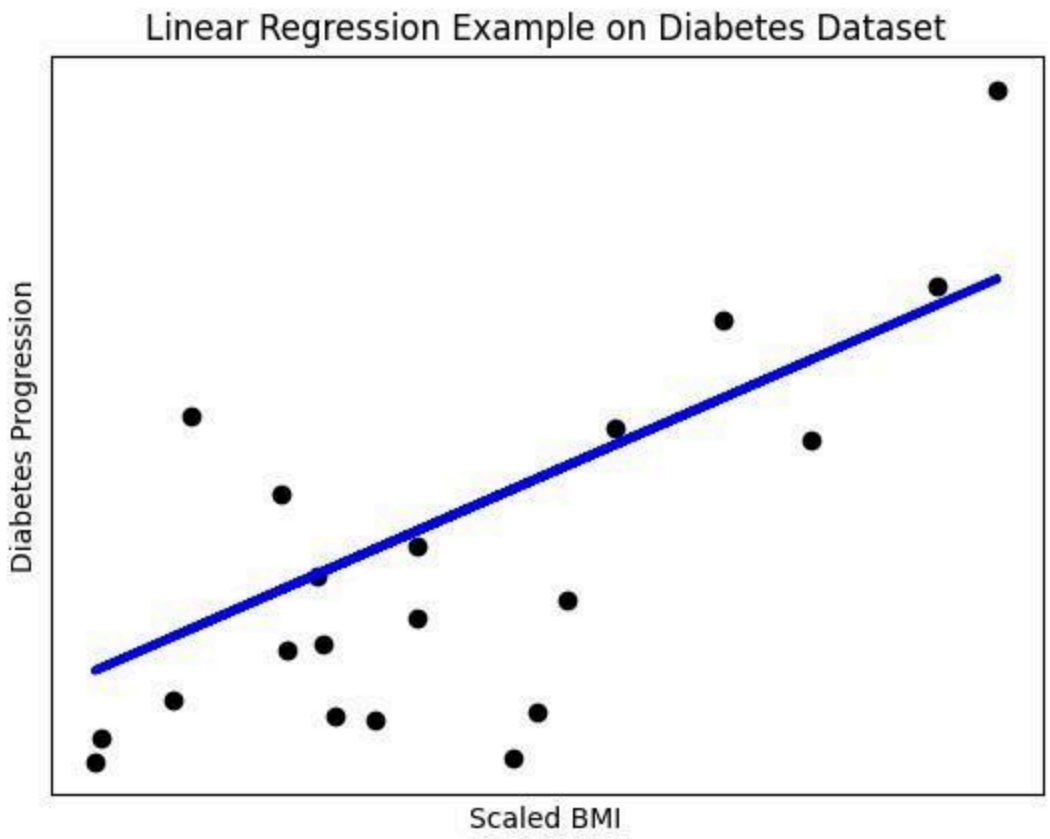
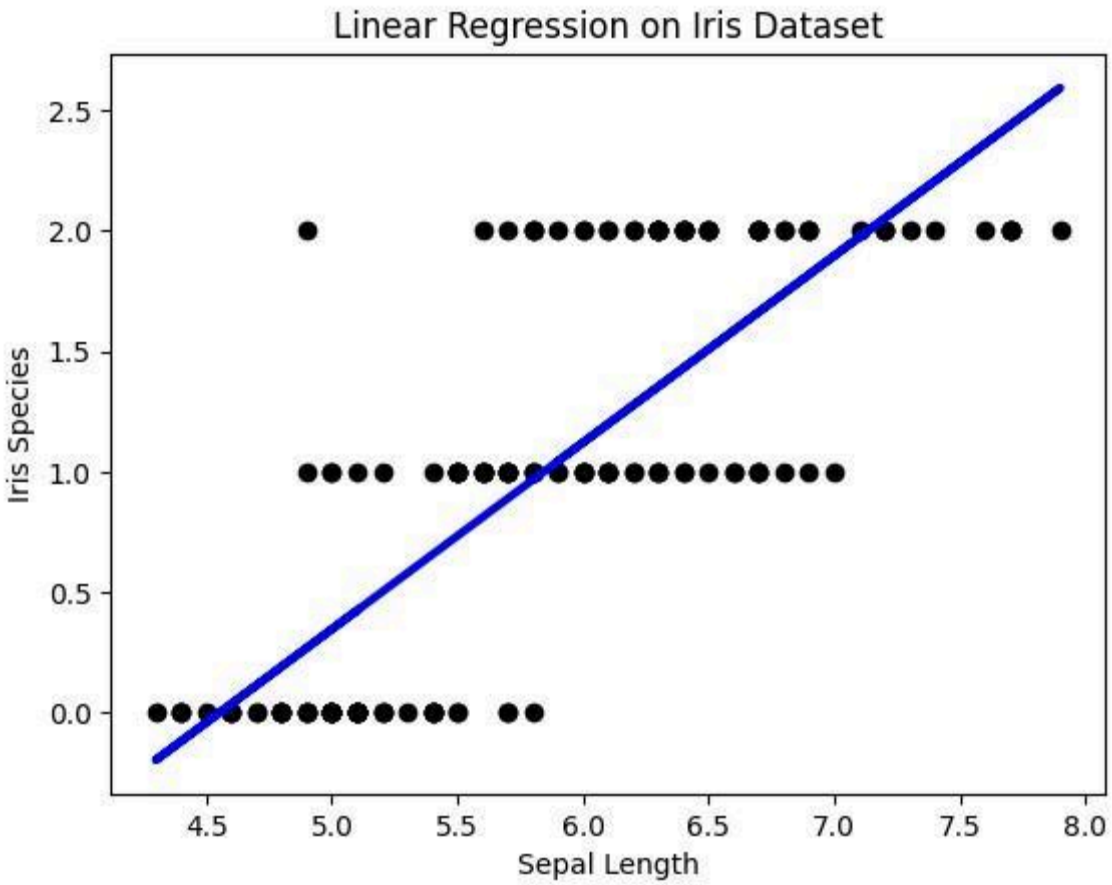
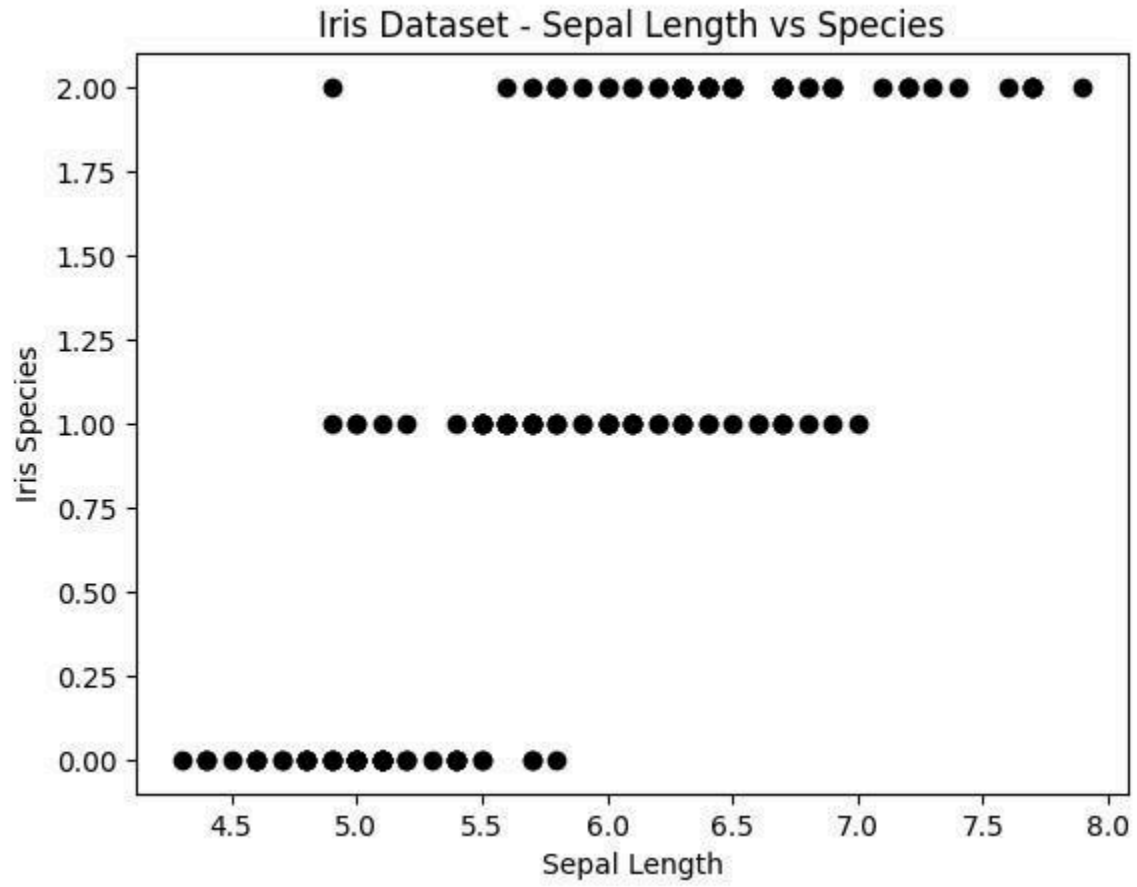
```
#(1)sklearn linear regression example using diabetes with
plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

diabetes = datasets.load_diabetes()

diabetes_X = diabetes.data[:, np.newaxis, 2]

diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

regr = linear_model.LinearRegression()

regr.fit(diabetes_X_train, diabetes_y_train)

diabetes_y_pred = regr.predict(diabetes_X_test)

plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue',
linewidth=3)

plt.xlabel('Scaled BMI')
plt.ylabel('Diabetes Progression')
plt.title('Linear Regression Example on Diabetes Dataset')
plt.xticks(())
plt.yticks(())

plt.show()
```


Linear Regression Example on Diabetes Dataset
Diabetes Progression
Scaled BMI

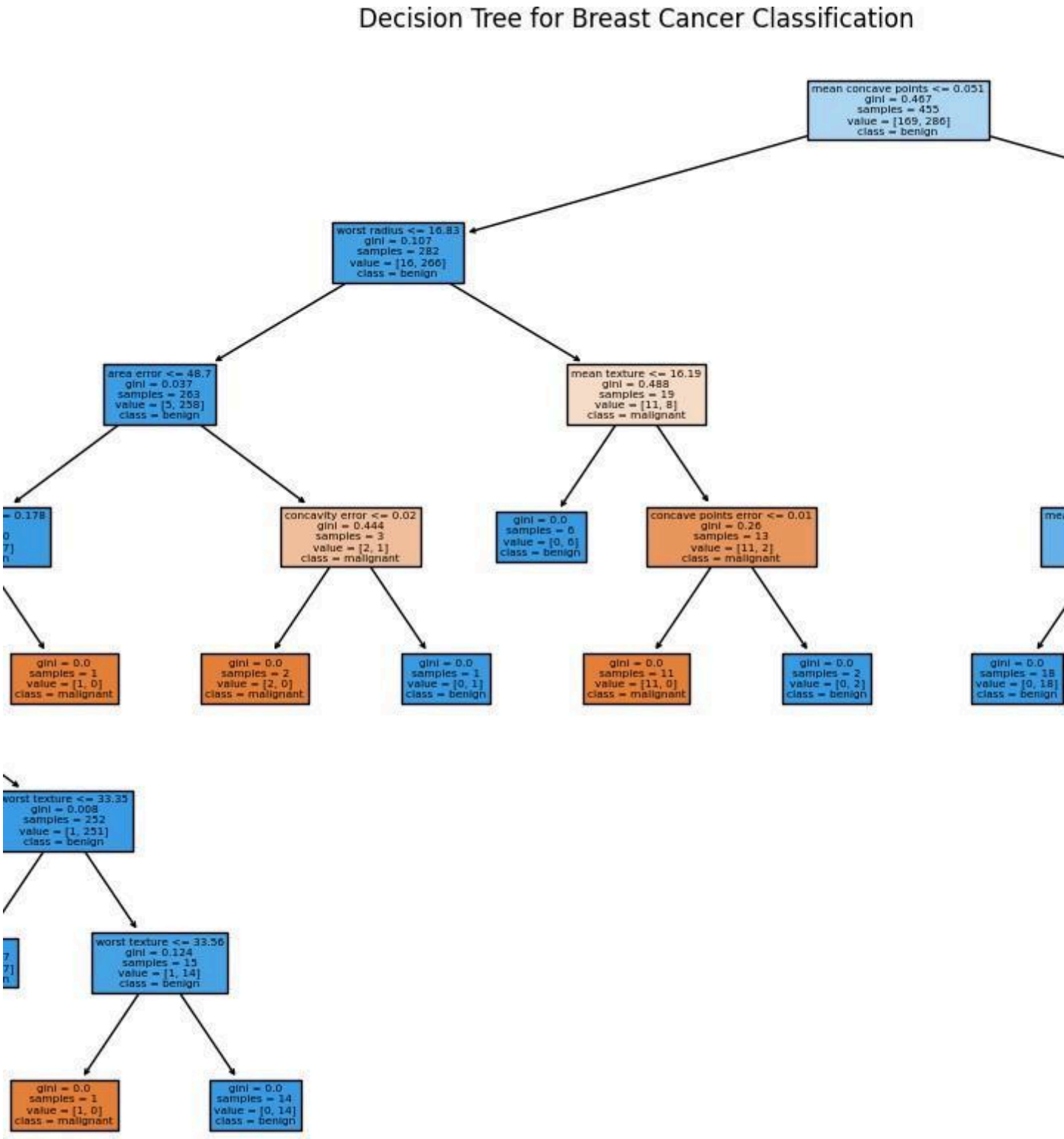
```
#(2)sklearn linear regression  using iris dataset
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model

iris = datasets.load_iris()

X = iris.data[:, np.newaxis, 0]
y = iris.target

regr = linear_model.LinearRegression().fit(X, y)

y_pred = regr.predict(X)

plt.scatter(X, y, color='black')
plt.plot(X, y_pred, color='blue', linewidth=3)

plt.xlabel('Sepal Length')
plt.ylabel('Iris Species')
plt.title('Linear Regression on Iris Dataset')

plt.show()
```


Linear Regression on Iris Dataset
Iris Species
Sepal Length
2.5
2.0
1.5
1.0
0.5
0.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

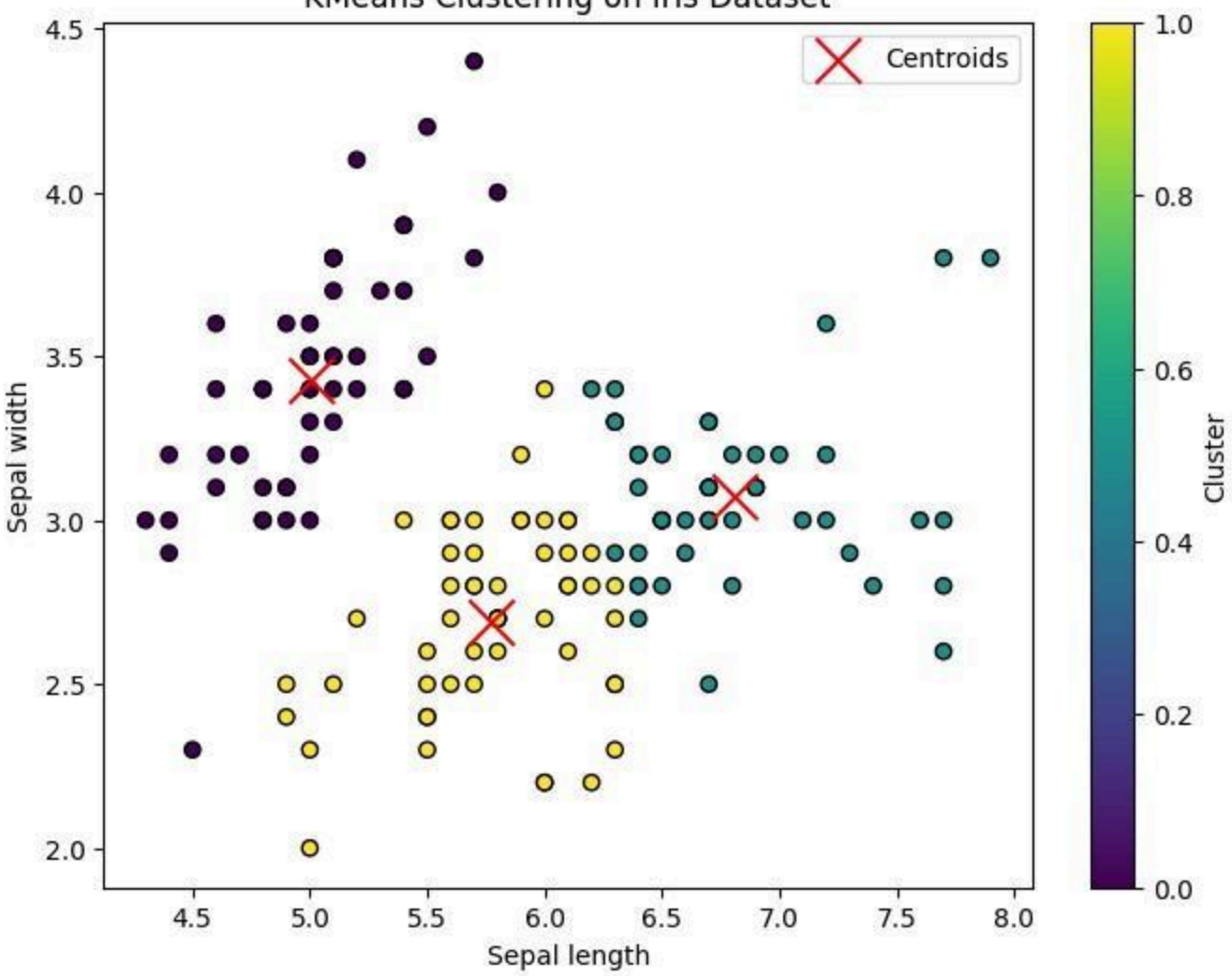
```python
#(3) Python program using Iris dataset before fitting it to a
Linear Regression Model with plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()

X = iris.data[:, np.newaxis, 0]
y = iris.target

plt.scatter(X, y, color='black')
plt.xlabel('Sepal Length')
plt.ylabel('Iris Species')
plt.title('Iris Dataset - Sepal Length vs Species')
plt.show()
```



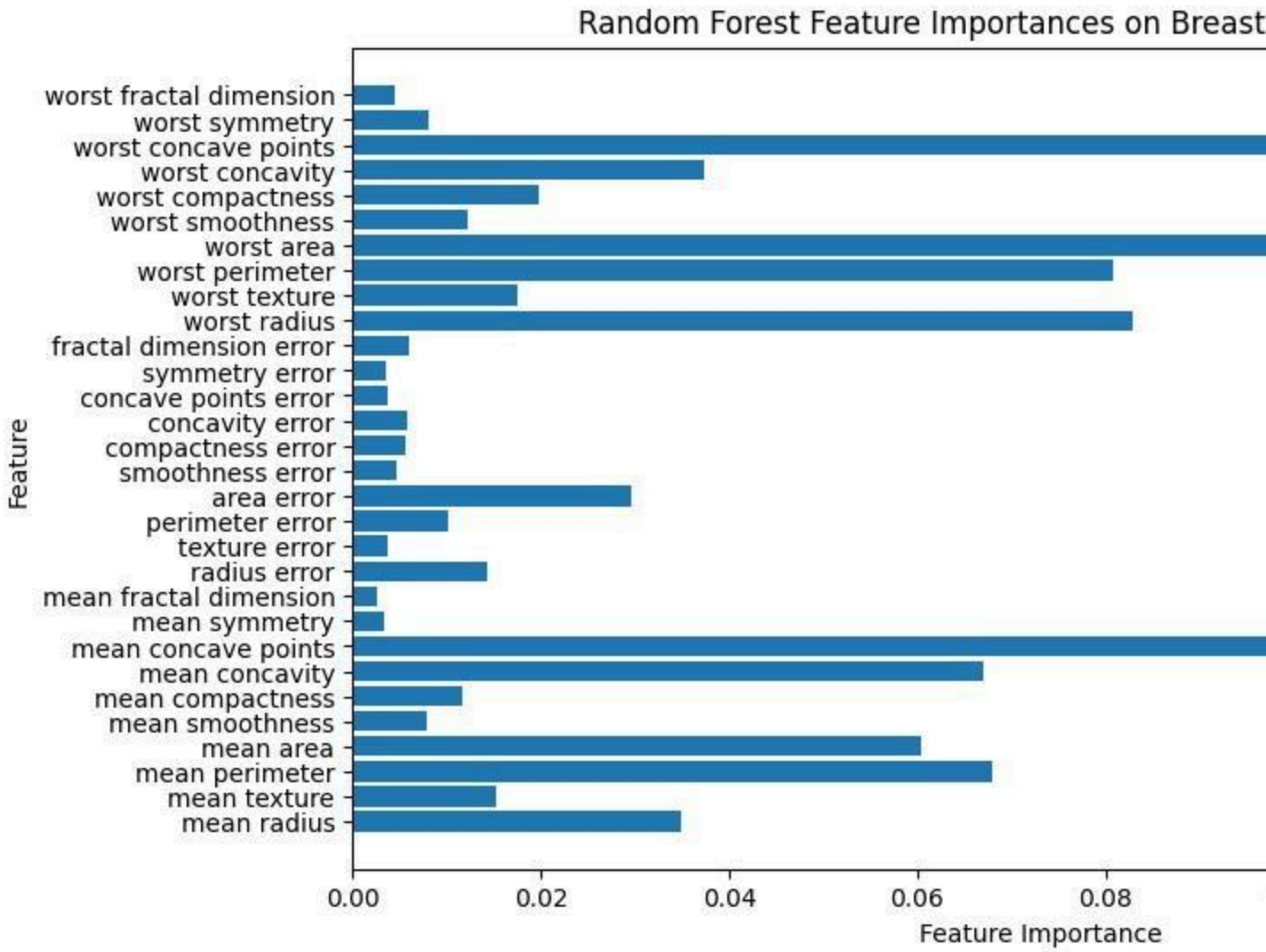
```python
#(6) Python code on sklearn decision tree using breast_cancer
with plot.
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split

breast_cancer = load_breast_cancer()
X = breast_cancer.data
y = breast_cancer.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

plt.figure(figsize=(20, 10))
plot_tree(clf, filled=True,
feature_names=breast_cancer.feature_names,
class_names=breast_cancer.target_names)
plt.title('Decision Tree for Breast Cancer Classification')
plt.show()
```


Decision Tree for Breast Cancer Classification
mean concave points <= 0.051
gini = 0.467
samples = 455
value = [169, 286]
class = benign
worst radius <= 16.83
gini = 0.107
samples = 282
value = [16, 266]
class = benign
area error <= 48.7
gini = 0.037
samples = 263
value = [5, 258]
class = benign
mean texture <= 16.19
gini = 0.488
samples = 19
value = [11, 8]
class = malignant
= 0.178
0
7]
n
concavity error <= 0.02
gini = 0.444
samples = 3
value = [2, 1]
class = malignant
gini = 0.0
samples = 6
value = [0, 6]
class = benign
concave points error <= 0.01
gini = 0.26
samples = 13
value = [11, 2]
class = malignant
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 2
value = [2, 0]
class = malignant
gini = 0.0
samples = 1
value = [0, 1]
class = benign
gini = 0.0
samples = 11
value = [11, 0]
class = malignant
gini = 0.0
samples = 2
value = [0, 2]
class = benign
me
gini = 0.0
samples = 18
value = [0, 18]
class = benign
orst texture <= 33.35
gini = 0.008
samples = 252
value = [1, 251]
class = benign
7]
n
worst texture <= 33.56
gini = 0.124
samples = 15
value = [1, 14]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 14
value = [0, 14]
class = benign

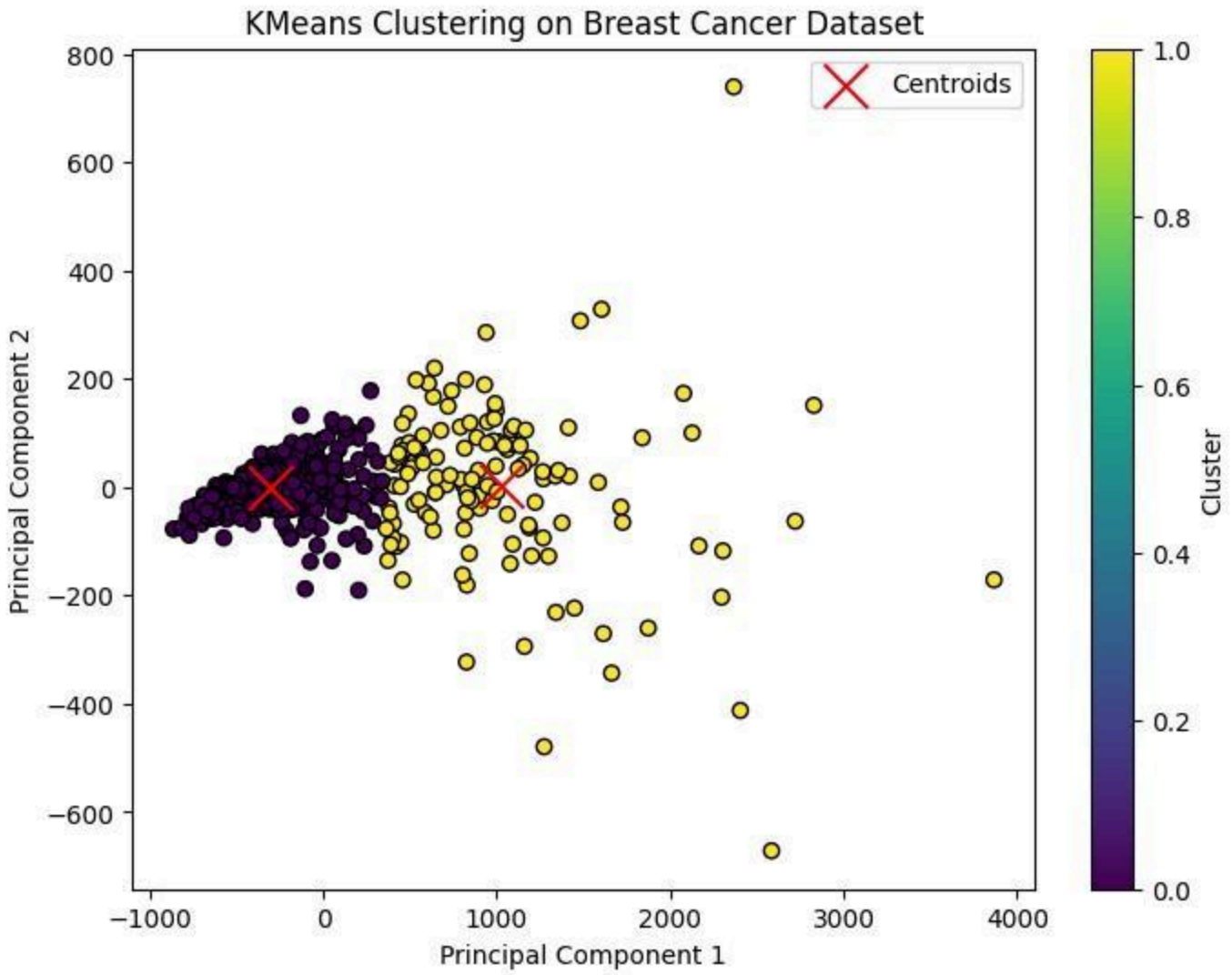
```python
#(7) Python code on sklearn Kmeans classifier using Iris dataset with
plot.
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

iris = load_iris()
X = iris.data[:, :2]

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis',
edgecolor='k')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], marker='x', s=300, c='red', label='Centroids')
plt.title('KMeans Clustering on Iris Dataset')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend()
plt.colorbar(label='Cluster')
plt.show()
```


KMeans Clustering on Iris Dataset
Centroids
Sepal width
Sepal length
Cluster
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
1.0
0.8
0.6
0.4
0.2
0.0

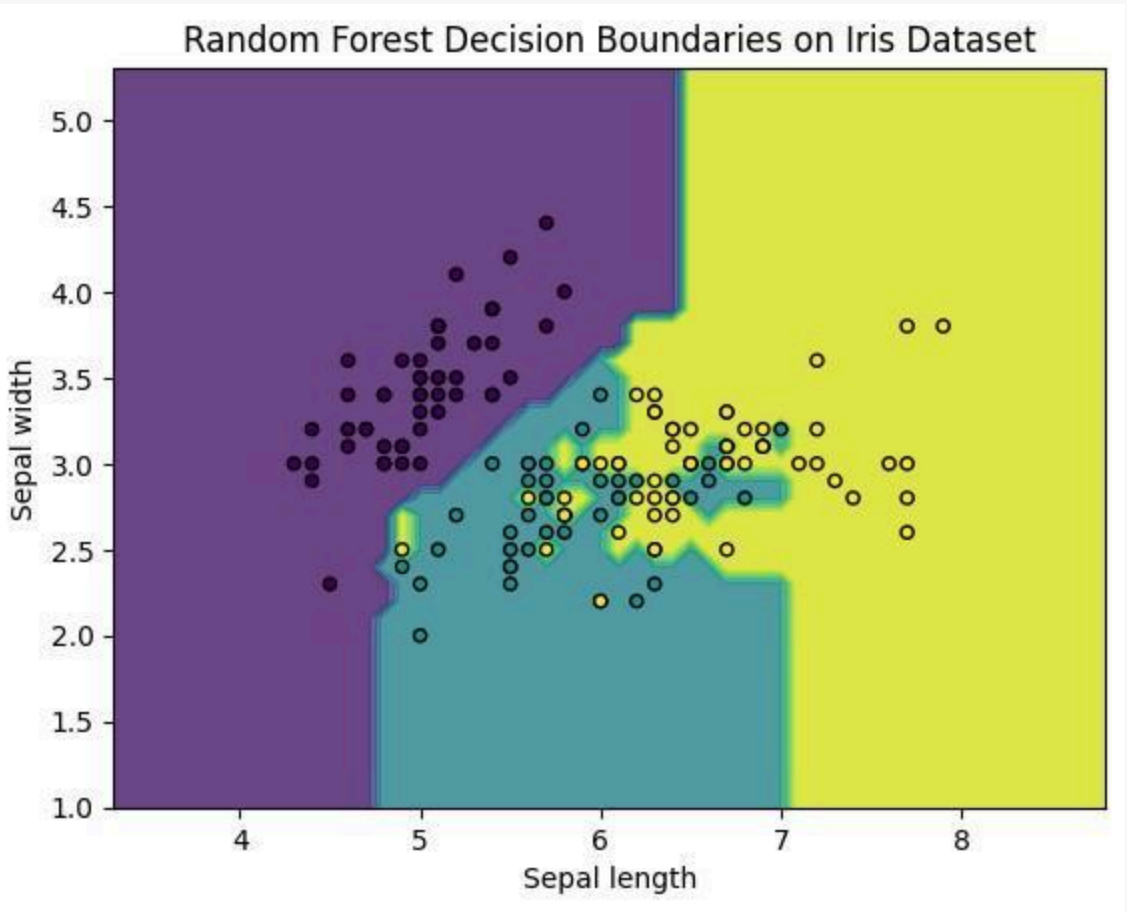
```python
#(8) Python code on sklearn Random Forest using breast_cancer with
plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier

breast_cancer = load_breast_cancer()
X = breast_cancer.data
y = breast_cancer.target

clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X, y)

plt.figure(figsize=(10, 6))
plt.barh(range(X.shape[1]), clf.feature_importances_, align='center')
plt.yticks(range(X.shape[1]), breast_cancer.feature_names)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Random Forest Feature Importances on Breast Cancer
Dataset')
plt.show()
```


