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:

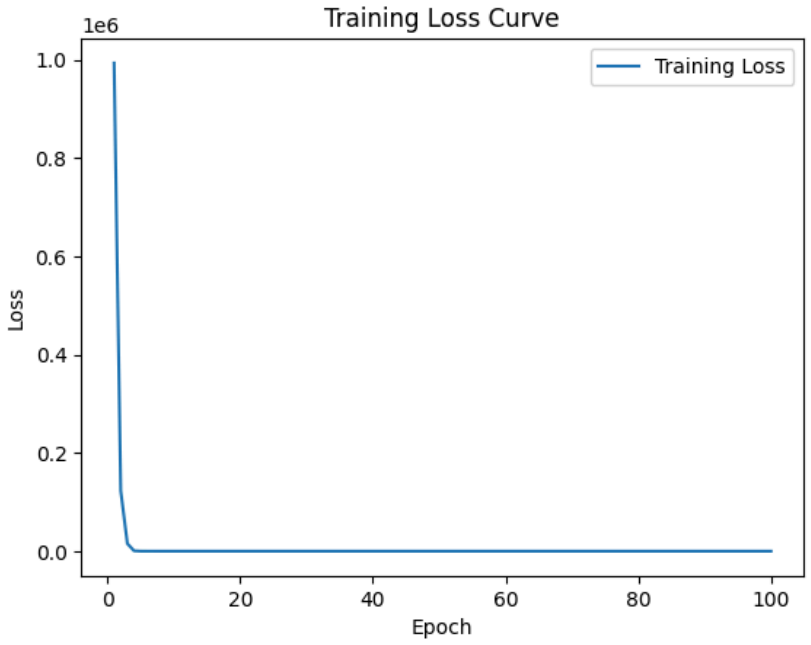
```
num_epochs = 100  # Number of epochs for training

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}")
```

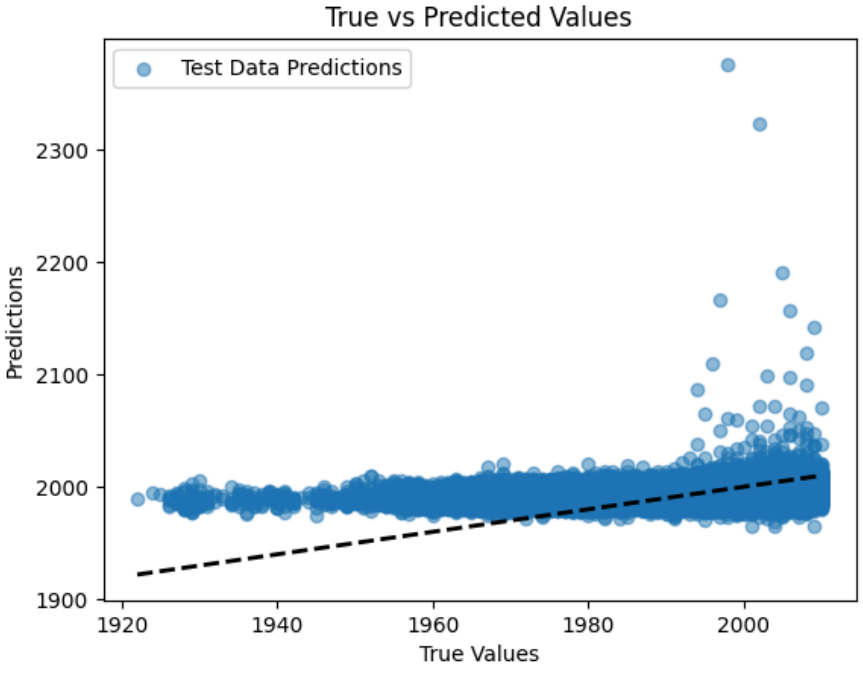
Nama : Yoga Patangga Balapradhana
NIM : 1103194138

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:





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.

Soal Analisis

1. **Jika menggunakan model MLP dengan 3 hidden layer (256-128-64) menghasilkan underfitting pada dataset ini, modifikasi apa yang akan dilakukan pada arsitektur? Jelaskan alasan setiap perubahan dengan mempertimbangkan bias-variance tradeoff.**

Penjelasan: Underfitting terjadi ketika model terlalu sederhana sehingga tidak dapat menangkap pola yang ada dalam data. Jika model MLP mengalami underfitting pada dataset, maka model tersebut tidak cukup kompleks untuk menangkap pola yang ada dalam

data. Dalam hal ini, kita dapat memodifikasi arsitektur model untuk meningkatkan kemampuannya dalam mempelajari pola-pola yang lebih kompleks tanpa menambah terlalu banyak variance yang bisa menyebabkan overfitting. Berikut adalah beberapa cara yang bisa dilakukan.

