In [2]:
```
#NANCY MAI - MACHINE LEARNING PROJECT
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

In [146…
```
df = pd.read_csv("/Users/nancymai/Desktop/data.csv")
df.head()
```

Out[146]:

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

5 rows × 33 columns

In [147…
```
df.drop('id',axis=1,inplace=True)
df.drop('Unnamed: 32',axis=1,inplace=True)
df.head()
```

Out[147]:

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

5 rows × 31 columns

In [148…
```
df.isnull().sum()
```

Out[148]:
```
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
dtype: int64
```

In [149…
```
df["diagnosis"].unique()
```

Out[149]:
```
array(['M', 'B'], dtype=object)
```

In [150…
```
df.shape
```

Out[150]:
```
(569, 31)
```

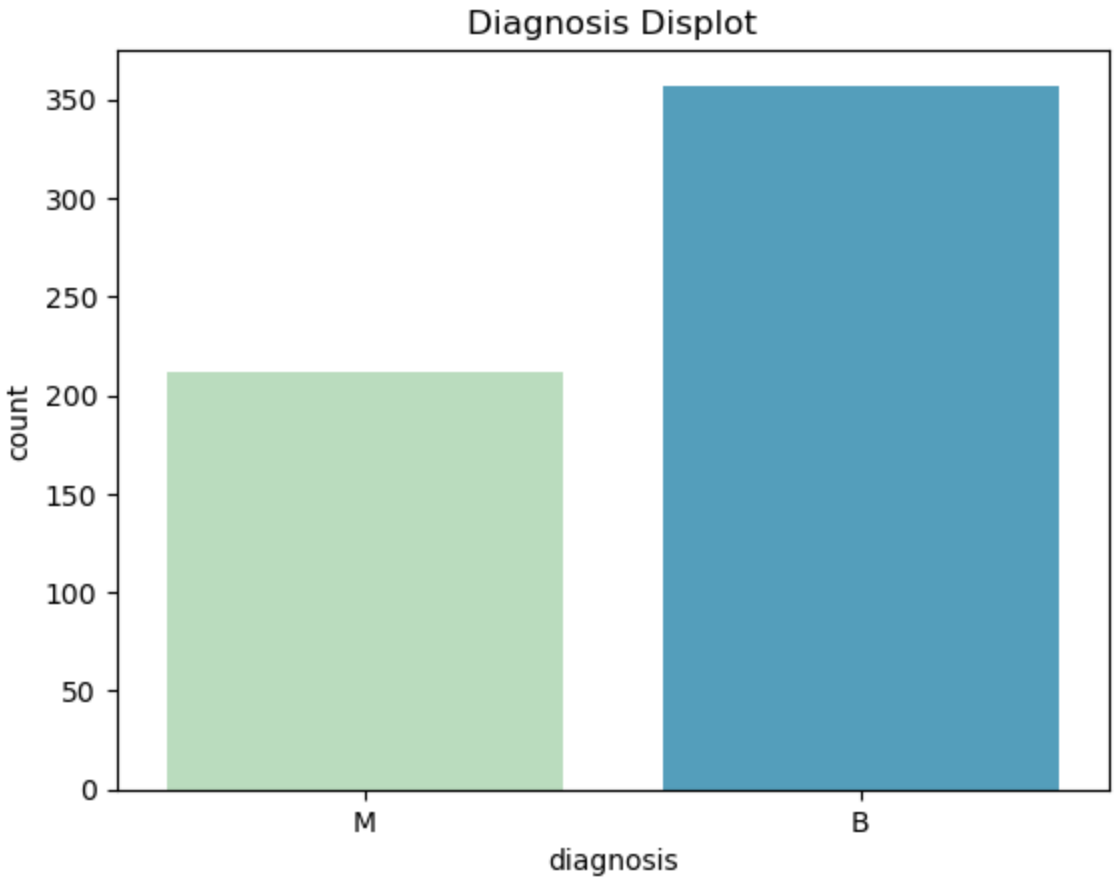
In [151…
```
#EDA
df.describe()
```

Out[151]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400

8 rows × 30 columns

In [152…
```
sns.countplot(x='diagnosis', data=df, palette='GnBu')
plt.title('Diagnosis Displot')
plt.show()
```



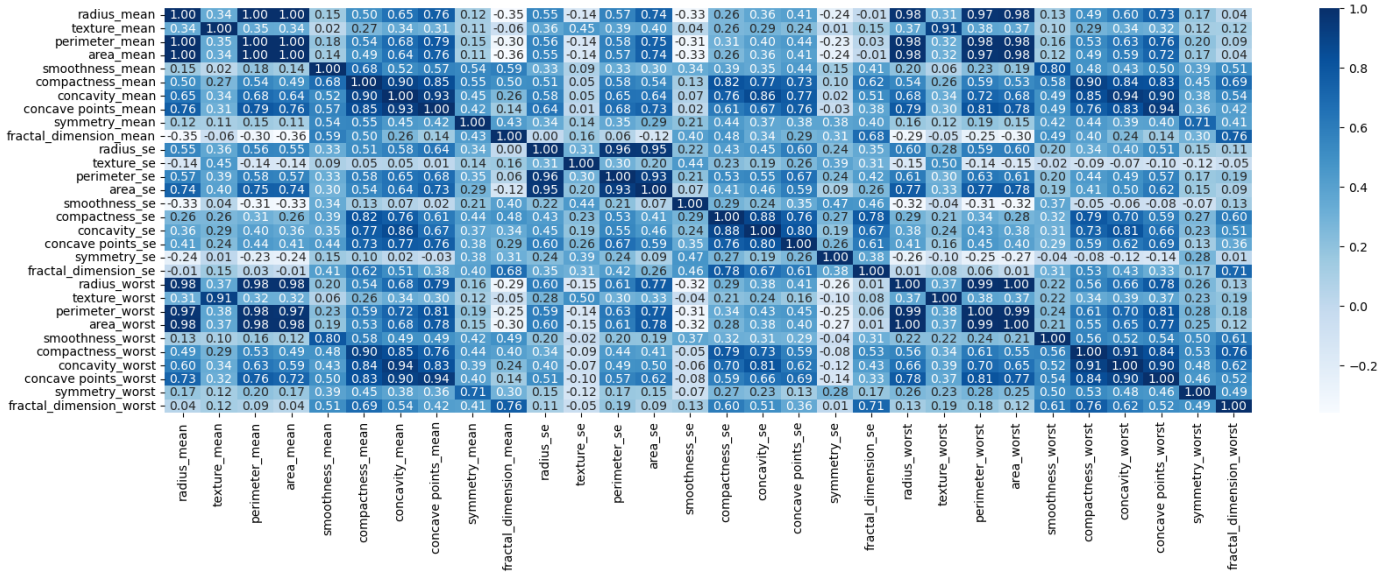
In [153…
```
df.groupby("diagnosis").mean()
```

Out[153]:

diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
B	12.146524	17.914762	78.075406	462.790196	0.092478	0.080(
M	17.462830	21.604906	115.365377	978.376415	0.102898	0.145

2 rows × 30 columns

In [154…
```
corMat = df.corr(method = "spearman")
plt.figure(figsize=(20,6))
sns.heatmap(corMat, annot=True, fmt='.2f', cmap='Blues')
plt.show()
```



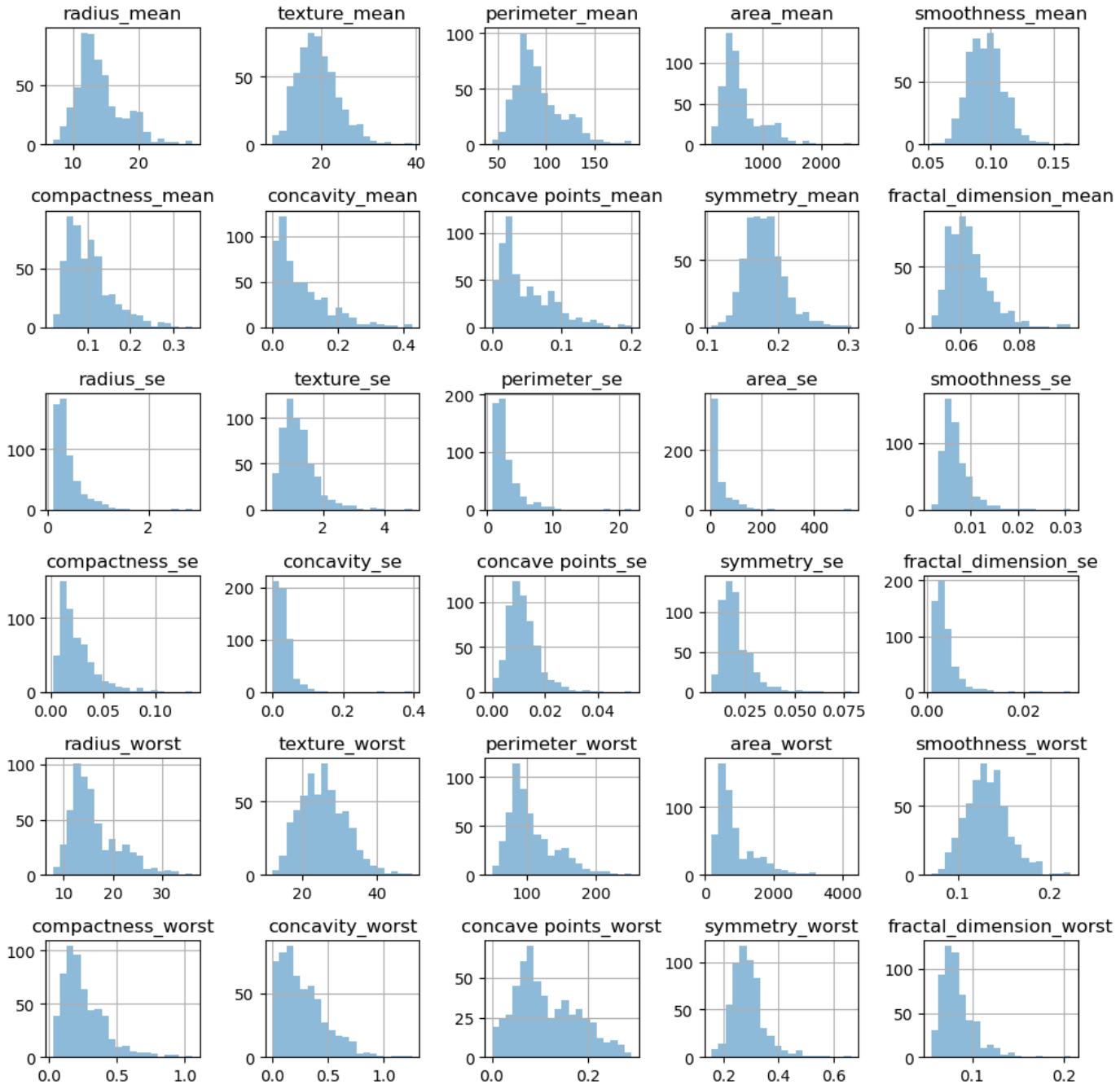
In [155…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from IPython.display import Image
from sklearn.tree import export_graphviz
from six import StringIO
import pydotplus
```

