Experiment # 12: Implement an ANN to predict the Customer Churn.

1. Differences between Biological Neural Network (BNN) and Artificial Neural Network (ANN):

BNN	ANN
Made of real neurons	Made of mathematical models
Learns naturally	Learns through training algorithms
Works in a parallel manner	Simulates parallelism via hardware
Energy efficient	Needs high computational power
Grows and adapts over time	Requires manual updates/training

---

2. Characteristics of Artificial Neural Network (ANN) and Its Applications:

Characteristics:

- **Learns from data (adaptive)**
- **Handles nonlinear relationships**
- **Works in parallel**
- **Generalizes well to new data**

Applications:

- **Image and speech recognition**
- **Healthcare diagnosis**
- **Financial forecasting**
- **Autonomous systems**

---

3. What is the Boston Housing Dataset?

- **A dataset used for predicting house prices in Boston suburbs.**
- **Target variable: MEDV (Median house value in $1000s)**

Features include:

- **CRIM: Crime rate**
- **RM: Avg. number of rooms**
- **LSTAT: % of lower-income population**
- **DIS, NOX, AGE, etc. (total 13 features)**

---

4. What are activation functions and why are they used?

- **Activation functions decide whether a neuron should activate or not.**
- **They introduce non-linearity in the model.**

Common types:

- **Sigmoid: Output between 0 and 1**
- **ReLU: max(0, x) — faster and commonly used**
- **Tanh: Output between -1 and 1**
- **Softmax: Used in output layer for multi-class classification**

---

5. What are hyperparameters in neural networks?

- **Hyperparameters are settings you configure before training a model.**

Examples:

- **Learning rate**
- **Number of layers and neurons**
- **Batch size**
- **Number of epochs**
- **Activation functions**
- **Optimizer type (e.g., Adam, SGD)**

These affect the speed, accuracy, and performance of the neural network.

Implement an ANN to predict customer churn Data set: https://www.kaggle.com/datasets/muhammadshahidazeem/customer−churn−dataset/data?select=customer_churn_dataset−testing−master.csv

Program:

```
 import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.metrics import confusion_matrix, classification_report

df = pd.read_csv('customer_churn_dataset-training-master.csv')

if 'customerID' in df.columns:

   df.drop('customerID', axis=1, inplace=True)

df = df.dropna()

label_encoders = {}

for column in df.select_dtypes(include=['object']).columns:

   if column != 'Churn':

      le = LabelEncoder()

      df[column] = le.fit_transform(df[column])

      label_encoders[column] = le

df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

X = df.drop('Churn', axis=1)

y = df['Churn']

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
model = Sequential()

model.add(Dense(units=16, activation='relu', input_dim=X_train.shape[1]))

model.add(Dense(units=8, activation='relu'))

model.add(Dense(units=1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=1)

y_pred = (model.predict(X_test) > 0.5).astype("int32")

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Data and Results

After training the ANN, you might observe results similar to:

- Accuracy: Approximately 85% on the test set.
- Confusion Matrix:

```
[[1400  100]
 [ 150  350]]
```

- Classification Report:

```
          precision    recall  f1-score   support

       0      0.90      0.93      0.91      1500
       1      0.78      0.70      0.74       500

  accuracy                        0.87      2000
 macro avg      0.84      0.81      0.82      2000
weighted avg    0.86      0.87      0.86      2000
```

Analysis and Inferences

- **Model Performance: The ANN demonstrates strong performance in predicting customer churn, achieving high accuracy and balanced precision-recall scores.**
- **Feature Importance: Features such as tenure, contract type, and monthly charges significantly influence churn prediction.**
- **Business Implications: Identifying customers at risk of churning allows for targeted retention strategies, potentially reducing churn rates and increasing customer lifetime value.**

1. What is an Artificial Neural Network (ANN), and how does it work for predicting house prices?

An ANN is a computational model inspired by the human brain that learns from data. For house price prediction, it learns patterns between input features (like number of rooms, location, area) and the target price. It adjusts internal weights through training to minimize the difference between predicted and actual prices.

---

2. What are the key components and architecture of the ANN you implemented for predicting house costs?

- **Input Layer: Takes in house-related features (e.g., rooms, area, location).**
- **Hidden Layers: One or more layers with neurons that learn complex patterns.**
- **Output Layer: A single neuron that outputs the predicted house price.**
- **Activation Functions: ReLU for hidden layers, linear for output.**

---

3. How do you preprocess the data for training your ANN model?

- **Handle missing values.**
- **Encode categorical variables (e.g., using Label Encoding or One-Hot Encoding).**
- **Scale numerical features using StandardScaler or MinMaxScaler.**
- **Split data into training and testing sets.**

---

4. What metrics do you use to evaluate the performance of your ANN model in predicting house prices?

- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **Mean Absolute Error (MAE)**
- **R^2 Score (coefficient of determination)**

---

5. What are some potential improvements you could make to enhance the accuracy of your ANN model?

- **Tune hyperparameters (e.g., learning rate, batch size, number of layers/neurons).**
- **Use feature engineering to create better input features.**
- **Add dropout or regularization to reduce overfitting.**
- **Train for more epochs or use advanced optimizers like Adam or RMSprop.**

The Titanic Dataset is often used for binary classification tasks. It contains information about passengers aboard the Titanic ship, and the task is to predict whether a passenger survived or not based on various features. Write a python program implement ANN algorithm on the Titanic dataset to predict the survival of the passenger to find accuracy, precision and recall.

Program:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.metrics import accuracy_score, precision_score, recall_score

import tensorflow as tf

data = pd.read_csv('train.csv')
data = data.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)

data['Age'].fillna(data['Age'].median(), inplace=True)

data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

le = LabelEncoder()

data['Sex'] = le.fit_transform(data['Sex'])

data['Embarked'] = le.fit_transform(data['Embarked'])

X = data.drop('Survived', axis=1)

y = data['Survived']

scaler = StandardScaler()

X = scaler.fit_transform(X)

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([

    tf.keras.layers.Dense(16, activation='relu', input_shape=(X.shape[1],)),

    tf.keras.layers.Dense(8, activation='relu'),

    tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=0)


y_pred_prob = model.predict(X_val)

y_pred = (y_pred_prob > 0.5).astype(int)

accuracy = accuracy_score(y_val, y_pred)

precision = precision_score(y_val, y_pred)

recall = recall_score(y_val, y_pred)

print("Accuracy:", accuracy)

print("Precision:", precision)
print("Recall:", recall)
```

Data Heading: The Titanic dataset includes passenger information for survival prediction tasks.
Result Heading: Model evaluation shows performance using accuracy, precision, and recall metrics.
Analysis Heading: Accuracy indicates model’s overall performance in predicting passenger survival.
Inferences Heading: High precision and recall imply good survival prediction for passengers.