Developing a Neural Network Classification Model

EXP NO: 02

Date

Developing a Neural Network Classification Model

AIM

To develop a neural network classification model for the given dataset.

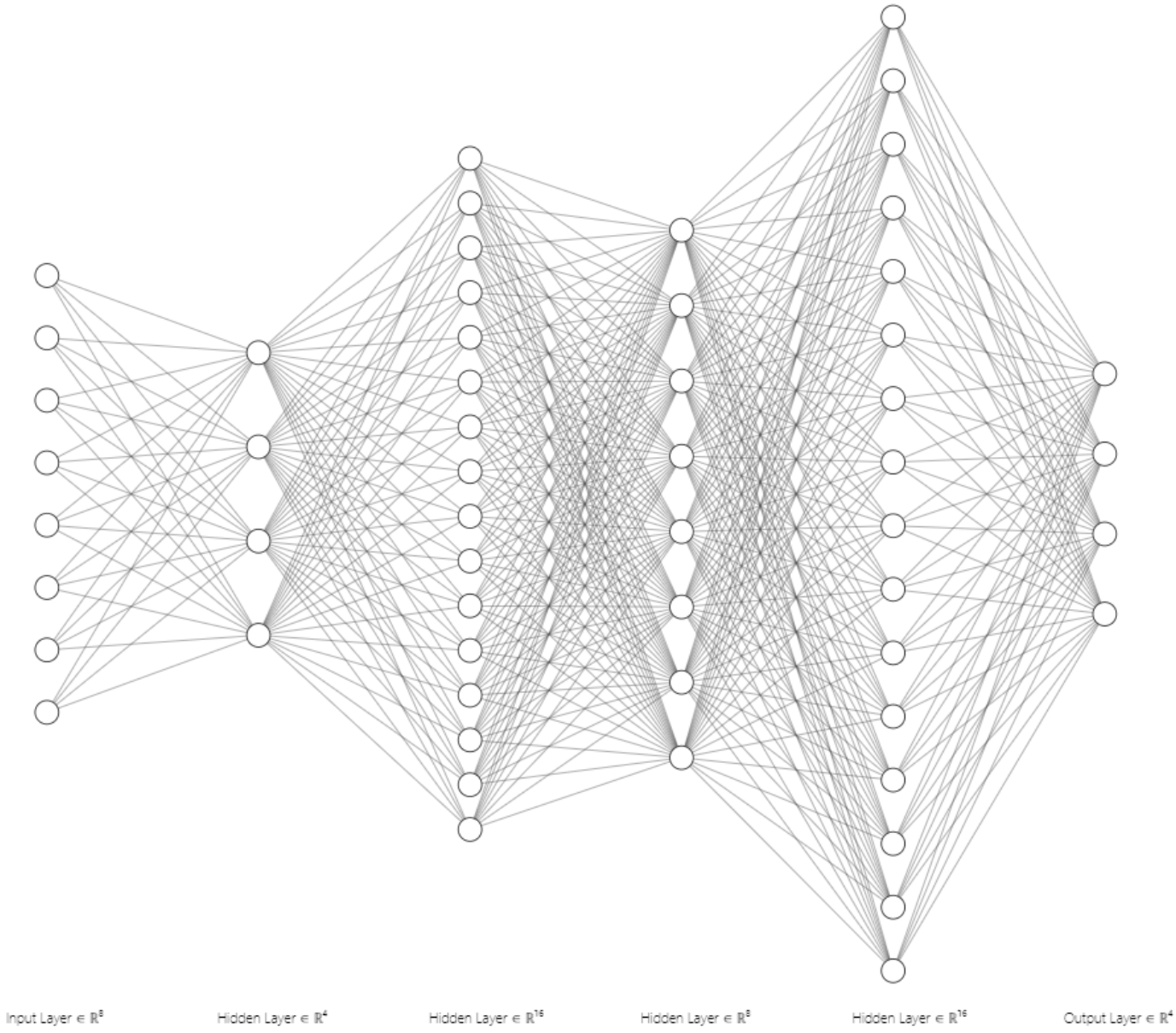
Problem Statement

An automobile company has plans to enter new markets with their existing products. After intensive market research, they've decided that the behavior of the new market is similar to their existing market.

In their existing market, the sales team has classified all customers into 4 segments (A, B, C, D). Then, they performed segmented outreach and communication for a different segment of customers. This strategy has work exceptionally well for them. They plan to use the same strategy for the new markets.

You are required to help the manager to predict the right group of the new customers.

Neural Network Model



DESIGN STEPS

STEP 1:

Loading the dataset.

