-1

ایمپورت لایبری ها :

```
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

لود کردن دیتا:

```
# Replace 'your_file.csv' with the actual path to your CSV file
file_path = 'airplane.csv'

# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Display the DataFrame
print(df)
```

حذف ستون اول و بقیه ستون هایی که تاثیری در رضایت مشتری ندارند:

```
# Assume you want to delete the column at index 2 (replace with the actual index)
column_index_to_delete = 0

# Delete the column by index
df = df.drop(df.columns[column_index_to_delete], axis=1)
```

```
columns_to_delete = ['id', 'Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', ]

# Drop the specified columns
df = df.drop(columns=columns_to_delete)
```

چون ستون رضایت string است به باینری تبدیل میکنیم :

```
print(df['satisfaction'].unique())

# Replace 'neutral' or 'dissatisfied' with 0, and 'satisfied' with 1
df['satisfaction'] = df['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1})

# Check unique values again to verify the changes
print(df['satisfaction'].unique())
```

```
['neutral or dissatisfied' 'satisfied']
[0 1]
```

برای بقیه ستون ها اگر بالای میانه باشد ۱ و اگر کمتر از آن باشد ۰ می گذاریم :

```
average_delay = df['Cleanliness'].mean()

# Replace values based on the condition
df['Cleanliness'] = df['Cleanliness'].apply(lambda x: 1 if x > average_delay else 0)
```

```
average_delay = df['Checkin service'].mean()

# Replace values based on the condition
df['Checkin service'] = df['Checkin service'].apply(lambda x: 1 if x > average_delay else 0)
```

در نهایت دیتا رو به تست و ترین تقسیم میکنیم :

```
target_col = 'satisfaction'

# Extract features and target variable
X = df.drop(columns=[target_col])
y = df[target_col]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2000)

# Standardize features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Convert data to PyTorch tensors
X_train_tensor = torch.Tensor(X_train)
y_train_tensor = torch.Tensor(y_train.values).view(-1, 1)

X_test_tensor = torch.Tensor(X_test)
y_test_tensor = torch.Tensor(y_test.values).view(-1, 1)
```

یک شبکه عصبی ساده طراحی میکنیم و برای آن مدل و لاس فانکشن و اپتیمایز تعیین میکنیم:

```
# Define a simple neural network model
class SatisfactionModel(nn.Module):
    def __init__(self, input_size):
        super(SatisfactionModel, self).__init__()
        self.fc1 = nn.Linear(input_size, 64)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(64, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.sigmoid(x)
        return x

# Instantiate the model, loss function, and optimizer
input_size = X_train.shape[1]
model = SatisfactionModel(input_size)
criterion = nn.BCELoss()  # Binary Cross Entropy loss for binary classification
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

سپس آن را ترین میکنیم و دقت را نمایش میدهیم:

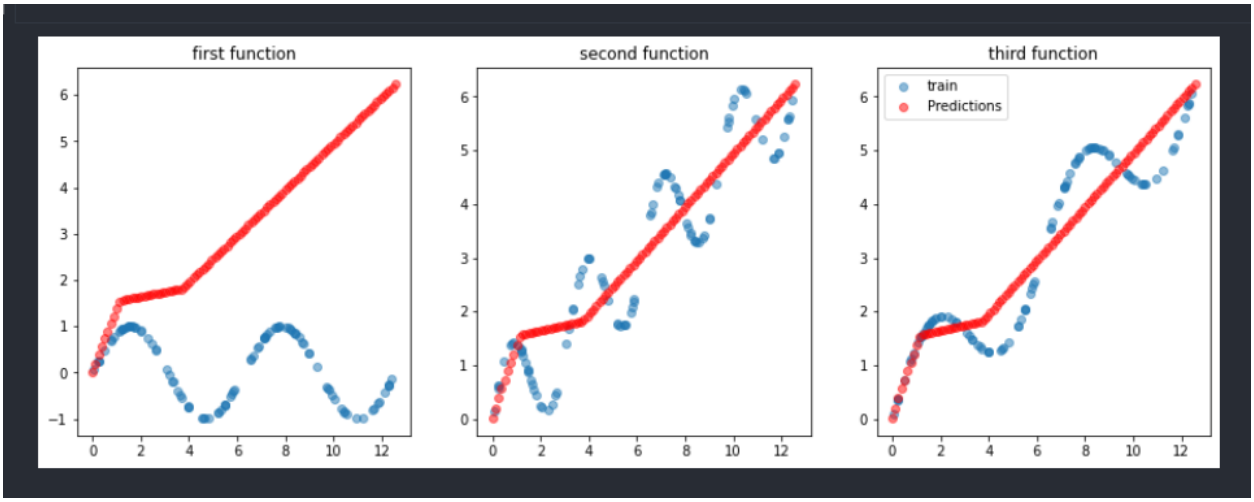
```
# Instantiate the model, loss function, and optimizer
input_size = X_train.shape[1]
model = SatisfactionModel(input_size)
criterion = nn.BCELoss()  # Binary Cross Entropy loss for binary classification
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training the neural network
epochs = 1000
for epoch in range(epochs):
    model.train()
    y_pred = model(X_train_tensor)
    loss = criterion(y_pred, y_train_tensor)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if epoch % 10 == 0:
        print(f'Epoch {epoch+1}/{epochs}, Loss: {loss.item()}')

# Evaluate the model on the test set
model.eval()
with torch.no_grad():
    y_pred_test = model(X_test_tensor)
    y_pred_binary = (y_pred_test > 0.5).float()

# Calculate accuracy on the test set
accuracy = (y_pred_binary == y_test_tensor).float().mean().item()
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

-2

تعریف تابع و تولید داده‌های آموزشی و تعریف مدل mlp

