Rizqy Asyraff Athallah

1103210158

Week 14

Markov model dan Hidden Markov Model

1. **Membaca dan Pra-Pemrosesan Dataset**

```
data.columns = data.columns.str.strip()
data = data.select_dtypes(include=[np.number]).dropna()
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

- **Menghapus kolom non-numerik**: Hanya kolom numerik yang digunakan untuk eksperimen.
- **Normalisasi**: StandardScaler digunakan untuk menyamakan skala fitur, yang penting untuk stabilitas pelatihan.

2. **Membagi Dataset**

```
X = data_scaled[:, :-1]
y = data_scaled[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

- **X**: Berisi fitur.
- **y**: Berisi target yang akan diprediksi.
- **Test set 20%**: Memastikan ada cukup data untuk evaluasi.

3. **Membuat DataLoader**

```
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)

# Create TensorDataset
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)

# Create DataLoader
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

- **Batching**: Membagi data ke dalam batch (ukuran 32).
- **Shuffling**: Melakukan pengacakan data pelatihan agar model tidak belajar dari urutan tertentu.

4. **Markov Model**

```
def train_markov(data):
    transition_counts = Counter((a, b) for a, b in zip(data[:-1],
data[1:]))
    total_counts = Counter(data[:-1])
    transition_probs = {k: v / total_counts[k[0]] for k, v in
transition_counts.items()}
    return transition_probs
```

- **Frekuensi Transisi**: Menghitung berapa kali setiap transisi terjadi dalam data.
- **Probabilitas Transisi**: Menghitung peluang transisi dari status ke status.

5. **Hidden Markov Model**

```
# Hidden Markov Model (HMM)
def train_hmm(X, n_components=2):
    model = hmm.GaussianHMM(n_components=n_components,
covariance_type="diag", n_iter=100)
    model.fit(X)
    return model
```

- **n_components=2**: Model HMM dengan dua state tersembunyi.
- **Covariance Diagonal**: Asumsi bahwa setiap fitur bersifat independen.

6. **Definisi RNN**

```
# RNN Model
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size,
pooling="max"):
        super(RNNModel, self).__init__()
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
        self.pooling = pooling
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        rnn_out, _ = self.rnn(x)
        if self.pooling == "max":
            x = torch.max(rnn_out, dim=1)[0]
        else:
            x = torch.mean(rnn_out, dim=1)
        return self.fc(x)
```

- **Pooling**: Menangkap informasi penting dari urutan output.
- **Arsitektur Sederhana**: Menggunakan satu layer RNN dan satu lapisan fully connected.

7. **Experiment RNN**

```
# RNN Experiments
for hidden_size in hidden_sizes:
    for pooling in pooling_types:
        for optimizer_name in optimizers:
            for epochs in epochs_list:
```

```
                model = train_rnn(hidden_size, pooling, optimizer_name,
epochs)
                model.eval()
                with torch.no_grad():
                    predictions = []
                    for batch_X, _ in test_loader:
                        pred = model(batch_X.unsqueeze(1))
                        predictions.extend(pred.squeeze().tolist())
                results.append({
                    "hidden_size": hidden_size,
                    "pooling": pooling,
                    "optimizer": optimizer_name,
                    "epochs": epochs,
                    "mse_loss": nn.MSELoss()(torch.tensor(predictions),
y_test_tensor).item()
                })
```

- **Hidden Size**: Mengontrol kapasitas jaringan.
- **Pooling**: Menentukan metode agregasi informasi.
- **Optimizer**: Mencoba SGD, RMSProp, dan Adam.
- **Epoch**: Menguji performa pada jumlah epoch berbeda.

8. Evaluasi dan Penyimpanan Hasil

```
results_df = pd.DataFrame(results)
results_df.to_csv("rnn_results.csv", index=False)
```

- CSV ini dapat digunakan untuk membuat laporan atau visualisasi performa model.