Ex No: 2	Implement single layer neural network for binary classification

AIM:

To implement single layer neural network for binary classification.

PROCEDURE:

1. Import the required libraries and packages.
2. Load the dataset.
3. Preprocess the dataset e.g.: Remove the null values.
4. Split the data into training and testing.
5. Drop Time and Class columns.
6. Build a Binary Classifier.
7. Train and Test the input data with the binary classifier.
8. Plot the training and validation accuracy using the per-epoch values in the history object (histogram plot).
9. Plot confusion matrix as heatmap.

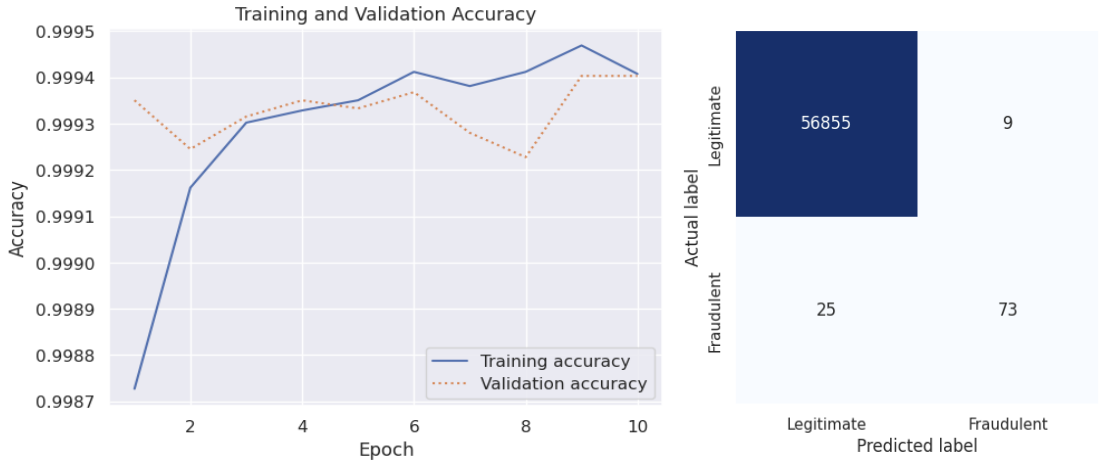
CODE:

```
# importing useful libraries

import pandas as pd

# load the dataset

df = pd.read_csv('creditcard.csv', low_memory=False)

df.head(10)

# Preprocessing

df.info()

df = df.convert_dtypes()

print(df.dtypes)

df.isnull().sum()

mean = df['V22'].mode().values[0]

df['V22'].fillna(value=mean, inplace=True)

mean = df['V23'].mode().values[0]

df['V23'].fillna(value=mean, inplace=True)

mean = df['V24'].mode().values[0]

df['V24'].fillna(value=mean, inplace=True)

mean = df['V25'].mode().values[0]

df['V25'].fillna(value=mean, inplace=True)

mean = df['V26'].mode().values[0]

df['V26'].fillna(value=mean, inplace=True)
```

```
mean = df['V27'].mode().values[0]
df['V27'].fillna(value=mean, inplace=True)
mean = df['V28'].mode().values[0]
df['V28'].fillna(value=mean, inplace=True)
mean = df['Amount'].mode().values[0]
df['Amount'].fillna(value=mean, inplace=True)
mean = df['Class'].mode().values[0]
df['Class'].fillna(value=mean, inplace=True)
df.isnull().sum().sum()
# Splitting the data into training and testing & Dropping Time and Class columns
from sklearn.model_selection import train_test_split
x = df.drop(['Time', 'Class'], axis=1)
y = df['Class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=0)
# Building a Binary Classifier
#Create a neural network for binary classification
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(128, activation='relu', input_dim=29))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
import numpy as np
x_train = np.asarray(x_train).astype(np.float32)
y_train = np.asarray(y_train).astype(np.float32)
x_test = np.asarray(x_test).astype(np.float32)
y_test = np.asarray(y_test).astype(np.float32)
# Training and Testing the input data with the binary classifier
hist = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=100)
# Plotting the training and validation accuracy using the per-epoch values in the history object
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
sns.set()
acc = hist.history['accuracy']
val = hist.history['val_accuracy']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, '-', label='Training accuracy')
plt.plot(epochs, val, ':', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.plot()

# Plotting confusion matrix
from sklearn.metrics import confusion_matrix
y_predicted = model.predict(x_test) > 0.5
mat = confusion_matrix(y_test, y_predicted)
labels = ['Legitimate', 'Fraudulent']
sns.heatmap(mat, square=True, annot=True, fmt='d', cbar=False, cmap='Blues',
            xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted label')
plt.ylabel('Actual label')
```

OUTPUT:



RESULTS:

No of epochs	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
1	0.9987	0.9994	0.0219	0.0058
2	0.9992	0.9992	0.0208	0.0111
3	0.9993	0.9993	0.0081	0.0053
4	0.9993	0.9994	0.0078	0.0094
5	0.9994	0.9993	0.0094	0.0049
6	0.9994	0.9994	0.0061	0.0042
7	0.9994	0.9993	0.0094	0.0084
8	0.9994	0.9992	0.0056	0.0046
9	0.9995	0.9994	0.0097	0.0041
10	0.9994	0.9994	0.0047	0.0037

CONCLUSION:

A single-layer neural network for binary classification, distinguishing transactions as fraudulent or non-fraudulent, has been implemented successfully.