Nama : Yoga Patangga Balapradhana
NIM : 1103194138

Penjelasan Regression Task

Data yang digunakan pada Regression Task ini adalah data RegresiUTSTelkom.csv yang disediakan oleh Teaching Assistant. Mula-mula file csv tersebut di-copy terlebih dahulu ke dalam Google Drive, untuk kemudian dapat dibaca dengan menggunakan baris perintah:

```
# Menghubungkan Google Colab dengan Google Drive
from google.colab import drive

# Mount Google Drive ke Colab
drive.mount('/content/drive')
file_path = '/content/drive/MyDrive/RegresiUTSTelkom.csv'
df = pd.read_csv(file_path)
print(df.head())
```

Setelah itu dilakukan eksplorasi data yang sudah dimuat tersebut menggunakan perintah:

```
print(df.shape)
print(df.info())
X = df.drop("2001", axis=1)  # Features
y = df["2001"]   # Target
```

```
(515344, 91)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515344 entries, 0 to 515343
Data columns (total 91 columns):
```

Sehingga dapat diketahui data tersebut memiliki 515.344 baris dan 91 kolom dengan tipe data integer pada kolom pertama (2001) dan tipe data float pada kolom sisanya. Kemudian setelah melihat data statistik dengan menggunakan perintah `df.describe()`, maka dapat disimpulkan kolom pertama adalah data tahun (dari tahun 1922 hingga tahun 2011) sehingga kolom tersebut dipilih sebagai target dan kolom sisanya sebagai features untuk diproses lebih lanjut.

Langkah selanjutnya adalah membagi dataset menjadi data latih dan data uji dengan persentase 70:30 menggunakan perintah `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)`. Setelah itu dilakukan feature engineering pada data latih dan data uji sebanyak 3 langkah, yaitu 1) melakukan scaling features menggunakan StandardScaler, 2) mengubah fitur menjadi bentuk PyTorch tensor, dan 3) membuat DataLoader untuk data latih dan data uji.

Setelah dilakukan feature engineering, maka langkah selanjutnya adalah membuat model MLP menggunakan fungsi aktivasi RelU dengan baris perintah berikut:

Nama : Yoga Patangga Balapradhana
NIM : 1103194138

```
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(MLP, self).__init__()
        self.layer1 = nn.Linear(input_size, hidden_size)  # First hidden layer
        self.layer2 = nn.Linear(hidden_size, output_size)  # Output layer

    def forward(self, x):
        x = torch.relu(self.layer1(x))  # Apply ReLU activation
        x = self.layer2(x)  # Output layer
        return x
```

Langkah berikutnya tentukan nilai inisiasi seperti input size, hidden size, output size, loss function, dan optimizer menggunakan baris perintah berikut:

```
# Define the model, loss function, and optimizer
input_size = X_train.shape[1]
hidden_size = 64
output_size = 1  # Single output for regression
model = MLP(input_size, hidden_size, output_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Pada eksperimen ini, ditentukan nilai hidden size = 64, output size = 1 (karena model regresi), loss function MSE, dan menggunakan optimizer Adam. Setelah itu, lakukan training loop sebanyak 100 epoch dengan menggunakan baris perintah berikut:

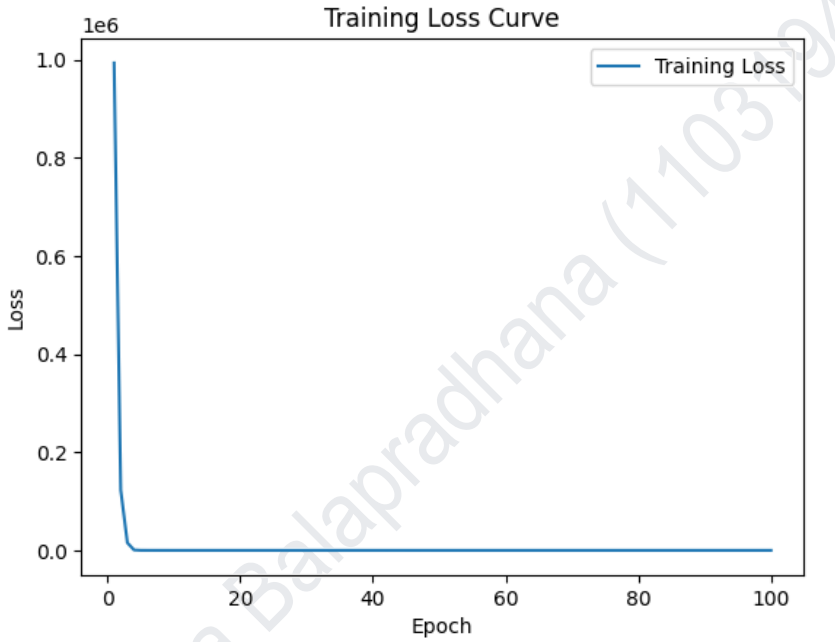
```
num_epochs = 100  # Number of epochs for training

train_losses = []  # Store the loss values to plot

for epoch in range(num_epochs):
    model.train()  # Set the model to training mode
    running_loss = 0.0

    for inputs, labels in train_loader:
        # Zero the gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(inputs)
        loss = criterion(outputs, labels)