```python
# تعریف مدل MLP
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(1, 10)  # لایه ورودی به لایه مخفی
        self.relu = nn.ReLU()        # تابع فعال‌سازی ReLU
        self.fc2 = nn.Linear(10, 1)  # لایه مخفی به لایه خروجی

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x

# ایجاد مدل
model = MLP()
```

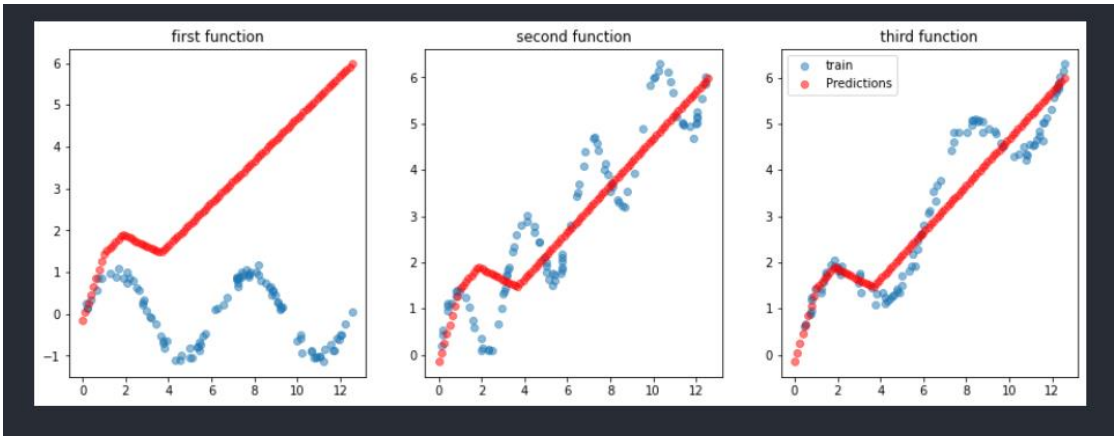
سپس مدل را آموزش میدهیم و نتیجه را نمایش می دهیم :




-3

همین کار ها را صرفا با اضافه کردن نویز انجام می دهیم.

```python
# تولید داده‌های آموزشی
def generate_data(func, num_samples, noise_std):
    x = np.sort(4 * np.pi * np.random.rand(num_samples))  # 4 و 0 نمونه‌های تصادفی بین pi*
    y = func(x) + noise_std * np.random.randn(num_samples)  # افزودن نویز
    return x, y
```

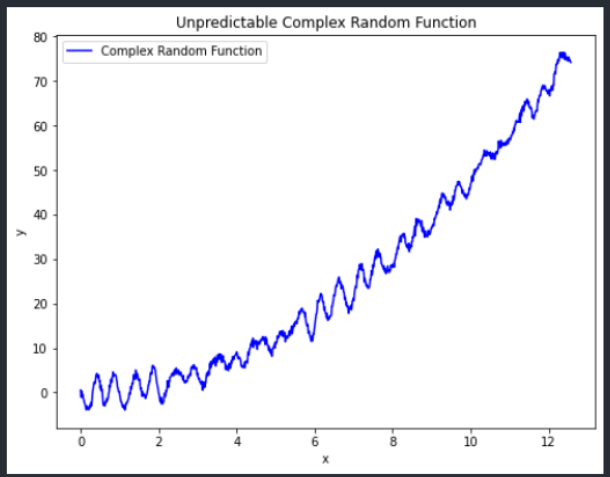



4-

در ابتدا یک تابع رندوم رو تولید می کنیم :

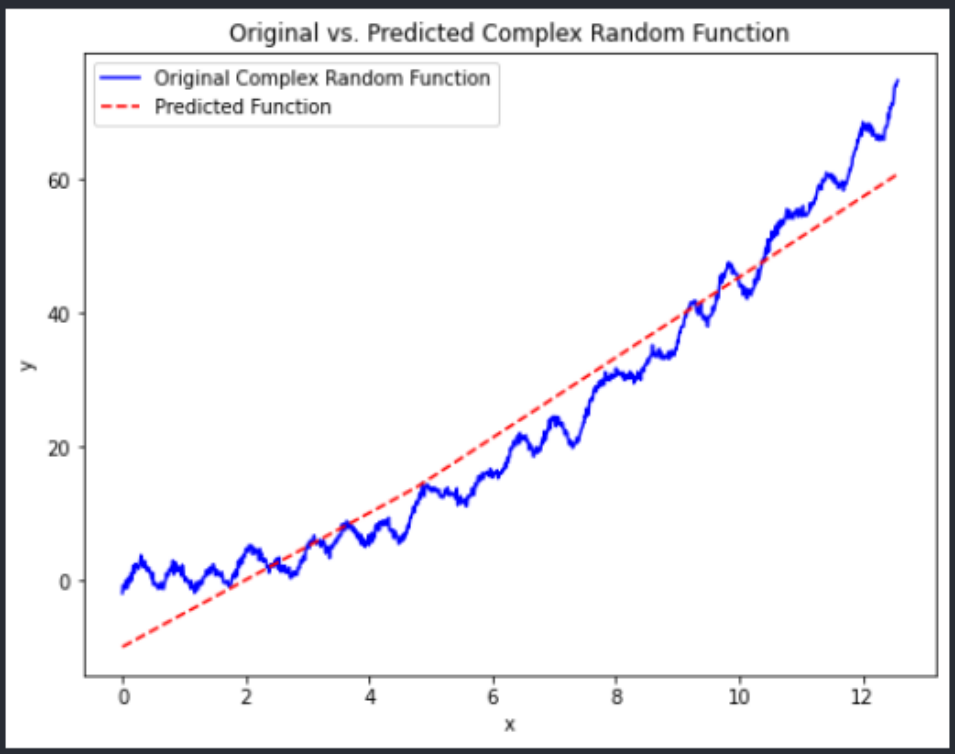
```python
# تولید داده‌های تابع تصادفی پیچیده
x_complex = np.linspace(0, 4 * np.pi, 1000)
y_complex = complex_random_function(x_complex)

