**Algorithms in Machine Learning**

**1. Linear Regression**

# importing required libraries

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# read the train and test dataset

train\_data = pd.read\_csv('train.csv')

test\_data = pd.read\_csv('test.csv')

print(train\_data.head())

# shape of the dataset

print('\nShape of training data :',train\_data.shape)

print('\nShape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Item\_Outlet\_Sales

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Item\_Outlet\_Sales'],axis=1)

train\_y = train\_data['Item\_Outlet\_Sales']

# seperate the independent and target variable on training data

test\_x = test\_data.drop(columns=['Item\_Outlet\_Sales'],axis=1)

test\_y = test\_data['Item\_Outlet\_Sales']

model = LinearRegression()

# fit the model with the training data

model.fit(train\_x,train\_y)

# coefficeints of the trained model

print('\nCoefficient of model :', model.coef\_)

# intercept of the model

print('\nIntercept of model',model.intercept\_)

# predict the target on the test dataset

predict\_train = model.predict(train\_x)

print('\nItem\_Outlet\_Sales on training data',predict\_train)

# Root Mean Squared Error on training dataset

rmse\_train = mean\_squared\_error(train\_y,predict\_train)\*\*(0.5)

print('\nRMSE on train dataset : ', rmse\_train)

# predict the target on the testing dataset

predict\_test = model.predict(test\_x)

print('\nItem\_Outlet\_Sales on test data',predict\_test)

# Root Mean Squared Error on testing dataset

rmse\_test = mean\_squared\_error(test\_y,predict\_test)\*\*(0.5)

print('\nRMSE on test dataset : ', rmse\_test)

**R Code**

#Load Train and Test datasets

#Identify feature and response variable(s) and values must be numeric and numpy arrays

x\_train <- input\_variables\_values\_training\_datasets

y\_train <- target\_variables\_values\_training\_datasets

x\_test <- input\_variables\_values\_test\_datasets

x <- cbind(x\_train,y\_train)

# Train the model using the training sets and check score

linear <- lm(y\_train ~ ., data = x)

summary(linear)

#Predict Output

predicted= predict(linear,x\_test)

## 2. Logistic Regression

# importing required libraries

import pandas as pd

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

print(train\_data.head())

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = LogisticRegression()

# fit the model with the training data

model.fit(train\_x,train\_y)

# coefficeints of the trained model

print('Coefficient of model :', model.coef\_)

# intercept of the model

print('Intercept of model',model.intercept\_)

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('Target on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('accuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('Target on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('accuracy\_score on test dataset : ', accuracy\_test)

**R Code**

x <- cbind(x\_train,y\_train)

# Train the model using the training sets and check score

logistic <- glm(y\_train ~ ., data = x,family='binomial')

summary(logistic)

#Predict Output

predicted= predict(logistic,x\_test)

## 3. Decision Tree

# importing required libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = DecisionTreeClassifier()

# fit the model with the training data

model.fit(train\_x,train\_y)

# depth of the decision tree

print('Depth of the Decision Tree :', model.get\_depth())

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('Target on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('accuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('Target on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('accuracy\_score on test dataset : ', accuracy\_test)

**R Code**

library(rpart)

x <- cbind(x\_train,y\_train)

# grow tree

fit <- rpart(y\_train ~ ., data = x,method="class")

summary(fit)

#Predict Output

predicted= predict(fit,x\_test)

## 4. SVM (Support Vector Machine)

# importing required libraries

import pandas as pd

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = SVC()

# fit the model with the training data

model.fit(train\_x,train\_y)

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('Target on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('accuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('Target on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('accuracy\_score on test dataset : ', accuracy\_test)

**R Code**

library(e1071)

x <- cbind(x\_train,y\_train)

# Fitting model

fit <-svm(y\_train ~ ., data = x)

summary(fit)

#Predict Output

predicted= predict(fit,x\_test)

**5. Naive Bayes**

# importing required libraries

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = GaussianNB()

# fit the model with the training data

model.fit(train\_x,train\_y)