        # Backward pass and optimization
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    # Compute average loss for this epoch
    avg_loss = running_loss / len(train_loader)
    train_losses.append(avg_loss)

    if (epoch+1) % 10 == 0:
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")
```

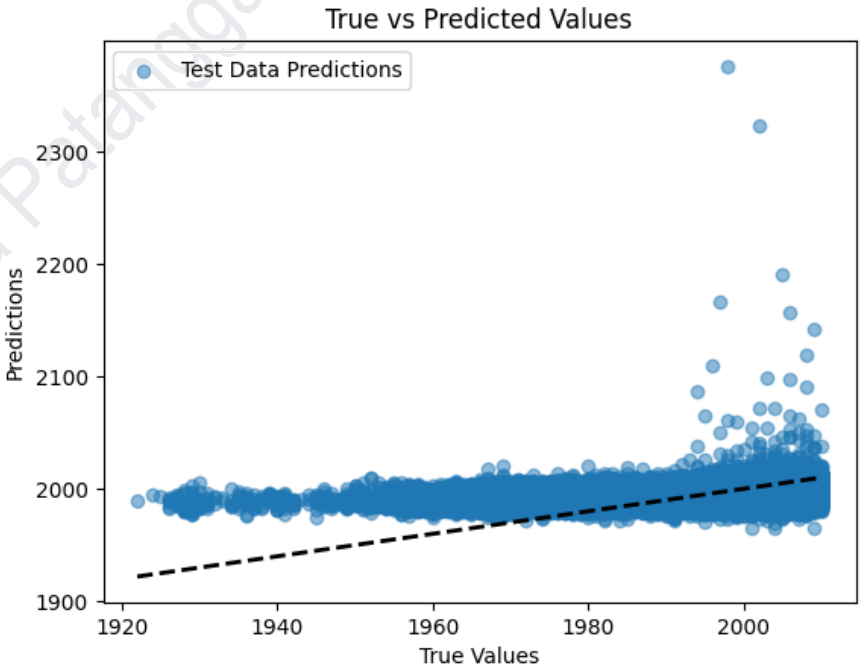
Nama : Yoga Patangga Balapradhana
NIM : 1103194138

Setelah training loop selesai dilakukan, langkah berikutnya adalah melakukan prediksi pada data uji menggunakan perintah:

```
model.eval()  # Set the model to evaluation mode
with torch.no_grad():  # We don't need to track gradients for evaluation
    y_pred_train = model(X_train_tensor).numpy()
    y_pred_test = model(X_test_tensor).numpy()
```

Langkah berikutnya adalah menggambar kurva training loss dan scatter plot untuk memvisualisasikan data aktual vs data prediksi dengan menggunakan `matplotlib`. Hasil kurva training loss dan data aktual vs data prediksi dapat dilihat pada gambar berikut:





Yoga Patangga Balapradhana (1103194138)

Nama : Yoga Patangga Balapradhana
NIM : 1103194138

Langkah terakhir yaitu menghitung matriks evaluasi RMSE, MSE, dan R^2 dengan perintah:

```
rmse = root_mean_squared_error(y_test, y_pred_test)
mse = mean_squared_error(y_test, y_pred_test)
r2 = r2_score(y_test, y_pred_test)
print(f"RMSE: {rmse:.4f}")
print(f"MSE: {mse:.4f}")
print(f"RSquared: {r2:.4f}")
```

```
RMSE: 9.3681
MSE: 87.7609
RSquared: 0.2681
```

MSE atau Mean Squared Error adalah ukuran yang digunakan untuk mengevaluasi seberapa baik sebuah model regresi dalam memprediksi nilai target. MSE mengukur rata-rata kuadrat selisih antara nilai prediksi dan nilai aktual, sekaligus memberikan gambaran tentang seberapa besar kesalahan rata-rata model dalam memprediksi nilai. Nilai MSE yang lebih kecil menunjukkan bahwa model memiliki performa yang lebih baik dan kesalahan prediksi yang lebih kecil.

RMSE atau Root Mean Squared Error adalah akar kuadrat dari MSE, yang juga merupakan ukuran yang sering digunakan untuk menilai performa model regresi. RMSE memberikan ukuran kesalahan dalam unit yang sama dengan data asli, sehingga lebih mudah dipahami dibandingkan MSE. Semakin kecil nilai RMSE maka semakin baik model tersbut dalam memprediksi data.

R^2 atau disebut juga Koefisien Determinasi adalah ukuran statistik yang digunakan untuk menilai sejauh mana model regresi menjelaskan variabilitas dalam data. Dengan kata lain, R^2 mengukur seberapa baik model memprediksi nilai target dengan membandingkan variabilitas yang dapat dijelaskan oleh model terhadap variabilitas total data. Nilai R^2 yang lebih tinggi menunjukkan bahwa model lebih baik dalam menjelaskan variasi dalam data.

Nilai RMSE dan MSE yang dihasilkan pada model MLP yang digunakan dalam Regression Task ini yaitu sebesar 9,37 dan 87,76 sedangkan nilai R^2 yang dihasilkan adalah sebesar 0,27. Nilai matriks evaluasi ini menunjukkan performa yang cukup baik pada model MLP yang digunakan.

Yoga Patangga Balapradhana (1103194138)