# نمایش تابع تصادفی پیچیده
plt.figure(figsize=(8, 6))
plt.plot(x_complex, y_complex, label='Complex Random Function', color='blue')
plt.title('Unpredictable Complex Random Function')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

[91]



یک شبکه عصبی ساده طراحی میکنیم و برای آن مدل و لاس فانکشن و اپتیمایز تعیین میکنیم. سپس آن را ترین می کنیم و نتیجه رو نمایش میدهیم :

```
epoch 1/1000, error: 1065.4736328125
epoch 101/1000, error: 62.60830307006836
epoch 201/1000, error: 31.877361297607422
epoch 301/1000, error: 31.82147216796875
epoch 401/1000, error: 31.704879760742188
epoch 501/1000, error: 31.441978454589844
epoch 601/1000, error: 30.87163734436035
epoch 701/1000, error: 29.757511138916016
epoch 801/1000, error: 28.08194923400879
epoch 901/1000, error: 26.285066604614258
```



6-ایمپورت لایبری ها :

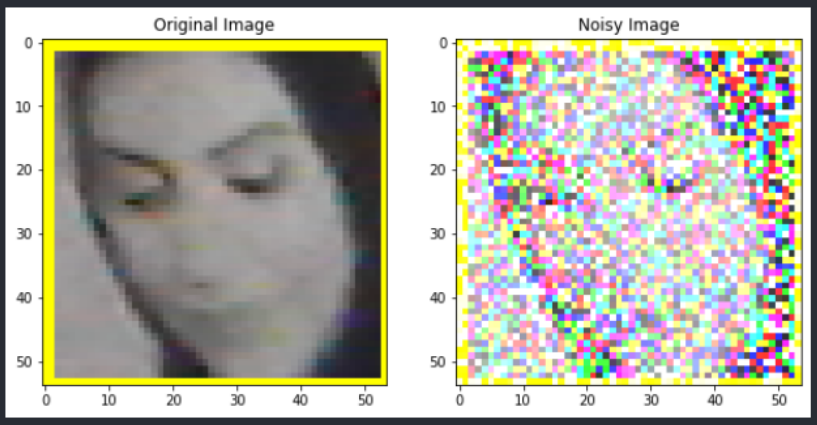
```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import torch
import torch.nn as nn
```

لود کردن عکس :

```
# Open a simple image from Image folder
image = cv2.imread("test_image_folder/test/test_image.png")
```

اضافه کردن نویز :

```
# Add Gaussian noise
mean = 0
std_dev = 25
noise = np.random.normal(mean, std_dev, image.shape).astype('uint8')
noisy_image = cv2.add(image, noise)

# Display the original and noisy images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(noisy_image)
plt.title('Noisy Image')

plt.show()
```



یک شبکه عصبی ساده طراحی میکنیم و برای آن مدل و لاس فانکشن و اپتیمایز تعیین میکنیم. سپس آن را ترین می کنیم و نتیجه رو نمایش میدهیم :

```python
# Display the noisy and denoised images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(noisy_image)
plt.title('Noisy Image')

plt.subplot(1, 2, 2)
plt.imshow(denoised_image)
plt.title('Denoised Image')

plt.show()
```