**GROUP – B**

**Assignment No: 3**

====================================================================

**Title:-** Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months. Perform following steps:

1. Read the dataset.

2. Distinguish the feature and target set and divide the data set into training and test sets.

3. Normalize the train and test data.

4. Initialize and build the model. Identify the points of improvement and implement the same.

5. Print the accuracy score and confusion matrix.

=====================================================================**Objective:-**

-To learn about Neural Network.

- To understand the concept of Normalization.

=====================================================================

**Theory:-**

* **Normalization**

Normalizing a set of data transforms the set of data to be on a similar scale. For machine learning models, our goal is usually to recenter and rescale our data such that is between 0 and 1 or -1 and 1, depending on the data itself. To accomplish this is to calculate the mean and the standard deviation on the set of data and transform each sample by subtracting the mean and dividing by the standard deviation, which is good if we assume that the data follows a normal distribution as this method helps us standardize the data and achieve a standard normal distribution. Normalization can help training of our neural networks as the different features are on a similar scale, which helps to stabilize the gradient descent step, allowing us to use larger learning rates or help models converge faster for a given learning rate. Normalization in machine learning too helps in increasing the convergence rate of machine learning algorithms such as clustering, neural networks, and regression. Typically since the algorithms work better when the data points are near to each other and inside the same range. With normalization, the data points are more homogenous and the machine learning algorithm can learn and make more accurate predictions

Mathematically, we can calculate normalization with the below formula:

**Xn = (X - Xminimum) / ( Xmaximum - Xminimum)**

Xn = Value of Normalization

Xmaximum = Maximum value of a feature

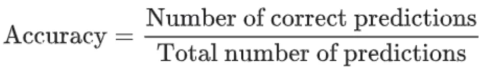
Xminimum = Minimum value of a feature

* **Neural Network**

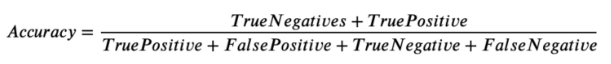
A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy. Neural networks can comprehend unstructured data and make general observations without explicit training.

* **Accuracy Score**

Accuracy score (or just Accuracy) is a Classification metric featuring a fraction of the predictions that a model got right. The metric is prevalent as it is easy to calculate and interpret. Also, it measures the model’s performance with a single value. To evaluate a Classification model using the Accuracy score need - The ground truth classes and the model’s predictions. Accuracy is a highly intuitive metric, so you should not experience any challenges in understanding it. The Accuracy score is calculated by dividing the number of correct predictions by the total prediction number:



The more formal formula is the following one:



Accuracy can be easily described using the Confusion matrix terms such as True Positive, True Negative, False Positive, and False Negative. Still, as described on the Confusion matrix page, these terms are mainly used for the binary Classification tasks.

* **Confusion Matrix**

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.



* **Dataset Description**

The data file bank\_churn.csv contains 12 features about 10000 clients of the bank. The features or variables are the following:

1. customer\_id, unused variable.
2. credit\_score, used as input.
3. country, used as input.
4. gender, used as input.
5. age, used as input.
6. tenure, used as input.
7. balance, used as input.
8. products\_number, used as input.
9. credit\_card, used as input.
10. active\_member, used as input.
11. estimated\_salary, used as input.
12. churn, used as the target. 1 if the client has left the bank during some period or 0 if he/she has not.

* Code Explanation:

*import pandas as pd*

*import seaborn as sns*

*df = pd.read\_csv('Churn\_Modelling.csv')*

*df.shape*

*df.columns*

*df.head()*

Import all libraries required for read values from dataset. The import seaborn portion of the code tells Python to bring the Seaborn library into your current environment.

*#input data*

*x= df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary']]*

*#output data*

*y=df['Exited']*

*x*

Now, let’s perform some exploratory data analysis to gain a better understanding of the independent variables in the dataset and their relationship with customer churn.

*sns.countplot(x=y)*

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot(), it can compare counts across nested variables.

*y.value\_counts()*

Pandas y.value\_counts() function return a Series containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element.

Imbalanced-learn is currently available on the PyPi’s repositories and you can install it via pip: *pip install imbalanced-learn*

*!pip install imbalanced-learn*

*from imblearn.over\_sampling import RandomOverSampler*

*ros = RandomOverSampler(random\_state=0)*

*x\_res, y\_res = ros.fit\_resample(x,y)*

*y\_res.value\_counts()*

Class to perform random over-sampling. Object to over-sample the minority class by picking samples at random with replacement. Fit the statistics and resample the data directly. Parameters - X : ndarray, shape (n\_samples, n\_features) Matrix containing the data which have to be sampled. y : ndarray, shape (n\_samples). Corresponding label for each sample in X.

*##normalization - standardization*

*from sklearn.preprocessing import StandardScaler*

*scaler = StandardScaler()*

*x\_scaled = scaler.fit\_transform(x\_res)*

*x\_scaled*

Standardize features by removing the mean and scaling to unit variance. fit\_transform() is used on the training data so that we can scale the training data and also learn the scaling parameters of that data.

*#cross-validation*

*from sklearn.model\_selection import train\_test\_split*

*x\_train,x\_test,y\_train,y\_test =train\_test\_split(x\_scaled,y\_res,random\_state=0,test\_size=0.25)*

*x\_test.shape*

*x\_res.shape*

*x\_train.shape*

For neural networks you have input features (X) and output labels (Y). It's very important to split your data into a training dataset and testing dataset. To make this easy sklearn has a function called train\_test\_split().

*from sklearn.neural\_network import MLPClassifier*

*ann=MLPClassifier(hidden\_layer\_sizes=(100,100,100),random\_state=0, max\_iter=100, activation ='relu')*

*import warnings*

*#We do not want to see warnings*

*warnings.filterwarnings("ignore")*

*ann.fit(x\_train,y\_train)*

*y\_pred = ann.predict(x\_test)*

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset. MLPClassifier supports only the Cross-Entropy loss function, which allows probability estimates by running the predict\_proba method. MLP trains using Backpropagation. MLPClassifier supports multi-class classification by applying Softmax as the output function. The model supports multi-label classification in which a sample can belong to more than one class. For each class, the raw output passes through the logistic function. Values larger or equal to 0.5 are rounded to 1, otherwise to 0. For a predicted output of a sample, the index where the value is 1 represents the assigned classes of that sample. In MLPClassifier() function following parameters are included:

1. hidden\_layer\_sizes : With this parameter we can specify the number of layers and the number of nodes we want to have in the Neural Network Classifier. Each element in the tuple represents the number of nodes at the ith position, where i is the index of the tuple. Thus, the length of the tuple indicates the total number of hidden layers in the neural network.
2. max\_iter: Indicates the number of epochs.
3. activation: The activation function for the hidden layers.
4. solver: This parameter specifies the algorithm for weight optimization over the nodes.

*From sklearn.metrics import ConfusionMatrixDisplay, classification\_report*

*from sklearn.metrics import accuracy\_score*

*y\_test.value\_counts()*

*ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred)*

*accuracy\_score(y\_test, y\_pred)*

*print(classification\_report(y\_test,y\_pred))*

Confusion Matrix visualization. It is recommend to use plot\_confusion\_matrix to create a ConfusionMatrixDisplay. All parameters are stored as attributes. Accuracy classification score. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y\_true.

=====================================================================

**Conclusion:-**

Thus we have studied how to classify data using Neural Network.

=====================================================================