Simple Loan Classification

Anthony Argueta and Gabriel Mendoza

Problems at Hand:

- Our goal was to predict whether a person will get their loan approved via the use of machine learning algorithms KNN, Logistic Regression, Neural Network, Random Forest, and XGBoost.
- The implementation of the Neural Network, KNN went very smoothly
- Unnecessary Features (More categorical than number)
- On the other hand the other 3 gave us the same problem.
- Accuracy Too Perfect: Logistic Regression,Random Forest, and XGBoost all showed 100% accuracy—felt unrealistic and likely caused by overfitting.
- The target could’ve been to imbalanced in weight (ie 90% Acceptance and 10% Decline)
- Data Leakage was the main concern.

Solutions:

- For the algorithms that we were having trouble like KNN, XGBoost, and Random Forest.
- Used StandardScaler: Ensured that models like KNN performed well by scaling features
- Added Comments and Results Tracking: Cleaned up code with clear outputs and accuracy reporting for all models
- Verified Dataset Splits: Used train_test_split(shuffle=True) to make sure data wasn't being leaked or reused across models
- Explained Metrics: Added classification reports and interpreted them to explain what the models were doing right or wrong.
- For XGBoost we tried to add more metrics like (max depths and mid_child_weights)

Results (so far):

Out of all the algorithms:

KNN - 92.3 %

Logistic Regression- 1.0 or 100%

Neural Network- 84.62%

XGBoost- 1.0 or 100%

RandomForest - 1.0 or 100%

Results

```
Epoch [10/100], Loss: 0.4551
Epoch [20/100], Loss: 0.4188
Epoch [30/100], Loss: 0.3836
Epoch [40/100], Loss: 0.3487
Epoch [50/100], Loss: 0.3157
Epoch [60/100], Loss: 0.2854
Epoch [70/100], Loss: 0.2583
Epoch [80/100], Loss: 0.2348
Epoch [90/100], Loss: 0.2146
Epoch [100/100], Loss: 0.1976
Test Accuracy: 0.8462
```

```
Random Forest Train Accuracy = 1.0
Random Forest Test Accuracy = 1.0

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         9

    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13
```

```
...  Logistic Regression Train Accuracy = 1.0
     Logistic Regression Test Accuracy = 1.0

     Classification Report:

                   precision    recall  f1-score   support

                0       1.00      1.00      1.00         4
                1       1.00      1.00      1.00         9

         accuracy                           1.00        13
        macro avg       1.00      1.00      1.00        13
     weighted avg       1.00      1.00      1.00        13
```

```
# XGBoost!

from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score


model = XGBClassifier(eval_metric='logloss', max_depth=3, min_child_weight=1)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", accuracy)
```

```
[93]  ✓ 0.0s
...  Test Accuracy: 1.0
```

```
KNN Train Accuracy= 0.9583333333333334
Knn Test accuracy 0.9230769230769231
```