CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS (CUCEI)

DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Carrera: Ingeniería en Computación

Nombre Materia: Seminario de Solución de Problemas de IA ll

Profesor: Valdez López Julio Esteban

SECCIÓN: D02

Nombre alumno: López Arellano Ricardo David

CODIGO: 217136143

Ejercicio 4

Fecha de entrega: 01/11/2023

Mejorar el desempeño del siguiente programa utilizando una red neuronal multicapa Y EL DATASET fashion_mnist = keras.datasets.fashion_mnist.load_data() :

```
from sklearn.datasets import fetch_openml
from sklearn.base import BaseEstimator

import numpy as np
class MeValeMLV(BaseEstimator):
    def fit(self, X, y=None):
        return self
    def predict(self, X):
        return np.zeros((len(X), 1), dtype = bool)
#download training data

print("Downloading data...")
mnist = fetch_openml('mnist_784', version=1)
print("...download complete")
print(mnist.keys())

x, y = mnist['data'], mnist['target']

x_train, x_test, y_train, y_test = x[:6000], x[6000:],y[:6000], y[6000:]

y_train_5 = (y_train == '5')

#creating stochastic gradient descent classifier
from sklearn.linear_model import SGDClassifier
#from sklearn.svm import SVC
sgd_clf = SGDClassifier( random_state = 42 )
#sgd_clf = SVC(gamma='auto')

print("starting training...")
sgd_clf.fit(x_train, y_train_5)
print("...training complete!")


#testing single input

print(sgd_clf.predict([x_train.iloc[0]]))

import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

some_number = np.array(x_train.iloc[0])
image = some_number.reshape(28,28)

plt.imshow(image, cmap='binary')
plt.axis('off')
```

```
plt.show()

#cross validation

from sklearn.model_selection import StratifiedKFold
from sklearn.base import clone

skfolds = StratifiedKFold( n_splits = 3, shuffle = True, random_state =
42)

for train_index, test_index in skfolds.split(x_train, y_train_5):
    clone_clf = clone(sgd_clf)

    x_train_folds = x_train.loc[train_index]
    y_train_folds = y_train_5.loc[train_index]
    x_test_fold = x_train.loc[test_index]
    y_test_fold = y_train_5.loc[test_index]

    clone_clf.fit(x_train_folds,y_train_folds)
    y_pred = clone_clf.predict(x_test_fold)
    n_correct = sum( y_pred == y_test_fold )
    print(n_correct / len(y_pred))

#confusion matrix
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
y_train_pred = cross_val_predict(sgd_clf, x_train, y_train_5, cv=3)


conf_mat = confusion_matrix(y_train_5, y_train_pred)

plt.matshow(conf_mat,cmap = plt.cm.gray)
plt.show()

#f1 score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score, recall_score

print("Presicion:",precision_score(y_train_5, y_train_pred))
print("Recall:",recall_score(y_train_5, y_train_pred))
print("F1 score",f1_score(y_train_5, y_train_pred))
```

1. Preparar los datos del dataset fashion_mnist:

```
from tensorflow import keras

# Cargar el dataset fashion_mnist
fashion_mnist = keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

```
# Normalizar las imágenes
X_train, X_test = X_train / 255.0, X_test / 255.0
```

2. Definir y compilar el modelo de red neuronal:

```
model = keras.models.Sequential([

    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

# Compilar el modelo
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

3. Entrenar el modelo con los datos de entrenamiento:

```
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

4. Realizar las predicciones y evaluar el rendimiento:

```
# Realizar predicciones en los datos de prueba
y_pred = model.predict(X_test)

# Evaluar el rendimiento del modelo
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score,
precision_score, recall_score

y_pred_binary = (y_pred > 0.5).astype(int)
confusion = confusion_matrix(y_test, y_pred_binary)
accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print("Confusion Matrix:")
print(confusion)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Este código utiliza una red neuronal multicapa para resolver el problema de clasificación en el conjunto de datos fashion_mnist y calcula métricas de rendimiento, como la matriz de confusión, precisión, recuperación y puntuación F1.