# 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:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

**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:**

]:

| | Name | Sex | Ticket | Cabin | Embarked |
|---|---|---|---|---|---|
| 0 | Braund, Mr. Owen Harris | male | A/5 21171 | NaN | S |
| 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | PC 17599 | C85 | C |
| 2 | Heikkinen, Miss. Laina | female | STON/O2. 3101282 | NaN | S |
| 3 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 113803 | C123 | S |
| 4 | Allen, Mr. William Henry | male | 373450 | NaN | S |

titanic_num.head()

**Output:**

Out[4]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | 22.0 | 1 | 0 | 7.2500 |
| 1 | 2 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 |
| 2 | 3 | 1 | 3 | 26.0 | 0 | 0 | 7.9250 |
| 3 | 4 | 1 | 1 | 35.0 | 1 | 0 | 53.1000 |
| 4 | 5 | 0 | 3 | 35.0 | 0 | 0 | 8.0500 |

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:**

```
Out[6]: Sex           0
        Cabin       687
        Embarked      2
        dtype: int64
```

## 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:**

```
In [8]: titanic_cat.head(20)
```

Out[8]:

| | Sex | Cabin | Embarked |
|---|---|---|---|
| 0 | male | B96 B98 | S |
| 1 | female | C85 | C |
| 2 | female | B96 B98 | S |
| 3 | female | C123 | S |
| 4 | male | B96 B98 | S |
| 5 | male | B96 B98 | Q |
| 6 | male | E46 | S |
| 7 | male | B96 B98 | S |
| 8 | female | B96 B98 | S |
| 9 | female | B96 B98 | C |
| 10 | female | G6 | S |
| 11 | female | C103 | S |
| 12 | male | B96 B98 | S |
| 13 | male | B96 B98 | S |
| 14 | female | B96 B98 | S |
| 15 | female | B96 B98 | S |
| 16 | male | B96 B98 | Q |
| 17 | male | B96 B98 | S |
| 18 | female | B96 B98 | S |
| 19 | female | B96 B98 | C |

**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:**

| | Sex | Cabin | Embarked |
|---|---|---|---|
| **0** | 1 | 47 | 2 |
| **1** | 0 | 81 | 0 |
| **2** | 0 | 47 | 2 |
| 3 | 0 | 55 | 2 |
| **4** | 1 | 47 | 2 |

```
titanic_num.isna().sum()
```

**Output:**

```
PassengerId      0
Survived         0
Pclass           0
Age            177
SibSp            0
Parch            0
Fare             0
dtype: int64
```

```
titanic_num.Age.fillna(titanic_num.Age.mean(), inplace=True)
titanic_num.isna().sum()
```

**Output:**

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:6392: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
PassengerId    0
Survived       0
Pclass         0
Age            0
SibSp          0
Parch          0
Fare           0
dtype: int64
```

**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:**

| | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 22.0 | 1 | 0 | 7.2500 |
| 1 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 |
| 2 | 1 | 3 | 26.0 | 0 | 0 | 7.9250 |
| 3 | 1 | 1 | 35.0 | 1 | 0 | 53.1000 |
| 4 | 0 | 3 | 35.0 | 0 | 0 | 8.0500 |

**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:**

| | Sex | Cabin | Embarked | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 47 | 2 | 0 | 3 | 22.0 | 1 | 0 | 7.2500 |
| 1 | 0 | 81 | 0 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 |
| 2 | 0 | 47 | 2 | 1 | 3 | 26.0 | 0 | 0 | 7.9250 |
| 3 | 0 | 55 | 2 | 1 | 1 | 35.0 | 1 | 0 | 53.1000 |
| 4 | 1 | 47 | 2 | 0 | 3 | 35.0 | 0 | 0 | 8.0500 |

**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)
```

```
(712, 712, 179, 179)
```

**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:**

```
Logistic Regression is 83.798883 percent accurate
KNN is 75.977654 percent accurate
Naive Bayes is 82.681564 percent accurate
Linear SVMs is 65.921788 percent accurate
Non Linear SVMs is 74.301676 percent accurate
Decision Trees is 81.005587 percent accurate
Random Forests is 85.474860 percent accurate
```