LAB 6 TASK 1

Logistic Regression

In [12]:

```
import pandas as pd
import torch.nn as nn
import torch
import numpy as np
import io
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [14]:

```
from google.colab import drive
data=pd.read_csv('/content/drive/My Drive/BuyComputer.csv')
data.drop(columns=['User ID',],axis=1,inplace=True)
data.head()
```

Out[14]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [15]:

```
#Label as last column
y = data.iloc[:,-1].values
```

In [16]:

```
#X as all columns excluding last
X = data.iloc[:,:-1].values
```

In [17]:

```
samples,inputs = X.shape
```

In [18]:

```
# Splitting data
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(X, y, test_size = 0.30,
```

In [19]:

```
sc = StandardScaler()
train_data = sc.fit_transform(train_data)
test_data = sc.transform(test_data)
```

In [20]:

```
train_data = torch.from_numpy(train_data.astype(np.float32))
test_data = torch.from_numpy(test_data.astype(np.float32))
train_target = torch.from_numpy(train_target.astype(np.float32))
test_target = torch.from_numpy(test_target.astype(np.float32))
```

In [22]:

```
train_target = train_target.view(train_target.shape[0], 1)
test_target = test_target.view(test_target.shape[0], 1)
```

In [23]:

```
#Model
class LogisticRegressionModel(nn.Module):
    def __init__(self, n_input_features):
        super(LogisticRegressionModel, self).__init__()
        self.linear = nn.Linear(n_input_features, 1)

    def forward(self, x):
        target_pred = torch.sigmoid(self.linear(x))
        return target_pred
```

In [24]:

```
model = LogisticRegressionModel(inputs)
```

In [30]:

```
num_epochs = 10000
learning_rate = 0.02
criterion = nn.BCELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

In [31]:

```python
for epoch in range(num_epochs):
    target_pred = model(train_data)
    loss = criterion(target_pred, train_target)

    loss.backward()
    optimizer.step()

    # zero grad before new step
    optimizer.zero_grad()

    if (epoch+1) % 50 == 0:
        print(f'epoch: {epoch+1}, loss = {loss.item():.4f}')


with torch.no_grad():
    target_predicted = model(test_data)
    target_predicted_cls = target_predicted.round()
    acc = target_predicted_cls.eq(test_target).sum() / float(test_target.shape[0])
    print(f'accuracy: {acc.item():.4f}')
```

```
epoch: 50, loss = 0.3283
epoch: 100, loss = 0.3283
epoch: 150, loss = 0.3283
epoch: 200, loss = 0.3283
epoch: 250, loss = 0.3283
epoch: 300, loss = 0.3283
epoch: 350, loss = 0.3283
epoch: 400, loss = 0.3283
epoch: 450, loss = 0.3283
epoch: 500, loss = 0.3283
epoch: 550, loss = 0.3283
epoch: 600, loss = 0.3283
epoch: 650, loss = 0.3283
epoch: 700, loss = 0.3283
epoch: 750, loss = 0.3283
epoch: 800, loss = 0.3283
epoch: 850, loss = 0.3283
epoch: 900, loss = 0.3283
epoch: 950, loss = 0.3283
```