Experiment No . 2

AIM

To implement an Artificial Neural Network (ANN) for binary classification to predict customer churn.

OBJECTIVE

1. To preprocess the dataset, including handling missing values and encoding categorical variables.
2. To split the dataset into training and testing sets.
3. To perform feature scaling for better ANN performance.
4. To build and train an ANN with two hidden layers.
5. To evaluate the ANN's performance on the test set.
6. To make predictions using the trained ANN model.

THEORY

An Artificial Neural Network (ANN) is a computational model that mimics the functioning of biological neural networks in the human brain. ANNs are used for various tasks, including classification and regression, due to their ability to learn and generalize from data.

Pre-Requisites for Artificial Neural Network Implementation

Following will be the libraries and software that we will be needing in order to implement ANN.

1. Python – 3.6 or later
2. Jupyter Notebook (Google Colab can also be used)
3. Pandas
4. Numpy
5. Tensorflow 2. x
6. *Scikit-Learn*

Dataset Attributes

we use a dataset with the following 14 attributes:

1. **RowNumber**:- Represents the number of rows
2. **CustomerID**: Unique identifier for each customer.
3. **Surname**: Customer's surname.
4. **CreditScore**: Credit score of the customer.
5. **Geography**: Country of the customer.
6. **Gender**: Gender of the customer.
7. **Age**: Age of the customer.
8. **Tenure**: Number of years the customer has been with the bank.
9. **Balance**: Account balance of the customer.
10. **NumOfProducts**: Number of bank products the customer uses.
11. **HasCrCard**: Whether the customer has a credit card (1) or not (0).
12. **IsActiveMember**: Whether the customer is an active member (1) or not (0).
13. **EstimatedSalary**: Estimated salary of the customer.
14. **Exited**: Whether the customer has exited the bank (1) or not (0) - this is the dependent variable.

Preprocessing Steps

Step 1: Import Libraries

First, we need to import all the necessary libraries for data manipulation, model building, and evaluation.

```
import pandas as pdimport numpy as npfrom sklearn.model_selection import train_test_splitfrom sklearn.preprocessing import LabelEncoder, OneHo
```

Step 2: Load Dataset

Load the dataset into a pandas DataFrame. For this example, let's assume we have a CSV file named `data.csv`.

```
data = pd.read_csv('data.csv')
```

Step 3: Preprocess Data

This involves handling missing values, encoding categorical variables, and splitting the dataset into independent variables (X) and dependent variables (Y).

Extract Independent and Dependent Variables

```
X = data.iloc[:, 3:-1].values  # Assuming relevant features start from the 4th columnY = data.iloc[:, -1].values  # Assuming the last column i
```

Encode Categorical Variables

Encode Gender (binary variable)

```
labelencoder_X = LabelEncoder()X[:, 2] = labelencoder_X.fit_transform(X[:, 2])  # Assuming Gender is the third column
```

Encode Geography (non-binary variable)

```
ct = ColumnTransformer([("Geography", OneHotEncoder(), [1])], remainder='passthrough')  # Assuming Geography is the second columnX = np.array
```

Step 4: Split Dataset

Split the dataset into training and testing sets to evaluate the model's performance on unseen data.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

Step 5: Feature Scaling

Standardize the features to ensure that they have a mean of 0 and a standard deviation of 1. This helps in faster convergence during training.

```
sc = StandardScaler()X_train = sc.fit_transform(X_train)X_test = sc.transform(X_test)
```

Step 6: Build ANN

Initialize ANN

```
ann = tf.keras.models.Sequential()
```

Add First Hidden Layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Add Second Hidden Layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Add Output Layer

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Step 7: Compile ANN

Compile the ANN using the Adam optimizer and binary cross-entropy loss function. The accuracy metric will be used to evaluate the performance.

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

Step 8: Train ANN

Train the ANN on the training data for a specified number of epochs with a batch size of 32.

```
ann.fit(X_train, Y_train, batch_size=32, epochs=10)
```

Step 9: Evaluate and Predict

Evaluate the model's accuracy on the test set and make predictions on new data.

Evaluate on Test Set

```
test_loss, test_accuracy = ann.evaluate(X_test, Y_test)print(f'Test accuracy: {test_accuracy}')
```

Predict on New Data

For predicting new observations, it's crucial to apply the same scaling as done during training.

```
new_data = [[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]]  # Example new datanew_data_scaled = sc.transform(new_data)prediction = ann.predi
```

Conclusion

In this experiment we implemented an ANN with two hidden layers to predict customer churn. The preprocessing steps included encoding categorical variables and scaling features.