STEP 2:

Checking the null values and converting the string datatype into integer or float datatype using label encoder.

STEP 3:

Split the dataset into training and testing.

STEP 4:

Create MinMaxScaler objects,fit the model and transform the data.

STEP 5:

Include screenshot of the dataset

STEP 6:

Train the model with the training data.

STEP 7:

Include Classification Report here Plot the training loss and validation loss.

STEP 8:

Include confusion matrix here Predicting the model through classification report,confusion matrix.

STEP 9:

Predict the new sample data.

PROGRAM

```
# Developed By: Balaji N
# Register Number:212220230006
# Import required packages
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
import pickle
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import BatchNormalization
import tensorflow as tf
import seaborn as sns
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import classification_report,confusion_matrix
import numpy as np
import matplotlib.pylab as plt
# Load the csv file
df=pd.read_csv("customers.csv")
```

```
df.head()
df.columns
df.dtypes
df.shape
# Check for null values and clean the dataset
df.isnull().sum()
df_clean=df.dropna(axis=0)
df_clean.isnull().sum()
df_clean.shape
df_clean.dtypes
df_clean['Gender'].unique()
df_clean['Ever_Married'].unique()
df_clean['Graduated'].unique()
df_clean['Profession'].unique()
df_clean['Spending_Score'].unique()
df_clean['Var_1'].unique()
df_clean['Segmentation'].unique()
# Apply encoders
category_list=[
    ['Male', 'Female'],
    ['No', 'Yes'],
    ['No', 'Yes'],
    ['Healthcare', 'Engineer', 'Lawyer', 'Artist', 'Doctor',
      'Homemaker', 'Entertainment', 'Marketing', 'Executive'],
     ['Low', 'High', 'Average']
]
enc = OrdinalEncoder(categories=category_list)
df1=df_clean.copy()
df1[['Gender',
             'Ever_Married',
              'Graduated','Profession',
              'Spending_Score']] = enc.fit_transform(df1[['Gender',
                                                          'Ever_Married',
                                                          'Graduated','Profession',
                                                          'Spending_Score']])
df1.dtypes
le = LabelEncoder()
df1['Segmentation'] = le.fit_transform(df1['Segmentation'])
df1.dtypes
df1 = df1.drop('ID',axis=1)
df1 = df1.drop('Var_1',axis=1)
df1.dtypes
corr = df1.corr()
# Find co-relation between fields
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns,
        cmap="BuPu",
        annot= True)
sns.pairplot(df1)
sns.distplot(df1['Age'])
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
```

```
sns.countplot(df1['Family_Size'])
plt.figure(figsize=(10,6))
sns.boxplot(x='Family_Size',y='Age',data=df1)
plt.figure(figsize=(10,6))
sns.scatterplot(x='Family_Size',y='Spending_Score',data=df1)
plt.figure(figsize=(10,6))
sns.scatterplot(x='Family_Size',y='Age',data=df1)
df1.describe()
df1['Segmentation'].unique()
x=df1[['Gender','Ever_Married','Age','Graduated','Profession','Work_Experience','Spending_Sc
y1 = df1[['Segmentation']].values
one_hot_enc = OneHotEncoder()
one_hot_enc.fit(y1)
y1.shape
y = one_hot_enc.transform(y1).toarray()
y.shape
y1[0]
y[0]
x.shape
# train the model
x_train,x_test,y_train,y_test=train_test_split(x,y,
                                               test_size=0.33,
                                               random_state=50)
x_train[0]
x_train.shape
scaler_age = MinMaxScaler()
scaler_age.fit(x_train[:,2].reshape(-1, 1))
x_train_scaled = np.copy(x_train)
x_test_scaled = np.copy(x_test)
x_train_scaled.shape
# To scale the Age column
x_train_scaled[:,2] = scaler_age.transform(x_train[:,2].reshape(-1, 1)).reshape(-1)
x_test_scaled[:,2] = scaler_age.transform(x_test[:,2].reshape(-1,1)).reshape(-1)
# Creating the model
ai_brain = Sequential([
    Dense(8,input_shape=[8]),
    Dense(4,activation='relu'),
    Dense(16,activation='tanh'),
    Dense(8,activation='relu'),
    Dense(16,activation='tanh'),
    Dense(4,activation='softmax')
])
ai_brain.compile(optimizer='adam',
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=2)
ai_brain.fit(x_train_scaled,y_train,
             epochs=2000,batch_size=256,
             validation_data=(x_test_scaled,y_test),
             callbacks=[early_stop]}
 metrics = pd.DataFrame(ai_brain.history.history)
 metrics.tail()
 metrics.plot()
```

```
 # Predictions
x_test_predictions = np.argmax(ai_brain.predict(x_test_scaled), axis=1)
x_test_predictions.shape
y_test_truevalue = np.argmax(y_test,axis=1)
y_test_truevalue.shape
print(confusion_matrix(y_test_truevalue,x_test_predictions))
print(classification_report(y_test_truevalue,x_test_predictions))
x_single_prediction = np.argmax(ai_brain.predict(x_test_scaled[1:2,:]), axis=1)
print(x_single_prediction)
print(le.inverse_transform(x_single_prediction))
```

