加载数据

In [ ]:
```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

train_df = pd.read_csv('/home/zhenghao/kaggle/train.csv')
test_df = pd.read_csv('/home/zhenghao/kaggle/test.csv')
```

随便选了三个features

In [ ]:
```
label_encoders = {}
categorical_features = ['Gender', 'Vehicle_Age', 'Vehicle_Damage']

for col in categorical_features:
    le = LabelEncoder()
    train_df[col] = le.fit_transform(train_df[col])
    test_df[col] = le.transform(test_df[col])
    label_encoders[col] = le
```

In [ ]:
```
X_train = train_df.drop(['id', 'Response'], axis=1)
y_train = train_df['Response']

X_test = test_df.drop(['id'], axis=1)
```

In [ ]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,

# 转换为tensor
X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
X_val = torch.tensor(X_val, dtype=torch.float32)
y_val = torch.tensor(y_val.values, dtype=torch.float32).view(-1, 1)
X_test = torch.tensor(X_test, dtype=torch.float32)
```

然后套了一个最弱智的模型

In [ ]:
```
class SimpleNN(nn.Module):
    def __init__(self, input_dim):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 16)
        self.fc2 = nn.Linear(16, 8)
        self.fc3 = nn.Linear(8, 1)
```

```
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.sigmoid(self.fc3(x))
        return x

input_dim = X_train.shape[1]
model = SimpleNN(input_dim)
```

In [ ]:
```
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

num_epochs = 100

for epoch in range(num_epochs):
    model.train()
    optimizer.zero_grad()
    outputs = model(X_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    optimizer.step()

    if (epoch+1) % 10 == 0:
        model.eval()
        with torch.no_grad():
            val_outputs = model(X_val)
            val_loss = criterion(val_outputs, y_val)
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}, Val Loss:
```

```
/home/zhenghao/.conda/envs/yolo/lib/python3.12/site-packages/torch/cuda/__init__.py:
619: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
Epoch [10/100], Loss: 0.7067, Val Loss: 0.7039
Epoch [20/100], Loss: 0.6782, Val Loss: 0.6752
Epoch [30/100], Loss: 0.6475, Val Loss: 0.6442
Epoch [40/100], Loss: 0.6142, Val Loss: 0.6106
Epoch [50/100], Loss: 0.5783, Val Loss: 0.5745
Epoch [60/100], Loss: 0.5402, Val Loss: 0.5363
Epoch [70/100], Loss: 0.5013, Val Loss: 0.4973
Epoch [80/100], Loss: 0.4630, Val Loss: 0.4592
Epoch [90/100], Loss: 0.4269, Val Loss: 0.4234
Epoch [100/100], Loss: 0.3949, Val Loss: 0.3920
```

看看蒙对了多少

In [ ]:
```
model.eval()
with torch.no_grad():
    val_outputs = model(X_val)
    val_outputs = val_outputs.round()
```

```
    accuracy = (val_outputs.eq(y_val).sum() / float(y_val.shape[0])).item()
    print(f'Validation Accuracy: {accuracy:.4f}')
```

```
Validation Accuracy: 0.8769
```

保存csv

In [ ]:

```
model.eval()
with torch.no_grad():
    predictions = model(X_test)
    predictions = predictions.round().numpy()

test_df['Response'] = predictions
submission_df = test_df[['id', 'Response']]
submission_df.to_csv('submission.csv', index=False)
```