Intelligent Systems

1. Decision Tree
2. Naïve Bayes
3. Linear Discriminant Analysis
4. Ensemble Learning
   - Bagging: Random Forest
   - Boosting:

     Gradient Boosting
     Ada Boosting

   - Stacking

Decision Tree

Regression algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv("F:/Intelligent systems/petrol_consumption.csv")  #change dataset
dataset.shape
dataset.head(20)    #show first 20 raw
X = dataset.drop('Petrol_Consumption', axis=1)
y = dataset['Petrol_Consumption']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.20, random_state=1)
X_train

from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor ()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
df= pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
```

```
df

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Classification algorithm

```
# Load libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model_selection import train_test_split # Import train_test_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation


# load dataset

pima = pd.read_csv("F:\Intelligent systems\diabetes.csv")

pima.head()


#split dataset in features and target variable

X = pima.drop('Outcome', axis=1)

y = pima['Outcome']
```

```
# Split dataset into training set and test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1) # 70% training


from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier


# Create Decision Tree classifer object

clf = DecisionTreeClassifier ()


# Train Decision Tree Classifer

clf = clf.fit(X_train,y_train)


#Predict the response for test dataset

y_pred = clf.predict(X_test)

y_pred


# Model Accuracy, how often is the classifier correct?

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))


# Create Decision Tree classifer object

clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
```

```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)


#Predict the response for test dataset
y_pred = clf.predict(X_test)


# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
df
*****************************************************************
```

Naive Bayes

```
# load the iris dataset
from sklearn.datasets import load_digits
iris = load_digits ()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.4,
random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB ()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)
y_pred [:20]
```

```
y_test[:20]


# comparing actual response values (y_test) with predicted response values
(y_pred)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
```

LDA

```
#Import Libraries
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

#load breast cancer data
BreastData = load_breast_cancer()

#X Data
X = BreastData.data

#y Data
y = BreastData.target

#Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=44)
```

```
LDAModel = LinearDiscriminantAnalysis ()
LDAModel.fit (X_train, y_train)

#Calculating Details
print('LDAModel Train Score is : ' , LDAModel.score(X_train, y_train))
print('LDAModel classea are : ' , LDAModel.classes_)
print('--------------------------------------------------')
y_pred = LDAModel.predict(X_test)
print('Predicted Value for LDAModel is : ' , y_pred[:20])

#Calculating Confusion Matrix
CM = confusion_matrix(y_test, y_pred)
print('Confusion Matrix is : \n', CM)

# drawing confusion matrix
sns.heatmap(CM, center = True)
plt.show()
```

Bagging:

Random Forest

Regression

```
#Import Libraries
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import median_absolute_error

#load boston data
BostonData = load_boston()

#X Data
X = BostonData.data
print ('X Data is \n', X[:10])
#print ('X shape is ', X.shape)
#print ('X Features are \n', BostonData.feature_names)

#y Data
```

```
y = BostonData.target

#print('y Data is \n' , y[:10])

#print('y shape is ' , y.shape)


#Splitting data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=44, shuffle =True)


#Splitted Data

#print ('X_train shape is ', X_train.shape)

#print ('X_test shape is ', X_test.shape)

#print ('y_train shape is ', y_train.shape)

#print ('y_test shape is ', y_test.shape)

RFRModel = RandomForestRegressor (n_estimators=200, max_depth=3,
random_state=33)

RFRModel.fit (X_train, y_train)


#Calculating Details

print ('Random Forest Regressor Train Score is: ', RFRModel.score(X_train,
y_train))

print ('Random Forest Regressor Test Score is: ', RFRModel.score(X_test, y_test))

print ('Random Forest Regressor No. of features are: ', RFRModel.n_features_)

print ('----------------------------------------------')
```

```
#Calculating Prediction

y_pred = RFRModel.predict(X_test)

print ('Predicted Value for Random Forest Regressor is: ', y_pred [:10])

y_test [:10]

RandomForestRegressorModel.feature_importances_


#Calculating Mean Absolute Error

MAEValue = mean_absolute_error (y_test, y_pred, multioutput=
'uniform_average')

print ('Mean Absolute Error Value is: ', MAEValue)


#Calculating Mean Squared Error

MSEValue = mean_squared_error(y_test, y_pred, multioutput='uniform_average')

print('Mean Squared Error Value is : ', MSEValue)


#Calculating Median Squared Error

MdSEValue = median_absolute_error(y_test, y_pred)

print('Median Squared Error Value is : ', MdSEValue )
```

