**Practical No: 9**

**SUPERVISED LEARNING METHODS USING PYTHON**

**AIM: There are 11 variables using which we must predict whether a person will survive the accident or not. Use SUPERVISED LEARNING METHODS of PYTHON.**

**Code:**

**Step 1:** First we need to import pandas and numpy**.** Pandas are basically use for table manipulations. Using Pandas package, we are going to upload Titanic training dataset and then by using head () function we will look at first five rows.

import pandas as pd

import numpy as np

titanic= pd.read\_csv("/content/sample\_data/train.csv")

titanic.head()

**Output:**A screenshot of a computer

Description automatically generated

**Step 2:** Create Two Data Frames, one containing categories and one containing numbers

titanic\_cat = titanic.select\_dtypes(object)

titanic\_num = titanic.select\_dtypes(np.number)

**Step 3:** Now we need to drop two columns (name column and ticket column)

titanic\_cat.head()

**Output:**A screenshot of a computer

Description automatically generated

titanic\_num.head()

**Output:**

A screenshot of a table

Description automatically generated

titanic\_cat.drop(['Name','Ticket'], axis=1, inplace=True)

titanic\_cat.head()

**Step 4: Now to find the null values present in the above column**

titanic\_cat.isnull().sum()

**Output:**

A screenshot of a computer

Description automatically generated

**Step 5: Replace all the null values present with the maximum count category**

titanic\_cat.Cabin.fillna(titanic\_cat.Cabin.value\_counts().idxmax(), inplace=True)

titanic\_cat.Embarked.fillna(titanic\_cat.Embarked.value\_counts().idxmax(), inplace=True)

**Step 6:** After successfully removing all the null values our new data set is ready.

titanic\_cat.head(20)

**Output:**

A table with numbers and letters

Description automatically generated

**Step 7:** The next step will be to replace all the categories with Numerical Labels. For that we will be using LabelEncoders Method.

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

titanic\_cat = titanic\_cat.apply(le.fit\_transform)

**Step 8:** Now we have only one column left which contain null value in it (Age). Let’s replace it with mean

titanic\_cat.head()

**Output:**

A white table with black text and numbers

Description automatically generated

titanic\_num.isna().sum()

**Output:**

A white background with black text

Description automatically generated

titanic\_num.Age.fillna(titanic\_num.Age.mean(), inplace=True)

titanic\_num.isna().sum()

**Output:A close-up of a computer screen

Description automatically generated**

**Step 9:** Now we need to remove the unnecessary columns, since the passengerid is an unnecessary column, we need to drop it

titanic\_num.drop(['PassengerId'], axis=1, inplace=True)

titanic\_num.head()

**Output:**

A table with numbers and letters

Description automatically generated

**Step 10:** Now we will combine two data frames and make it as one

titanic\_final = pd.concat([titanic\_cat,titanic\_num],axis=1)

titanic\_final.head()

**Output:**

A table with numbers and text

Description automatically generated

**Step 11:** Now we will define dependent and independent variables

X=titanic\_final.drop(['Survived'],axis=1)

Y= titanic\_final['Survived']

**Step 12:** Now we will be taking 80% of the data as our training set, and remaining 20% as our test set.

X\_train = np.array(X[0:int(0.80\*len(X))])

Y\_train = np.array(Y[0:int(0.80\*len(Y))])

X\_test = np.array(X[int(0.80\*len(X)):])

Y\_test = np.array(Y[int(0.80\*len(Y)):])

len(X\_train), len(Y\_train), len(X\_test), len(Y\_test)



**Step 13:** Now we will import all the algorithms

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import LinearSVC

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

**Step 14:** Now we will initialize them in respective variables

LR = LogisticRegression()

KNN = KNeighborsClassifier()

NB = GaussianNB()

LSVM = LinearSVC()

NLSVM = SVC(kernel='rbf')

DT = DecisionTreeClassifier()

RF = RandomForestClassifier()

**Step 15:** Now we will train our model

LR\_fit = LR.fit(X\_train, Y\_train)

KNN\_fit = KNN.fit(X\_train, Y\_train)

NB\_fit = NB.fit(X\_train, Y\_train)

LSVM\_fit = LSVM.fit(X\_train, Y\_train)

NLSVM\_fit = NLSVM.fit(X\_train, Y\_train)

DT\_fit = DT.fit(X\_train, Y\_train)

RF\_fit = RF.fit(X\_train, Y\_train)

**Step 16:** Now we need to predict the test data set and compare the accuracy score

LR\_pred = LR\_fit.predict(X\_test)

KNN\_pred = KNN\_fit.predict(X\_test)

NB\_pred = NB\_fit.predict(X\_test)

LSVM\_pred = LSVM\_fit.predict(X\_test)

NLSVM\_pred = NLSVM\_fit.predict(X\_test)

DT\_pred = DT\_fit.predict(X\_test)

RF\_pred = RF\_fit.predict(X\_test)

from sklearn.metrics import accuracy\_score

print("Logistic Regression is %f percent accurate" % (accuracy\_score(LR\_pred, Y\_test)\*100))

print("KNN is %f percent accurate" % (accuracy\_score(KNN\_pred, Y\_test)\*100))

print("Naive Bayes is %f percent accurate" % (accuracy\_score(NB\_pred, Y\_test)\*100))

print("Linear SVMs is %f percent accurate" % (accuracy\_score(LSVM\_pred, Y\_test)\*100))

print("Non Linear SVMs is %f percent accurate" % (accuracy\_score(NLSVM\_pred, Y\_test)\*100))

print("Decision Trees is %f percent accurate" % (accuracy\_score(DT\_pred, Y\_test)\*100))

print("Random Forests is %f percent accurate" % (accuracy\_score(RF\_pred, Y\_test)\*100))

**Final Output:**

A white background with black text

Description automatically generated