Cara pertama adalah dengan menambah jumlah hidden layer. Dengan menambah layer, model dapat menangkap hubungan yang lebih kompleks dan lebih dalam di dalam data. Selain itu, penambahan layer akan mengurangi bias model karena model menjadi lebih fleksibel. Namun terlalu banyak layer bisa menyebabkan overfitting jika model belajar terlalu banyak detail yang tidak relevan atau noise dalam data.

Berikutnya adalah dengan menambah jumlah neuron di setiap layer. Underfitting juga bisa disebabkan jumlah neuron di setiap hidden layer terlalu sedikit untuk menangkap kompleksitas data. Oleh karena itu, menambah jumlah neuron di setiap hidden layer bisa membantu model lebih baik dalam menangkap pola dalam data. Jika sebelumnya jumlah hidden layer adalah 256-128-64, maka bisa ditambahkan menjadi 512-256-128.

Selain itu, cara lainnya adalah dengan mengganti fungsi aktivasi. Pada MLP, fungsi aktivasi yang digunakan untuk neuron di setiap hidden layer juga berperan penting. Jika model yang digunakan saat ini menggunakan fungsi aktivasi ReLU, bisa diganti dengan fungsi aktivasi lain seperti sigmoid atau tanh.

2. **Selain MSE, loss function apa yang mungkin cocok untuk dataset ini? Bandingkan kelebihan dan kekurangannya, serta situasi spesifik di mana alternatif tersebut lebih unggul daripada MSE.**

Penjelasan: Selain MSE, MAE atau Mean Absolute Error juga dapat digunakan sebagai loss function untuk dataset ini. MAE mengukur rata-rata dari nilai mutlak selisih antara nilai prediksi dan nilai aktual. Ini adalah salah satu loss function yang paling sederhana dan sering digunakan untuk masalah regresi. MAE memiliki keunggulan yaitu tidak terlalu sensitif terhadap outlier dikarenakan MAE menggunakan nilai absolut, yang berarti tidak ada pembesaran kesalahan seperti pada MSE. Namun, MAE memiliki kelemahan yaitu kurang stabil saat optimasi karena MAE menghasilkan gradien yang konstan untuk kesalahan prediksi yang dapat menyebabkan kesulitan dalam beberapa algoritma optimasi. Hal ini dapat menyebabkan model lebih sulit untuk mencapai nilai yang konvergen. Dengan keunggulan dan kelemahan ini, MAE lebih cocok digunakan untuk data dengan banyak nilai outlier, serta data dengan distribusi target yang tidak simetris atau tidak memiliki pola Gaussian.

Selain itu, Huber Loss juga dapat digunakan karena fungsi ini menggabungkan keuntungan dari MSE dan MAE. Huber Loss memberikan penalti kuadrat untuk kesalahan kecil (seperti MSE) dan penalti absolut untuk kesalahan besar (seperti MAE). Dengan kata lain, Huber Loss dapat menghindari penalti yang besar terhadap outlier seperti MSE, tetapi tetap sensitif terhadap kesalahan kecil seperti MAE.

3. **Jika salah satu fitur memiliki range nilai 0-1, sedangkan fitur lain 100-1000, bagaimana ini memengaruhi pelatihan MLP? Jelaskan mekanisme matematis (e.g., gradien, weight update) yang terdampak.**

Nama : Yoga Patangga Balapradhana
NIM : 1103194138

Penjelasan: Ketika menggunakan MLP untuk pelatihan model dan fitur yang digunakan memiliki rentang nilai yang sangat berbeda, hal ini dapat mempengaruhi proses pelatihan model secara signifikan. Algoritma pembelajaran berbasis gradient descent yang digunakan dalam pelatihan MLP mengandalkan perhitungan gradien untuk memperbarui bobot. Jika fitur-fitur memiliki skala yang sangat berbeda, maka nilai gradien yang dihitung untuk setiap fitur juga akan memiliki skala yang berbeda dan akan mempengaruhi update bobot dan proses konvergensi model.

Fitur dengan skala kecil (misalnya 0-1) memiliki perhitungan gradien yang cenderung lebih kecil, sehingga pembaruan bobot terkait fitur ini akan lebih lambat dan model akan lebih sulit untuk belajar secara efisien dari fitur tersebut. Sedangkan fitur dengan skala besar (misalnya 100-1000) memiliki perhitungan gradien yang cenderung lebih besar pula, sehingga pembaruan bobot akan lebih cepat. Jika fitur ini sangat dominan dalam pembaruan bobot, maka dapat menyebabkan model lebih sensitif terhadap fitur tersebut dan mengabaikan fitur lain dengan skala lebih kecil.

4. **Tanpa mengetahui nama fitur, bagaimana Anda mengukur kontribusi relatif setiap fitur terhadap prediksi model? Jelaskan metode teknikal (e.g., permutation importance, weight analysis) dan keterbatasannya.**

Penjelasan: Untuk kontribusi relatif setiap fitur terhadap prediksi model tanpa mengetahui nama fitur, kita dapat menggunakan berbagai metode yang dapat menilai kontribusi setiap fitur terhadap performa model. Beberapa metode yang umum digunakan untuk menilai pentingnya fitur dalam model antara lain: Koefisien Regresi, L1 Regularization (Lasso), Feature Importance dengan Decision Tree dan Ensemble Methods, Permutation Feature Importance, dan SHAP (SHapley Additive exPlanations).

