Practical 6

Aim: To implement Neural Network.

Code:

*import* pandas *as* pd

*from* sklearn.preprocessing *import* StandardScaler

*from* sklearn.neural\_network *import* MLPClassifier

*from* sklearn.preprocessing *import* LabelEncoder

*from* sklearn.feature\_extraction *import* DictVectorizer

data=pd.read\_csv('data.csv')

cols\_to\_retain=['Alt','Bar','Fri','Hun','Pat','Price','Rain','Res','Type','Est']

X\_feature=data[cols\_to\_retain]

X\_dict=X\_feature.T.to\_dict().values()

vect=DictVectorizer(sparse=False)

X\_vector=vect.fit\_transform(X\_dict)

print(X\_vector)

X\_Train=X\_vector[:-1]

X\_Test=X\_vector[-1:]

print('Train Set')

print(X\_Train)

print('test set')

print(X\_Test)

le=LabelEncoder()

y\_Train=le.fit\_transform(data['Goal'][:-1])

scaler=StandardScaler()

scaler.fit(X\_Train)

X\_Train=scaler.transform(X\_Train)

X\_Test=scaler.transform(X\_Test)

mlp=MLPClassifier(hidden\_layer\_sizes=(10,10,10),max\_iter=1000)

mlp.fit(X\_Train,y\_Train)

print(le.inverse\_transform(mlp.predict(X\_Test)))

print("Weight between input and first hidden layer:")

print(mlp.coefs\_[0])

print("\nWeights between first hidden and second hidden layer:")

print(mlp.coefs\_[1])

Output:





