In [4]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import warnings

warnings.filterwarnings('ignore')
```

In [5]:

```
# Perceptron Class
class Perceptron:
    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        self.eta = eta
        self.n_iter = n_iter
        self.random_state = random_state
        self.w_ = None
        self.errors_ = None

    def fit(self, X, y):
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[1:] += update * xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, 0)
```

In [6]:

```
# Main Execution
if __name__ == "__main__":

    # Perceptron Demo
    X_simple = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
    y_simple = np.array([0, 0, 0, 1])

    scaler_simple = StandardScaler()
    X_simple_std = scaler_simple.fit_transform(X_simple)

    ppn = Perceptron(eta=0.1, n_iter=20)
    ppn.fit(X_simple_std, y_simple)

    # MLP on Churn
    filepath = 'LabAssig5_stuff/churn.csv'

    try:
        # Load Data
        df = pd.read_csv(filepath)

        # Preprocessing
        y = df['Churn'].astype(bool).astype(int)
        X = df.drop('Churn', axis=1)

        categorical_cols = ['State', 'Area code', 'International plan', 'Voice mail plan']
```

```python
        numerical_cols = [col for col in X.columns if col not in categorical_cols]

        # Train/Test Split
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=42, stratify=y
        )

        # Build Pipeline
        numeric_transformer = Pipeline(steps=[
            ('scaler', StandardScaler())
        ])
        categorical_transformer = Pipeline(steps=[
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ])
        preprocessor = ColumnTransformer(
            transformers=[
                ('num', numeric_transformer, numerical_cols),
                ('cat', categorical_transformer, categorical_cols)
            ])

        mlp = MLPClassifier(
            hidden_layer_sizes=(100, 50),
            max_iter=1000,
            alpha=0.001,
            random_state=42,
            early_stopping=True,
            n_iter_no_change=10
        )

        # Create the final full pipeline
        # We add SMOTE() to oversample the minority class (Churn=1)
        # This step is applied ONLY to the training data during .fit()
        clf_pipeline = Pipeline(steps=[
            ('preprocessor', preprocessor),
            ('smote', SMOTE(random_state=42)),
            ('classifier', mlp)
        ])

        # Train Model
        clf_pipeline.fit(X_train, y_train)

        # Evaluate Model
        y_pred = clf_pipeline.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        print(f"\nModel Accuracy: {accuracy:.4f} ({accuracy*100:.2f}%)")

        print("\nConfusion Matrix:")
        cm = confusion_matrix(y_test, y_pred)
        print(cm)

        print("\nClassification Report:")
        report = classification_report(
            y_test,
            y_pred,
            target_names=['Not Churn (0)', 'Churn (1)']
        )
        print(report)

    except FileNotFoundError:
        print(f"Error: The file '{filepath}' was not found.")
        print("Please make sure the file is in the correct directory.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        import traceback
        traceback.print_exc()
```

```
Model Accuracy: 0.8657 (86.57%)

Confusion Matrix:
[[156  16]
 [ 11  18]]

Classification Report:
               precision    recall  f1-score   support

Not Churn (0)       0.93      0.91      0.92       172
    Churn (1)       0.53      0.62      0.57        29

     accuracy                           0.87       201
    macro avg       0.73      0.76      0.75       201
 weighted avg       0.88      0.87      0.87       201
```