Random Forest Feature Importances on Breast
worst fractal dimension
worst symmetry
worst concave points
worst concavity
worst compactness
worst smoothness
worst area
worst perimeter
worst texture
worst radius
fractal dimension error
symmetry error
concave points error
concavity error
compactness error
smoothness error
area error
perimeter error
texture error
radius error
mean fractal dimension
mean symmetry
mean concave points
mean concavity
mean compactness
mean smoothness
mean area
mean perimeter
mean texture
mean radius
Feature
0.00
0.02
0.04
0.06
0.08
Feature Importance

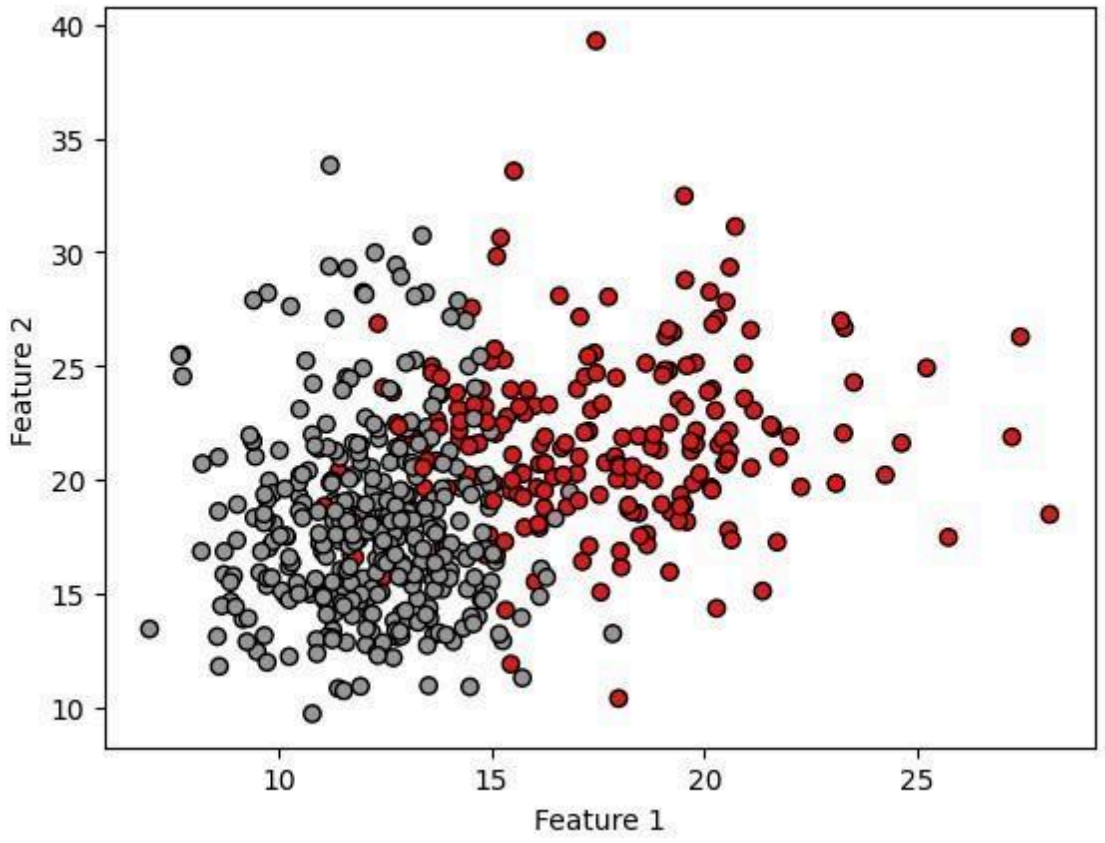
```
#(9) Python code on sklearn Kmeans classifier using breast_cancer
dataset with plot.
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

breast_cancer = load_breast_cancer()
X = breast_cancer.data

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X_pca)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans.labels_,
cmap='viridis', edgecolor='k')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], marker='x', s=300, c='red', label='Centroids')
plt.title('KMeans Clustering on Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.colorbar(label='Cluster')
plt.show()
```