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('Target on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('accuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('Target on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('accuracy\_score on test dataset : ', accuracy\_test)

**R Code**

library(e1071)

x <- cbind(x\_train,y\_train)

# Fitting model

fit <-naiveBayes(y\_train ~ ., data = x)

summary(fit)

#Predict Output

predicted= predict(fit,x\_test)

## 6. kNN (k- Nearest Neighbors)

# importing required libraries

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = KNeighborsClassifier()

# fit the model with the training data

model.fit(train\_x,train\_y)

# Number of Neighbors used to predict the target

print('\nThe number of neighbors used to predict the target : ',model.n\_neighbors)

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('\nTarget on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('accuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('Target on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('accuracy\_score on test dataset : ', accuracy\_test)

**R Code**

library(knn)

x <- cbind(x\_train,y\_train)

# Fitting model

fit <-knn(y\_train ~ ., data = x,k=5)

summary(fit)

#Predict Output

predicted= predict(fit,x\_test)

## 7. K-Means

# importing required libraries

import pandas as pd

from sklearn.cluster import KMeans

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# shape of the dataset

print('Shape of training data :',train\_data.shape)

print('Shape of testing data :',test\_data.shape)

# Now, we need to divide the training data into differernt clusters

# and predict in which cluster a particular data point belongs.

model = KMeans()

# fit the model with the training data

model.fit(train\_data)

# Number of Clusters

print('\nDefault number of Clusters : ',model.n\_clusters)

# predict the clusters on the train dataset

predict\_train = model.predict(train\_data)

print('\nCLusters on train data',predict\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_data)

print('Clusters on test data',predict\_test)

# Now, we will train a model with n\_cluster = 3

model\_n3 = KMeans(n\_clusters=3)

# fit the model with the training data

model\_n3.fit(train\_data)

# Number of Clusters

print('\nNumber of Clusters : ',model\_n3.n\_clusters)

# predict the clusters on the train dataset

predict\_train\_3 = model\_n3.predict(train\_data)

print('\nCLusters on train data',predict\_train\_3)

# predict the target on the test dataset

predict\_test\_3 = model\_n3.predict(test\_data)

print('Clusters on test data',predict\_test\_3)

**R Code**

library(cluster)

fit <- kmeans(X, 3) # 5 cluster solution

## 8. Random Forest

# importing required libraries

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# read the train and test dataset

train\_data = pd.read\_csv('train-data.csv')

test\_data = pd.read\_csv('test-data.csv')

# view the top 3 rows of the dataset

print(train\_data.head(3))

# shape of the dataset

print('\nShape of training data :',train\_data.shape)

print('\nShape of testing data :',test\_data.shape)

# Now, we need to predict the missing target variable in the test data

# target variable - Survived

# seperate the independent and target variable on training data

train\_x = train\_data.drop(columns=['Survived'],axis=1)

train\_y = train\_data['Survived']

# seperate the independent and target variable on testing data

test\_x = test\_data.drop(columns=['Survived'],axis=1)

test\_y = test\_data['Survived']

model = RandomForestClassifier()

# fit the model with the training data

model.fit(train\_x,train\_y)

# number of trees used

print('Number of Trees used : ', model.n\_estimators)

# predict the target on the train dataset

predict\_train = model.predict(train\_x)

print('\nTarget on train data',predict\_train)

# Accuray Score on train dataset

accuracy\_train = accuracy\_score(train\_y,predict\_train)

print('\naccuracy\_score on train dataset : ', accuracy\_train)

# predict the target on the test dataset

predict\_test = model.predict(test\_x)

print('\nTarget on test data',predict\_test)

# Accuracy Score on test dataset

accuracy\_test = accuracy\_score(test\_y,predict\_test)

print('\naccuracy\_score on test dataset : ', accuracy\_test)

**R Code**

library(randomForest)

x <- cbind(x\_train,y\_train)

# Fitting model

fit <- randomForest(Species ~ ., x,ntree=500)

summary(fit)

#Predict Output

predicted= predict(fit,x\_test)