Koefisien regresi dapat digunakan untuk mengukur pentingnya fitur, yaitu nilai koefisien yang lebih besar menunjukkan pengaruh yang lebih besar dari fitur tersebut terhadap prediksi. Namun, metode ini hanya dapat digunakan untuk model regresi linier atau model yang dapat diinterpretasikan dalam bentuk koefisien linier.

L1 regularization (Lasso) dapat digunakan pada regresi linier untuk melakukan seleksi fitur dengan cara menambahkan penalti terhadap jumlah nilai absolut dari koefisien regresi model. Dengan menambahkan penalti terhadap ukuran koefisien, Lasso mendorong koefisien untuk mendekati nol, dan beberapa koefisien dapat menjadi nol sepenuhnya, sehingga fitur yang tidak penting secara efektif dihapus. Lasso memiliki keterbatasan yaitu hanya dapat digunakan untuk model regresi linier, sehingga tidak cocok untuk model berbasis pohon atau non-linier.

Model Decision Tree dan algoritma Ensemble seperti Random Forest atau Gradient Boosting juga dapat mengukur pentingnya fitur berdasarkan pengurangan impurity. Pada prinsipnya, setiap keputusan dalam Decision Tree didasarkan pada pemilihan fitur terbaik yang memisahkan data dengan cara yang mengurangi ketidakmurnian (impurity). Impurity yang sering digunakan termasuk Gini impurity atau Entropy. Namun, model ini memiliki keterbatasan yaitu tidak dapat digunakan untuk untuk model linier, karena ia hanya relevan untuk model berbasis pohon atau non-linier.

Nama : Yoga Patangga Balapradhana
NIM : 1103194138

Permutation Feature Importance adalah metode model-agnostik yang mengukur seberapa pentingnya suatu fitur dengan mengacak (permutasi) nilai fitur tersebut dan menghitung penurunan dalam performa model. Jika permutasi suatu fitur menyebabkan penurunan performa yang signifikan, itu berarti fitur tersebut sangat penting. Sebaliknya, jika performa tidak banyak berubah setelah permutasi, fitur tersebut dianggap kurang penting. Metode ini memiliki keterbatasan yaitu memerlukan model yang sudah dilatih sebelumnya sehingga memakan waktu lebih lama tergantung pada ukuran dataset dan kompleksitas modelnya.

SHAP adalah metode yang didasarkan pada teori permainan dan memberikan penilaian yang lebih adil mengenai kontribusi setiap fitur dalam prediksi model. SHAP menghitung kontribusi marginal dari setiap fitur terhadap prediksi dengan cara membandingkan prediksi model saat fitur tersebut ada atau tidak ada dalam model. Metode ini memiliki keterbatasan yaitu bisa sangat mahal secara komputasi, terutama untuk model yang besar atau dataset yang sangat besar.

5. **Bagaimana Anda mendesain eksperimen untuk memilih learning rate dan batch size secara optimal? Sertakan analisis tradeoff antara komputasi dan stabilitas pelatihan.**

Penjelasan: Desain eksperimen untuk memilih memilih learning rate dan batch size secara optimal dapat dilakukan dengan menggunakan beberapa teknik pencarian seperti Grid Search, Random Search, atau Bayesian Optimization.

Metode Grid Search mencoba semua kombinasi dari learning rate dan batch size yang telah ditentukan dalam grid pencarian, sehingga metode ini adalah pendekatan yang sangat sistematis dan menyeluruh. Metode ini dapat memberikan solusi yang sangat komprehensif untuk menemukan kombinasi learning rate dan batch size yang optimal dan cocok untuk ruang pencarian yang terbatas. Namun, jika ada banyak kombinasi learning rate dan batch size, pencarian grid dapat menjadi sangat lambat dan dapat terjadi overfitting pada set parameter tertentu.

Random Search memilih kombinasi learning rate dan batch size secara acak dari ruang pencarian yang telah ditentukan, sehingga lebih efisien dibandingkan grid search terutama jika ada banyak parameter atau ruang pencarian yang besar. Namun, metode ini bisa jadi tidak memberikan solusi yang optimal karena tidak mencoba semua kombinasi.

Bayesian Optimization adalah metode yang lebih canggih untuk memilih learning rate dan batch size dengan memanfaatkan probabilitas dan fungsi akuisisi. Metode ini sangat efisien dalam mencari learning rate dan batch size optimal dengan sedikit percobaan sehingga cocok untuk ruang pencarian yang besar. Namun, pada praktiknya implementasinya lebih kompleks karena memerlukan perangkat tambahan dan tidak selalu tersedia secara langsung dalam semua library machine learning.