In [156…

```
plt.figure()
df.hist(alpha=0.5, bins=20,figsize=(10,10));
plt.tight_layout()
```

```
<Figure size 640x480 with 0 Axes>
```



In [157…

```
#Train Test Split (for model validation)

from sklearn.model_selection import train_test_split
Y = df['diagnosis']
X = df.drop(columns=['diagnosis'])
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=9)
```

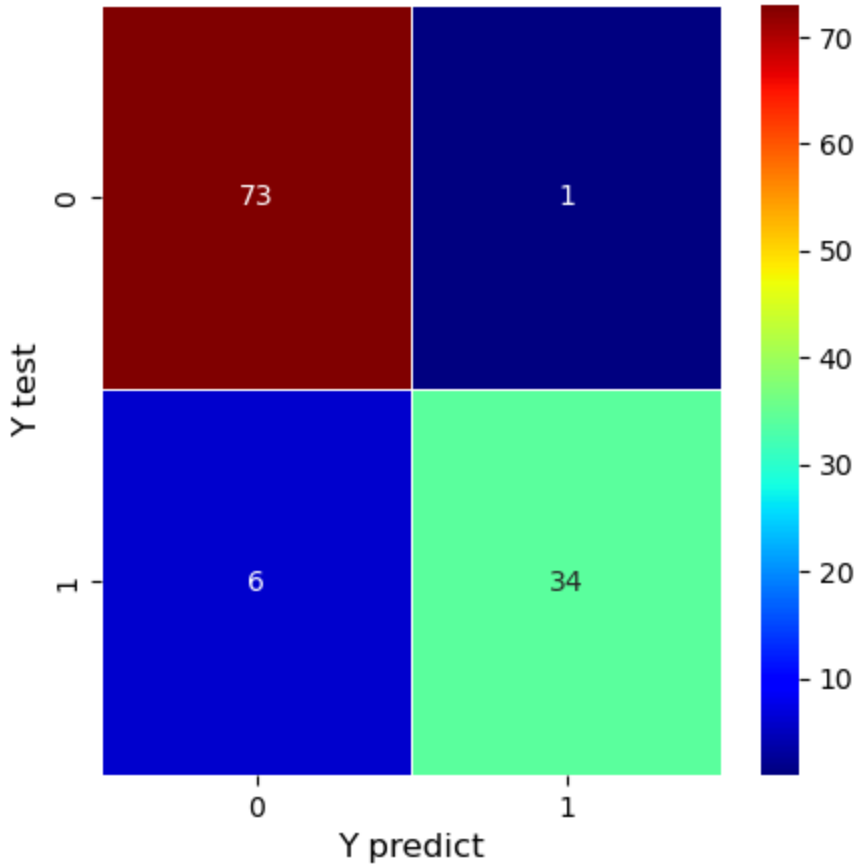
In [158…

```
from sklearn.linear_model import LogisticRegression

#define model
logreg = LogisticRegression(C=10)

#train model
logreg.fit(X_train, Y_train)

#predict target values
Y_predict1 = logreg.predict(X_test)
```

```
/Users/nancymai/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logisti
c.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [159…
```
#confusion matrix to determine performance of logreg
from sklearn.metrics import confusion_matrix

logreg_cm = confusion_matrix(Y_test, Y_predict1)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(logreg_cm, annot=True, linewidth=0.7, fmt='g', ax=ax, cmap="jet")
plt.ylabel('Y test', fontsize=12)
plt.xlabel('Y predict', fontsize=12)
plt.show()
```



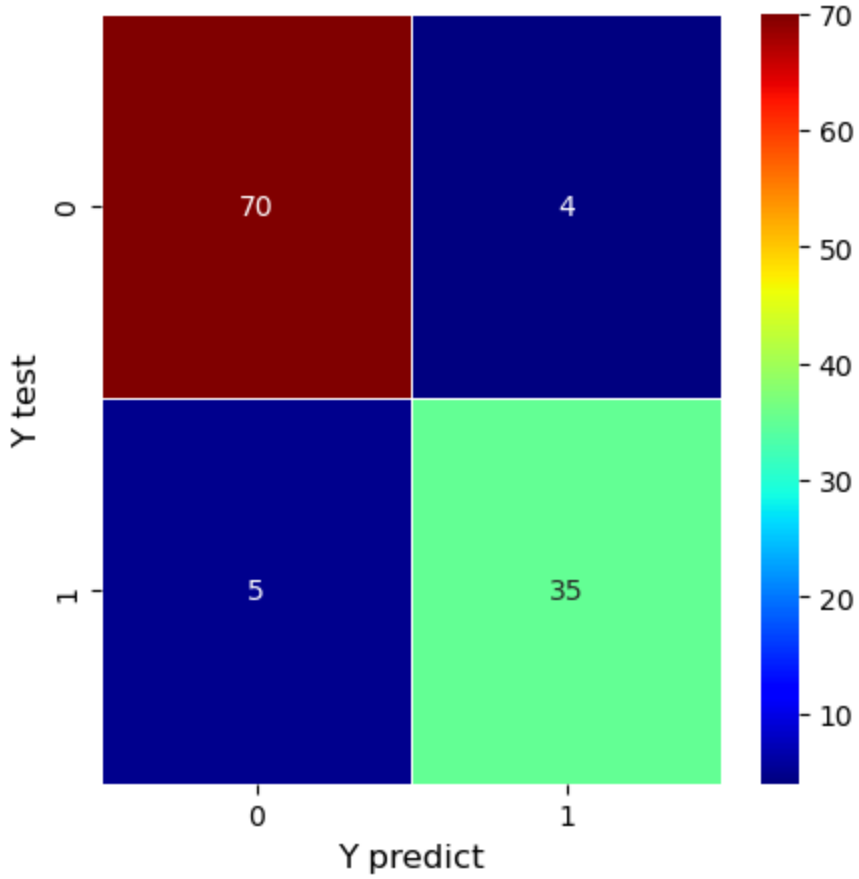
In [160…
```
#test score
logreg_score = logreg.score(X_test, Y_test)
print(logreg_score)
```

```
0.9385964912280702
```

In [161…
```
#Decision Tree (classification algorithm)
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
#define model
dtree = DecisionTreeClassifier(random_state=42)

#train model
dtree.fit(X_train, Y_train)

#predict target values
Y_predict2 = dtree.predict(X_test)
```

In [162…
```
#confusion matrix to determine performance of dtree
dtree_cm = confusion_matrix(Y_test, Y_predict2)
```

```
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(dtree_cm, annot=True, linewidth=0.7, fmt='g', ax=ax, cmap="jet")
plt.ylabel('Y test', fontsize=12)
plt.xlabel('Y predict', fontsize=12)
plt.show()
```



In [163…

```
#test score
dtree_score = dtree.score(X_test, Y_test)
print(dtree_score)
```

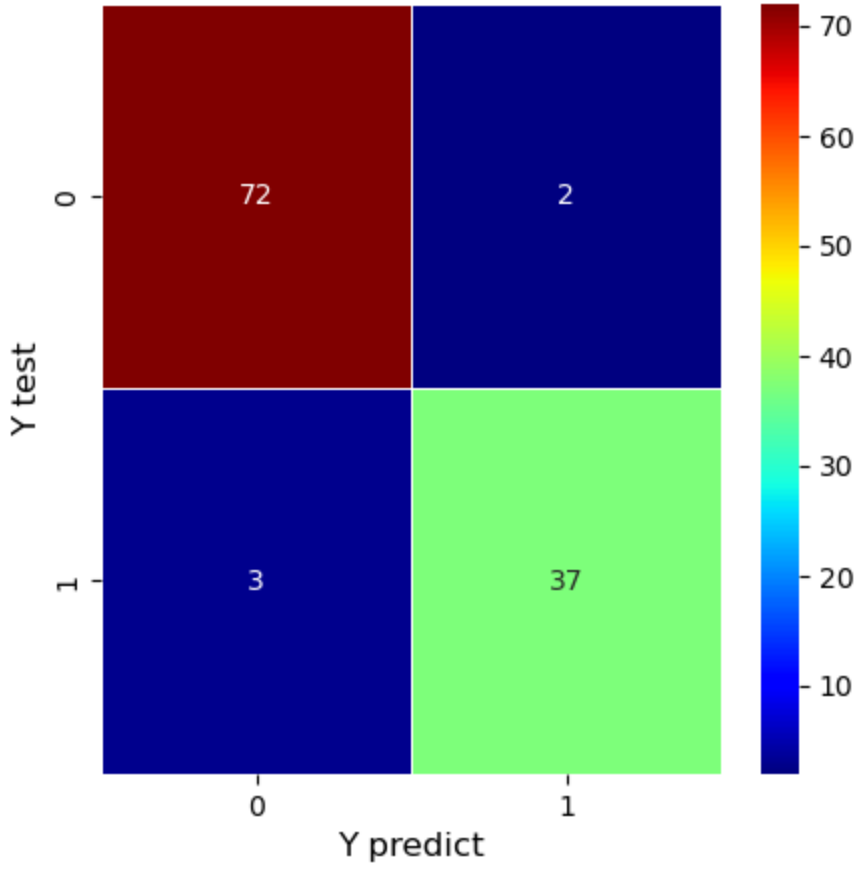
```
0.9210526315789473
```

In [164…

```
# 3. Random Forest Classification
from sklearn.ensemble import RandomForestClassifier

#define model
rf = RandomForestClassifier(n_estimators=100, random_state=9, n_jobs=-1)

#train model
rf.fit(X_train, Y_train)

#predict target values
Y_predict3 = rf.predict(X_test)
```

In [165…

```
#confusion matrix to determine performance of rf
rf_cm = confusion_matrix(Y_test, Y_predict3)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(rf_cm, annot=True, linewidth=0.7, fmt='g', ax=ax, cmap="jet")
plt.ylabel('Y test', fontsize=12)
plt.xlabel('Y predict', fontsize=12)
plt.show()
```



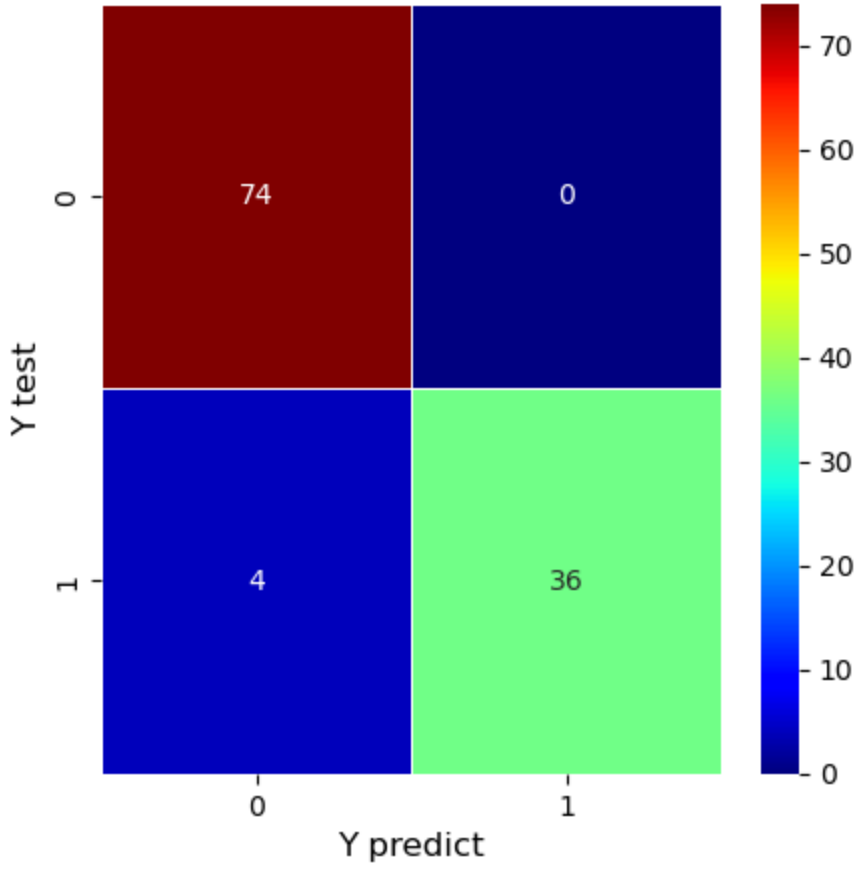
In [166…
```
#test score
rf_score = rf.score(X_test, Y_test)
print(rf_score)
```

```
0.956140350877193
```

In [167…
```
#Bagging (ensemble learning technique)
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve
```

In [168…
```
#define model
bag_clf = BaggingClassifier(DecisionTreeClassifier(random_state=42), n_estimators=500,
        max_samples=100, bootstrap=True, n_jobs=1, random_state=42)

#train model
bag_clf.fit(X_train, Y_train)

#predict target values
Y_predict4 = bag_clf.predict(X_test)
```

In [169…
```
#confusion matrix to determine performance of bag_clf
bag_clf_cm = confusion_matrix(Y_test, Y_predict4)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(bag_clf_cm, annot=True, linewidth=0.7, fmt='g', ax=ax, cmap="jet")
plt.ylabel('Y test', fontsize=12)
plt.xlabel('Y predict', fontsize=12)
plt.show()
```



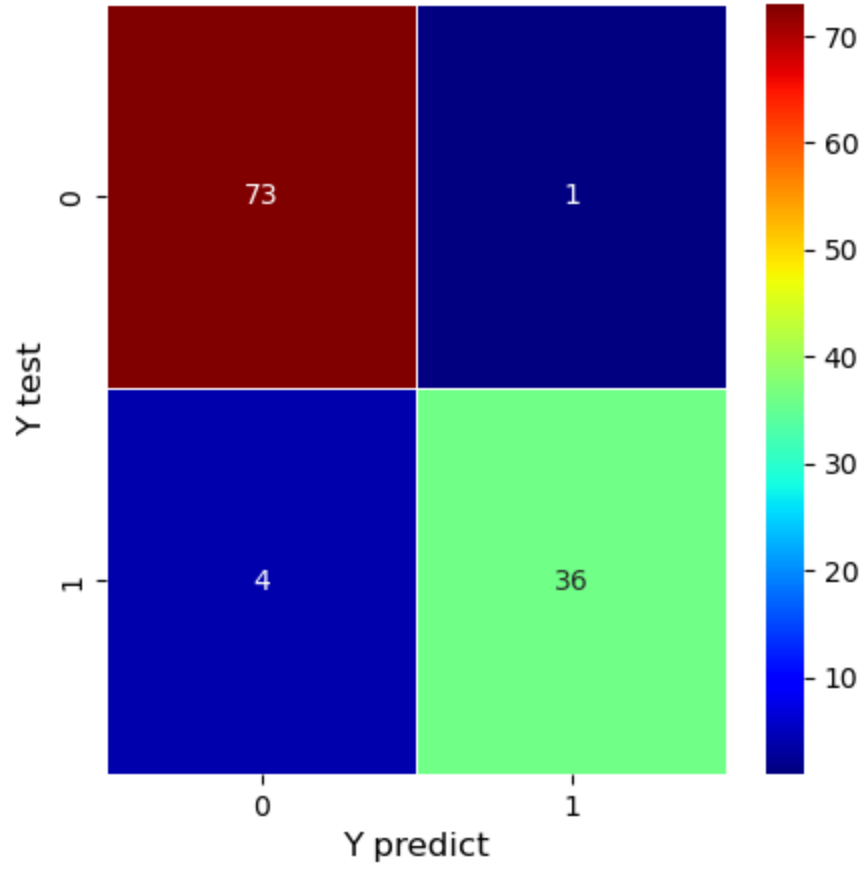
In [170…
```
#test score
print(accuracy_score(Y_test, Y_predict4))
```

```
0.9649122807017544
```

In [171…
```
#Adaptive Boost Classifier

from sklearn.ensemble import AdaBoostClassifier
```

In [172…
```
#define model
ada_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200,
          algorithm="SAMME.R", learning_rate=0.5, random_state=42)

#train model
ada_clf.fit(X_train, Y_train)

#predict target values
Y_predict5 = ada_clf.predict(X_test)
```

In [173…
```
#confusion matrix to determine performance of ada_clf
ada_clf_cm = confusion_matrix(Y_test, Y_predict5)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(ada_clf_cm, annot=True, linewidth=0.7, fmt='g', ax=ax, cmap="jet")
plt.ylabel('Y test', fontsize=12)
plt.xlabel('Y predict', fontsize=12)
plt.show()
```



In [174…
```
#test score
print(accuracy_score(Y_test, Y_predict5))
```

```
0.956140350877193
```

In [175…
```
#Confusion Matrices (altogether to compare)
Testscores = pd.Series([logreg_score, dtree_score, rf_score,
                        accuracy_score(Y_test, Y_predict4), accuracy_score(Y_test, Y_pre
                        index=['Logistic Regression Score', 'Decision Tree Score',
                               'Random Forest Score', 'Bagging Score', 'Adaptive Boost S
print(Testscores)
```

```
Logistic Regression Score    0.938596
Decision Tree Score          0.921053
Random Forest Score          0.956140
Bagging Score                0.964912
Adaptive Boost Score         0.956140
dtype: float64
```

In [176…
```
#The best model to be used for diagnosing breast cancer as found in this analysis is the
```