```
# Importing the Libraries
"""
import numpy as np
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifierc
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
"""# Importing the dataset"""
url="https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
names= ['id', 'clump_thickness', 'uniform_cell_size', 'uniform_cell_shape',
      'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
      'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class']
dataset=pd.read_csv(url,names=names)
dataset.head()
"""# cleaning the dataset"""
# Preprocess the data
dataset.replace('?',-99999, inplace=True)
print(dataset.axes)
dataset.drop(['id'], 1, inplace=True)
# Let explore the dataset and do a few visualizations
print(dataset.loc[10])
# Print the shape of the dataset
print(dataset.shape)
# Describe the dataset
print(dataset.describe())
dataset['class'].value_counts()
"""# Plot histograms for each variable"""
dataset.hist(figsize = (10, 10))
plt.show()
# Create scatter plot matrix
scatter_matrix(dataset, figsize = (18,18))
plt.show()
"""# spliting the dataset into train set and test set"""
from sklearn.model_selection import train_test_split
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
"""# training the kneighbours model"""
from sklearn.neighbors import KNeighborsClassifier
neighbour=KNeighborsClassifier()
neighbour.fit(x_train,y_train)
y_pred=neighbour.predict(x_test)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_knn=accuracy_score(y_test,y_pred)
"""# training the svm model"""
from sklearn.svm import SVC
sv=SVC(kernel='linear',random_state=0)
sv.fit(x_train,y_train)
y_pred1=sv.predict(x_test)
from sklearn.metrics import accuracy_score,confusion_matrix
cm1=confusion_matrix(y_test,y_pred1)
print(cm1)
accuracy_svm=accuracy_score(y_test,y_pred1)
"""# training the Naive bayes model"""
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_nb=accuracy_score(y_test, y_pred)
"""# Training the Decision tree model"""
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_dt=accuracy_score(y_test, y_pred)
"""# Training the Random forest Model"""
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_rf=accuracy_score(y_test, y_pred)
"""# Training the Logistic Regression Model"""
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0,max_iter=1000)
classifier.fit(x_train, y_train)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
accuracy_lr=accuracy_score(y_test, y_pred)
"""# comparing the accuracy score of all models"""
print(accuracy_lr)
print(accuracy_nb)
print(accuracy_dt)
print(accuracy_rf)
print(accuracy_knn)
print(accuracy_svm)
```