Experiment No. 4
Design a fully connected deep neural network with at least 2 hidden layers on Titanic survival classification by choosing appropriate Learning Algorithms
Date of Performance:
Date of Submission:

Aim: Design a fully connected deep neural network with at least 2 hidden layers on Titanic survival classification by choosing appropriate Learning Algorithms.

Objective: Ability to perform experimentations and deciding upon the best learning algorithm for a 3 layered neural network.

Theory:

In deep learning, we have the concept of loss, which tells us how poorly the model is performing at that current instant. Now we need to use this loss to train our network such that it performs better. Essentially what we need to do is to take the loss and try to minimize it, because a lower loss means our model is going to perform better. The process of minimizing (or maximizing) any mathematical expression is called optimization.

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

How do Optimizers work?

For a useful mental model, you can think of a hiker trying to get down a mountain with a blindfold on. It's impossible to know which direction to go in, but there's one thing she can know: if she's going down (making progress) or going up (losing progress). Eventually, if she keeps taking steps that lead her downwards, she'll reach the base.

Similarly, it's impossible to know what your model's weights should be right from the start. But with some trial and error based on the loss function (whether the hiker is descending), you can end up getting there eventually.

How you should change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use. Optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible.

Various optimizers are researched within the last few couples of years each having its advantages and disadvantages. Read the entire article to understand the working, advantages, and disadvantages of the algorithms.

Algorithms:

Gradient Descent

Stochastic Gradient Descent (SGD)

Mini Batch Stochastic Gradient Descent (MB-SGD)

SGD with momentum

Nesterov Accelerated Gradient (NAG)

Adaptive Gradient (AdaGrad)

AdaDelta

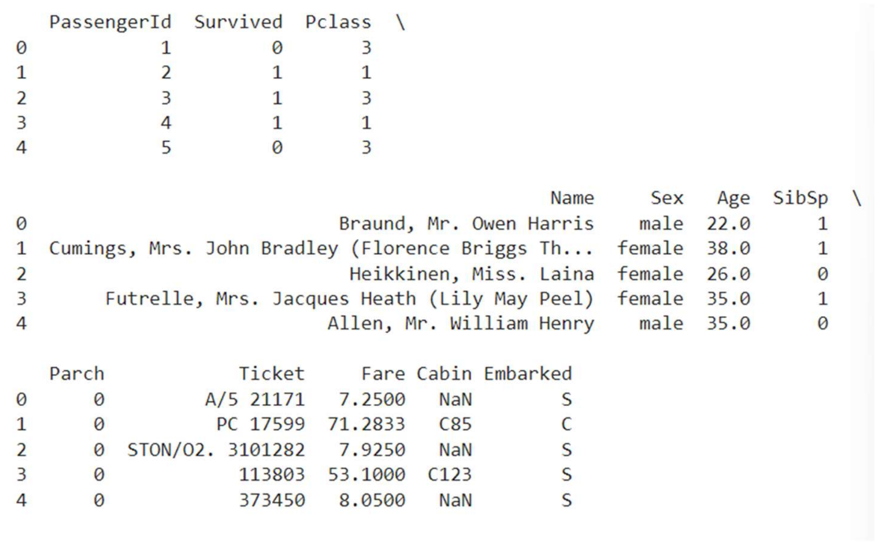
RMSprop

Adam

Program:

```
import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import accuracy_score


url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"

titanic_data = pd.read_csv(url)
```

