  0.43783468,
         0.50629498,  1.94974985]])
```

Machine Learning Models

Logistic Regression

In [44]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Out[44]:

```
LogisticRegression()
```

In [45]:

```
y_pred = lr.predict(X_test)
```

In [46]:

```
y_pred
```

Out[46]:

```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [47]:

```
y_test
```

Out[47]:

```
425    0
38     1
439    0
301    0
63     0
      ..
89     0
290    0
98     0
453    0
242    0
Name: diagnosis, Length: 171, dtype: int64
```

In [48]:

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

```
0.9707602339181286
```

In [49]:

```
lr_acc = accuracy_score(y_test, y_pred)
print(lr_acc)
```

```
0.9707602339181286
```

In [50]:

```
results = pd.DataFrame()
results
```

Out[50]:

—

In [51]:

```
tempResults = pd.DataFrame({'Algorithm':['Logistic Regression Method'], 'Accuracy':[lr_acc]
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[51]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.97076

Decision Tree Classifier

In [52]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

Out[52]:

```
DecisionTreeClassifier()
```

In [53]:

```
y_pred = dtc.predict(X_test)
y_pred
```

Out[53]:

```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [54]:

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

```
0.9122807017543859
```

In [55]:

```
dtc_acc = accuracy_score(y_test, y_pred)
print(dtc_acc)
```

```
0.9122807017543859
```

In [56]:

```
tempResults = pd.DataFrame({'Algorithm':['Decision tree Classifier Method'], 'Accuracy':[dt
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[56]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.970760
0	Decision tree Classifier Method	0.912281

Random Forest Classifier

In [57]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

Out[57]:

```
RandomForestClassifier()
```

In [58]:

```
y_pred = rfc.predict(X_test)
y_pred
```

Out[58]:

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [59]:

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

```
0.9532163742690059
```

In [60]:

```
rfc_acc = accuracy_score(y_test, y_pred)
print(rfc_acc)
```

```
0.9532163742690059
```

In [61]:

```
tempResults = pd.DataFrame({'Algorithm':['Random Forest Classifier Method'], 'Accuracy':[rf
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[61]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.970760
0	Decision tree Classifier Method	0.912281
0	Random Forest Classifier Method	0.953216

Support Vector Classifier

In [62]:

```
from sklearn import svm
svc = svm.SVC()
svc.fit(X_train,y_train)
```

Out[62]:

```
SVC()
```

In [63]:

```
y_pred = svc.predict(X_test)
y_pred
```

Out[63]:

```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [64]:

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

```
0.9649122807017544
```

In [65]:

```
svc_acc = accuracy_score(y_test, y_pred)
print(svc_acc)
```

```
0.9649122807017544
```

In [66]:

```
tempResults = pd.DataFrame({'Algorithm':['Support Vector Classifier Method'], 'Accuracy':[s
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[66]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.970760
0	Decision tree Classifier Method	0.912281
0	Random Forest Classifier Method	0.953216
0	Support Vector Classifier Method	0.964912

In [ ]: