```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix,
explained_variance_score, mean_absolute_error

# Iris Dataset Preparation
iris_dataset = datasets.load_iris()

print("Iris Dataset Feature and target names - \n")
print(iris_dataset.feature_names)
print(iris_dataset.target_names)

X = iris_dataset.data
Y = iris_dataset.target

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)

#Classification using Naive Bayes Classifier
gaussian_nb_classifier = GaussianNB().fit(X_train, Y_train)
Y_pred = gaussian_nb_classifier.predict(X_test)

print("Iris Dataset GaussianNB Classifier Confusion matrix - \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Iris Dataset GaussianNB Classifier Classification Report - \n")
print(classification_report(Y_test,Y_pred))

bernoulli_nb_classifier = BernoulliNB(alpha=1,
binarize=0.9).fit(X_train, Y_train)
Y_pred = bernoulli_nb_classifier.predict(X_test)

print("Iris Dataset BernoulliNB Classifier Confusion matrix - \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Iris Dataset BernoulliNB Classifier Classification Report - \
n")
print(classification_report(Y_test,Y_pred))

multinomial_nb_classifier = MultinomialNB().fit(X_train, Y_train)
Y_pred = multinomial_nb_classifier.predict(X_test)

print("Iris Dataset MultinomialNB Classifier Confusion matrix - \n")
print(confusion_matrix(Y_test,Y_pred))
```

```
print("\n")
print("Iris Dataset MultinomialNB Classifier Classification Report - \
n")
print(classification_report(Y_test,Y_pred))

# Classification using Decision Tree Classifier
gini_decision_tree_classifier =
DecisionTreeClassifier(criterion="gini",max_depth=15)
gini_decision_tree_classifier.fit(X_train,Y_train)
Y_pred = gini_decision_tree_classifier.predict(X_test)

print("Iris Dataset Decision Tree  Classifier(gini) Confusion matrix -
\n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Iris Dataset Decision Tree Classifier(gini) Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

fig=plt.figure(figsize=(20,20))
plot_tree(gini_decision_tree_classifier)

entropy_decision_tree_classifier =
DecisionTreeClassifier(criterion="entropy",max_depth=15)
entropy_decision_tree_classifier.fit(X_train,Y_train)
Y_pred = entropy_decision_tree_classifier.predict(X_test)

print("Iris Dataset Decision Tree Classifier(entropy) Confusion matrix
- \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Iris Dataset Decision Tree Classifier(entropy) Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

fig=plt.figure(figsize=(20,20))
plot_tree(entropy_decision_tree_classifier)

# Breast Cancer Dataset Preparation
breast_cancer_dataset = datasets.load_breast_cancer()

print("Breast Cancer Dataset Feature and target names - \n")
print(breast_cancer_dataset.feature_names)
print(breast_cancer_dataset.target_names)

X = breast_cancer_dataset.data
Y = breast_cancer_dataset.target

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
```

```
#Classification using Naive Bayes Classifier
gaussian_nb_classifier = GaussianNB().fit(X_train, Y_train)
Y_pred = gaussian_nb_classifier.predict(X_test)

print("Breast Cancer Dataset GaussianNB Classifier Confusion matrix -
\n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Breast Cancer Dataset GaussianNB Classifier Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

bernoulli_nb_classifier = BernoulliNB().fit(X_train, Y_train)
Y_pred = bernoulli_nb_classifier.predict(X_test)

print("Breast Cancer Dataset BernoulliNB Classifier Confusion matrix -
\n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Breast Cancer Dataset BernoulliNB Classifier Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

multinomial_nb_classifier = MultinomialNB().fit(X_train, Y_train)
Y_pred = multinomial_nb_classifier.predict(X_test)

print("Breast Cancer Dataset MultinomialNB Classifier Confusion matrix
- \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Breast Cancer Dataset MultinomialNB Classifier Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

# Classification using Decision Tree Classifier
gini_decision_tree_classifier =
DecisionTreeClassifier(criterion="gini",max_depth=15)
gini_decision_tree_classifier.fit(X_train,Y_train)
Y_pred = gini_decision_tree_classifier.predict(X_test)

print("Breast Cancer Dataset Decision Tree  Classifier(gini) Confusion
matrix - \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Breast Cancer Dataset Decision Tree Classifier(gini)
Classification Report - \n")
print(classification_report(Y_test,Y_pred))
```

```
fig=plt.figure(figsize=(20,20))
plot_tree(gini_decision_tree_classifier)

entropy_decision_tree_classifier =
DecisionTreeClassifier(criterion="entropy",max_depth=15)
entropy_decision_tree_classifier.fit(X_train,Y_train)
Y_pred = entropy_decision_tree_classifier.predict(X_test)

print("Breast Dataset Decision Tree Classifier(entropy) Confusion
matrix - \n")
print(confusion_matrix(Y_test,Y_pred))
print("\n")
print("Breast Dataset Decision Tree Classifier(entropy) Classification
Report - \n")
print(classification_report(Y_test,Y_pred))

fig=plt.figure(figsize=(20,20))
plot_tree(entropy_decision_tree_classifier)

# Diabetes Dataset Preparation
diabetes_dataset = datasets.load_diabetes()

print("Diabetes Dataset Feature names - \n")
print(diabetes_dataset.feature_names)

X = diabetes_dataset.data
Y = diabetes_dataset.target

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)

# Regresssion using Linear Regression
l_regression = LinearRegression()
l_regression.fit(X_train,Y_train)
Y_pred = l_regression.predict(X_test)

print("1) The model explains,",
np.round(explained_variance_score(Y_test,Y_pred)*100,2),"% variance of
the target w.r.t features is")
print("2) The Mean Absolute Error of model is:",
np.round(mean_absolute_error(Y_test,Y_pred ),2))

Diabetes Dataset Feature names -

['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
Diabetes Dataset Linear Regression Classification Report -

1) The model explains, 57.66 % variance of the target w.r.t features
is
2) The Mean Absolute Error of model is: 41.22
```