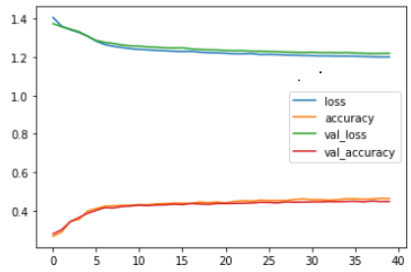
Dataset Information

	ID	Gender	Ever_Marr	Age	Graduated	Profession	Work_Exp	Spending_	Family_Siz	Var_1	Segmentation
2	462809	Male	No	22	No	Healthcare	1	Low	4	Cat_4	D
3	462643	Female	Yes	38	Yes	Engineer		Average	3	Cat_4	A
4	466315	Female	Yes	67	Yes	Engineer	1	Low	1	Cat_6	B
5	461735	Male	Yes	67	Yes	Lawyer	0	High	2	Cat_6	B
6	462669	Female	Yes	40	Yes	Entertainment		High	6	Cat_6	A
7	461319	Male	Yes	56	No	Artist	0	Average	2	Cat_6	C
8	460156	Male	No	32	Yes	Healthcare	1	Low	3	Cat_6	C
9	464347	Female	No	33	Yes	Healthcare	1	Low	3	Cat_6	D
10	465015	Female	Yes	61	Yes	Engineer	0	Low	3	Cat_7	D
11	465176	Female	Yes	55	Yes	Artist	1	Average	4	Cat_6	C
12	464041	Female	No	26	Yes	Engineer	1	Low	3	Cat_6	A
13	464942	Male	No	19	No	Healthcare	4	Low	4	Cat_4	D
14	461230	Female	No	19	No	Executive	0	Low		Cat_3	D
15	459573	Male	Yes	70	No	Lawyer		Low	1	Cat_6	A
16	460849	Female	Yes	58	No	Doctor	0	Low	1	Cat_3	B
17	460563	Female	No	41	No	Healthcare	1	Low	2	Cat_1	C
18	466865	Female	No	32	No	Homemak	9	Low	5	Cat_3	D
19	461644	Male	No	31	No	Healthcare	1	Low	6	Cat_6	B
20	466772	Male	Yes	58	Yes	Entertainm	1	Average	4	Cat_6	B
21	464291	Female	Yes	79	Yes	Artist	0	High	1	Cat_6	C
22	466084	Male	Yes	49	Yes	Homemak	12	Low	1	Cat_3	A
23	459675	Female	No	18	No	Healthcare	3	Low	4	Cat_6	D
24	465602	Male	Yes	33	Yes	Artist	13	Low	2	Cat_3	A
25	459168	Female	No	36	Yes	Artist	5	Low	2	Cat_6	B
26	461021	Female		58	No	Executive	1	Average	3	Cat_3	B
27	465083	Male	Yes	56	No	Artist	1	Average	3	Cat_6	C

OUTPUT

Training Loss, Validation Loss Vs Iteration Plot

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0753154490>
```



Classification Report

```
print(classification_report(y_test_truevalue,x_test_predictions)
```

```
              precision    recall  f1-score   support

           0       0.38      0.54      0.44       757
           1       0.30      0.09      0.14       686
           2       0.47      0.51      0.49       686
           3       0.55      0.62      0.58       782

    accuracy                           0.45      2911
   macro avg       0.42      0.44      0.41      2911
weighted avg       0.43      0.45      0.42      2911
```

Confusion Matrix

```
[119] print(confusion_matrix(y_test_truevalue,x_test_predictions))
```

```
[[405  47 132 173]
 [295  63 219 109]
 [155  74 351 106]
 [224  25  51 482]]
```

New Sample Data Prediction

```
[111] x_single_prediction = np.argmax(ai_brain.predict(x_test_scaled[1:2,:]), axis=1
```

```
[112] print(x_single_prediction)
```

```
[2]
```

```
print(le.inverse_transform(x_single_prediction))
```

```
['C']
```

Include your sample input and output here

RESULT

Thus,a neural network classification model for the given dataset is developed.