Classification

```
#Import Libraries
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

#load breast cancer data
BreastData = load_breast_cancer()

#X Data
X = BreastData.data
#print('X Data is \n' , X[:10])
#print('X shape is ' , X.shape)
#print('X Features are \n' , BreastData.feature_names)

#y Data
y = BreastData.target
#print('y Data is \n' , y[:10])
```

```
#print('y shape is ' , y.shape)

#print('y Columns are \n' , BreastData.target_names)


#Splitting data

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.33,
random_state=44)


#print('X_train shape is ' , X_train.shape)

#print('X_test shape is ' , X_test.shape)

#print('y_train shape is ' , y_train.shape)

#print('y_test shape is ' , y_test.shape)


RFC = RandomForestClassifier (criterion = 'gini', n_estimators=100,
max_depth=2, random_state=33) #criterion can be also: entropy

RFC.fit (X_train, y_train)


#Calculating Details

print(' RFC Train Score is : ' , RFC.score(X_train, y_train))

print('RandomForestClassifierModel Test Score is : ' , RFC.score(X_test, y_test))

print('RandomForestClassifierModel features importances are : ' ,
RFC.feature_importances_)

print('--------------------------------------------------')

#Calculating Prediction

y_pred = RFC
```

```
.predict(X_test)


print('Predicted Value for RandomForestClassifierModel is : ' , y_pred[:10])

print('actual Value for RandomForestClassifierModel is : ' , y_test[:10])


#Calculating Confusion Matrix

CM = confusion_matrix(y_test, y_pred)

print('Confusion Matrix is : \n', CM)


# drawing confusion matrix

sns.heatmap(CM, center = True)

plt.show()
```

Boosting:

Gradient Boosting

Classification algorithm

```
# Import models and utility functions

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.datasets import load_digits


dataset=load_digits ()

X = dataset.data

y=dataset.target

X.shape


# Splitting dataset

train, test_X, train_y, test_y = train_test_split (X, y, test_size = 0.25, random_state = 20)


# Instantiate Gradient Boosting Regressor

gbc = GradientBoostingClassifier (n_estimators=300,

                                  learning_rate=0.02,

                                  random_state=100, max_features=5)
```

```
# Fit to training set
gbc.fit (train_X, train_y)

# Predict on test set
pred_y = gbc.predict(test_X)

# accuracy
acc = accuracy_score(test_y, pred_y)
print ("Gradient Boosting Classifier accuracy is: {:.2f}". format (acc))
=============================================================
```

Regression Algorithms

```
# Import the necessary libraries
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_diabetes
dataset=load_digits()
X = dataset.data
y = dataset.target

# Splitting dataset
train_X, test_X, train_y, test_y = train_test_split (X, y, test_size = 0.25)
```

```
 # Instantiate Gradient Boosting Regressor
gbr = GradientBoostingRegressor (learning_rate=0.1,
                               n_estimators=300,
                               max_depth = 1,
                               random_state = 30,
                               max_features = 5)

# Fit to training set
gbr.fit (train_X, train_y)

# Predict on test set
pretty = gbr.predict(test_X)

# test set RMSE
test_rmse = mean_squared_error(test_y, pred_y) ** (1 / 2)

# Print rmse
print ('Root mean Square error: {:.2f}'.format(test_rmse))
================================================================
```

ADA

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier


# Reading the dataset from the csv file
dataset = load_iris ()
X = dataset.data
y = dataset.target
Train, X_val, Y_train, Y_val = train_test_split (X, y, test_size=0.25,
random_state=28)


# Creating adaboost classifier model
adb = AdaBoostClassifier ()
adb_model = adb.fit(X_train,Y_train)
print ("The accuracy of the model on validation set is",
adb_model.score(X_val,Y_val))
y_pred=adb.predict(X_val)
y_pred
y_val
```

Stacking Methods

Regression

```
from sklearn.ensemble import StackingRegressor #Stacking regressor

from sklearn.linear_model import LinearRegression

from sklearn.datasets import  load_boston

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import LinearSVR

BostonData = load_boston()


#X Data

X = BostonData.data


#y Data

y = BostonData.target

estimators = [('Decision tree', DecisionTreeRegressor()),

              ('Random Forest', RandomForestRegressor()),

              ('SVR', LinearSVR(random_state=42))]


from sklearn.model_selection import train_test_split
```

```
stackingreg = StackingRegressor(estimators=estimators,
final_estimator=LinearRegression())


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


stackingreg.fit (X_train, y_train)

y_pred = stackingreg.predict(X_test)


#Model Evaluation

from sklearn.metrics import mean_absolute_error

from sklearn.metrics import mean_squared_error

from sklearn.metrics import median_absolute_error


print('Mean Absolute Error (MAE):', mean_absolute_error(y_test, y_pred))

print('Mean Squared Error (MSE):', mean_squared_error(y_test, y_pred))

print('Mean Squared Log Error:',median_absolute_error(y_test, y_pred))
```

Classification

```
from sklearn.ensemble import StackingClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC
from sklearn.datasets import  load_iris

IrisData = load_iris()


#X Data

X = IrisData.data

#y Data

y = IrisData.target


estimators = [('dtrClassifier', DecisionTreeClassifier()), ('svc', SVC())]

stackingCLS = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())

from sklearn.model_selection import train_test_split


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

stackingCLS.fit(X_train, y_train)


import sklearn.metrics as accuracy_score

from sklearn.metrics import confusion_matrix


y_predict =stackingCLS.predict(X_test)  #Get the splitted part for testing to be predicted, and check the accuray
```

```
#Calculating Confusion Matrix

CM = confusion_matrix(y_test, y_predict)

print('Confusion Matrix is : \n', CM)

print('The Score is : ',stackingCLS.score(X_test , y_test))
```