KMeans Clustering on Breast Cancer Dataset
Centroids
Principal Component 2
Principal Component 1
Cluster
800
600
400
200
0
−200
−400
−600
−1000
0
1000
2000
3000
4000
1.0
0.8
0.6
0.4
0.2
0.0

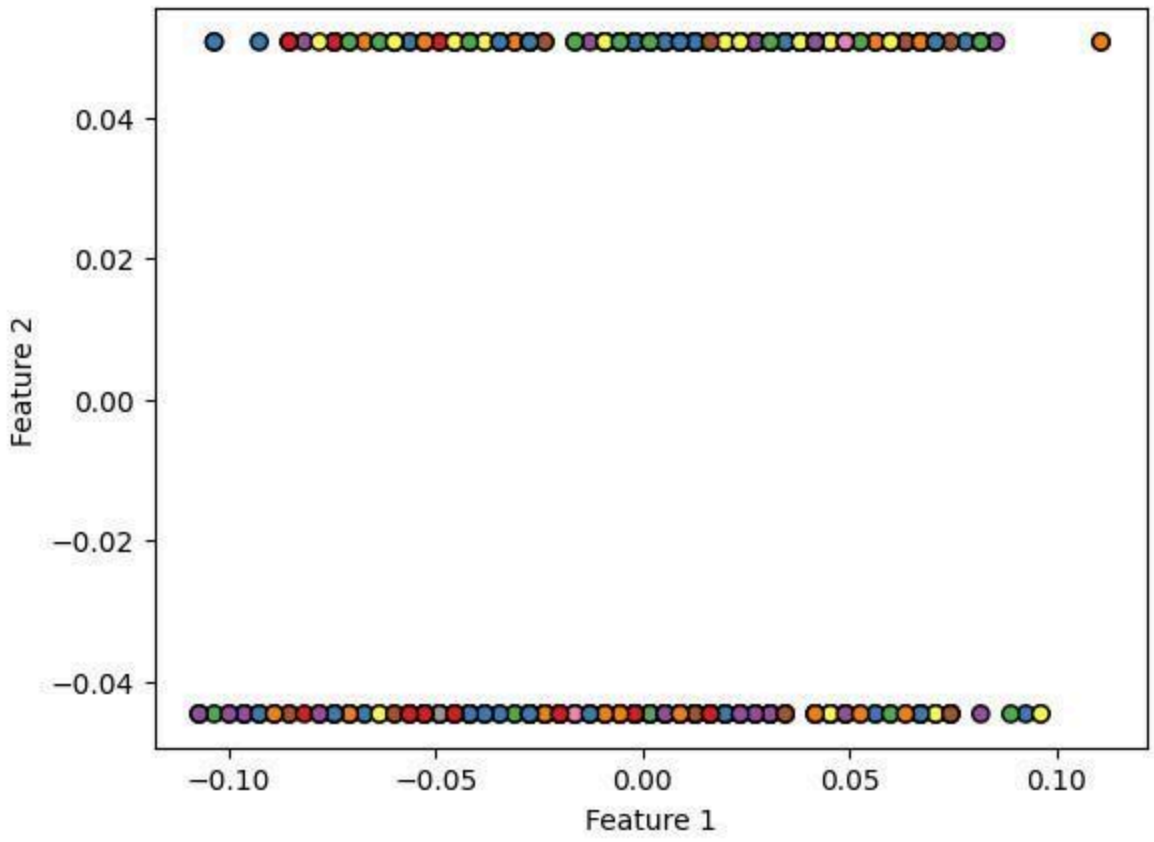
```
#(10) Python code on sklearn Random Forest using Iris dataset with
plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier

iris = load_iris()
X = iris.data[:, :2]

y = iris.target
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min,
y_max, 0.1))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', s=20)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('Random Forest Decision Boundaries on Iris Dataset')
plt.show()
```


Random Forest Decision Boundaries on Iris Dataset
Sepal width
Sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

```
##5. Python code on sklearn logistic regression using breast_cancer
with plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

from sklearn.linear_model import LogisticRegression

breast_cancer = datasets.load_breast_cancer()
X = breast_cancer.data
y = breast_cancer.target

logreg = LogisticRegression()

logreg.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

plt.show()
```



```
##4. Python code on sklearn logistic regression using diabetes with
plot
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.linear_model import LogisticRegression

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

logreg = LogisticRegression()
logreg.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

plt.show()
```


0.04
0.02
0.00
−0.02
−0.04
Feature 2
−0.10
−0.05
0.00
0.05
0.10
Feature 1