Installation

First, install the necessary libraries:

```
bash
Copy code
pip install numpy pandas scikit-learn matplotlib seaborn
```

1. Linear Regression

Dataset: Synthetic dataset

```
python
Copy code
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Create a synthetic dataset
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)

# Plotting
plt.scatter(X_test, y_test, color='black', label='Actual data')
plt.scatter(X_test, predictions, color='red', label='Predictions')
plt.title('Linear Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

2. Logistic Regression

Dataset: Iris dataset

```
python
Copy code
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# Model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('Logistic Regression predictions:', predictions)
```

3. Support Vector Machine (SVM)

Dataset: Iris dataset

```
python
Copy code
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split

# Load dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = svm.SVC()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('SVM predictions:', predictions)
```

4. Decision Tree

Dataset: Iris dataset

```
python
Copy code
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('Decision Tree predictions:', predictions)
```

5. Random Forest

Dataset: Iris dataset

```
python
Copy code
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('Random Forest predictions:', predictions)
```

6. K-Nearest Neighbors (KNN)

Dataset: Iris dataset

```
python
Copy code
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('KNN predictions:', predictions)
```

7. Naive Bayes

Dataset: Iris dataset

```
python
Copy code
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
```

```
# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = GaussianNB()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('Naive Bayes predictions:', predictions)
```

8. Gradient Boosting Machines (GBM)

Dataset: Iris dataset

```
python
Copy code
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = GradientBoostingClassifier()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('GBM predictions:', predictions)
```

9. AdaBoost

Dataset: Iris dataset

```
python
Copy code
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = AdaBoostClassifier()
```

```
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('AdaBoost predictions:', predictions)
```

10. Neural Networks

Dataset: Iris dataset

```
python
Copy code
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Model
model = MLPClassifier(max_iter=1000)
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
print('Neural Network predictions:', predictions)
```