print(titanic_data.head())



```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

titanic_data = titanic_data.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin'])

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

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

titanic_data = pd.get_dummies(titanic_data, columns=['Sex', 'Embarked'], drop_first=True)

X = titanic_data.drop(columns=['Survived'])

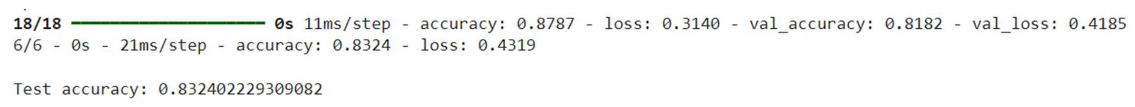
y = titanic_data['Survived']

scaler = StandardScaler()

X = scaler.fit_transform(X)

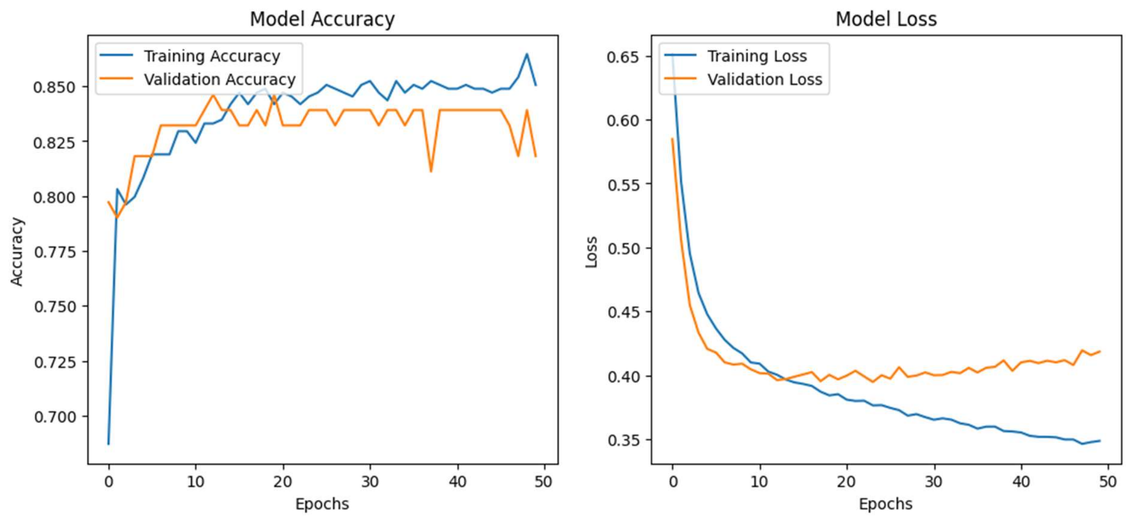
```
encoder = LabelEncoder()

y = encoder.fit_transform(y)



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



model = models.Sequential()





model.add(layers.InputLayer(input_shape=(X_train.shape[1],)))



model.add(layers.Dense(64, activation='relu'))



model.add(layers.Dense(32, activation='relu'))

on

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



model.compile(optimizer='adam',

        loss='binary_crossentropy',

        metrics=['accuracy'])



history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)

print(f'\nTest accuracy: {test_acc}')
```



```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8787 - loss: 0.3140 - val_accuracy: 0.8182 - val_loss: 0.4185
6/6 - 0s - 21ms/step - accuracy: 0.8324 - loss: 0.4319

Test accuracy: 0.832402229309082
```

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



accuracy = accuracy_score(y_test, y_pred)

print(f"Test Accuracy: {accuracy}")



import matplotlib.pyplot as plt



plt.figure(figsize=(12, 5))



plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend(loc='upper left')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend(loc='upper left')

plt.show()
```



Conclusion:

Calculate and comment on the accuracy and reasons for choosing a particular learning algorithm.

The accuracy of a learning algorithm is calculated by dividing the number of correct predictions by the total number of predictions, expressed as a percentage. The choice of a specific learning algorithm depends on the nature of the data, the problem at hand, and the desired trade-offs between accuracy, interpretability, and computational efficiency. For instance, if the data is high-dimensional and complex, a deep learning algorithm like a CNN might be chosen for its ability to learn intricate patterns, whereas a simpler algorithm like logistic regression might be preferred for smaller datasets with a need for quick and interpretable results. The goal is to select an algorithm that best balances these factors to achieve high accuracy while meeting other project requirements.