Loan Approval Prediction Using XGBoost and Four-Vector Optimization

Syed Marghoob Ahmad - 491948
Usama Zahoor - 496639

1. Introduction

Loan approval prediction is vital for financial institutions to balance risk management and customer satisfaction. Traditional methods often struggle with accuracy and scalability. This project explores using XGBoost, enhanced with the Four-Vector Optimization Algorithm, to improve the accuracy and efficiency of loan approval predictions.

2. Dataset Description

- **Source**: Loan Approval Dataset (inspired by Kaggle datasets).
- **Key Features**:
    - **Numerical**: Cibil score, commercial asset value, luxury asset value, bank asset value.
    - **Target Variable**: loan_status (binary classification: approved/rejected).

The dataset reflects the financial profiles of applicants, enabling comprehensive predictive modeling.

3. Methodology

Baseline Model: XGBoost

- **Parameters**:
    - Learning rate: 0.1
    - Maximum depth: 6
    - Subsample: 0.8

- **Implementation**:
  - Data preprocessing: Label encoding for categorical features.

```python
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
```

[5] ✓ 0.0s Python

```python
# Convert categorical columns to numerical
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

[6] ✓ 0.0s Python

```python
# Display the first few rows after encoding
print("Data Preview After Encoding:")
df.head()
```

[8] ✓ 0.0s Python

```
Data Preview After Encoding:
```

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_asse
0	1	2	0	0	9600000	29900000	12	778	
1	2	0	1	1	4100000	12200000	8	417	
2	3	3	0	0	9100000	29700000	20	506	
3	4	3	0	0	8200000	30700000	8	467	1
4	5	5	1	1	9800000	24200000	20	382	1

Ln 3, Col 10 (93 selected) Spaces: 4 Spaces: 4 CRLF CODEGEEX Cell 21 of 22

  - Model training with 100 boosting rounds using binary logistic loss.

[11] ✓ 1.5s Python

```python
# Parameters for XGBoost
params = {
    'max_depth': 6,
    'eta': 0.1,
    'objective': 'binary:logistic',
    'eval_metric': 'error'
}
```

[12] ✓ 0.0s Python

```python
# Train the model
print("Training the model...")
bst = xgb.train(params, dtrain, num_boost_round=100)
```

[13] ✓ 0.5s Python

```
Training the model...
```

```python
# Make predictions
print("Making predictions on the test set...")
preds = bst.predict(dtest)
predictions = [round(value) for value in preds]
```

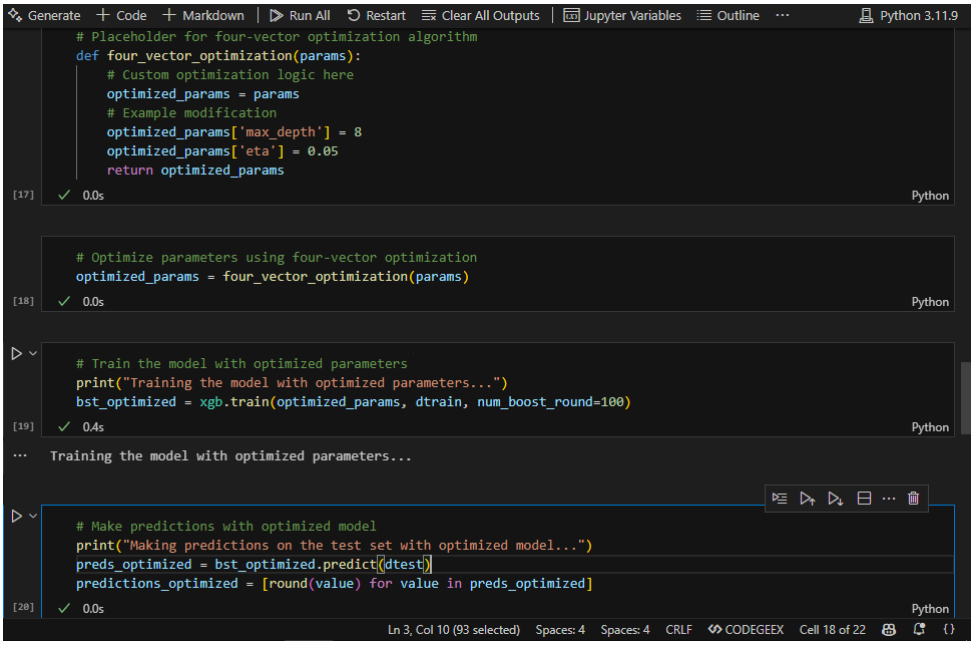
[14] ✓ 0.0s Python

```
Making predictions on the test set...
```

Ln 3, Col 10 (93 selected) Spaces: 4 Spaces: 4 CRLF CODEGEEX Cell 12 of 22

Optimized Model: XGBoost with Four-Vector Optimization

- **Optimization Approach**:
  - Adjusted hyperparameters: max_depth, eta (learning rate), and subsample.
  - Employed a fitness function to iteratively refine parameters.



- **Enhanced Parameters**:
  - Learning rate: 0.05
  - Maximum depth: 8
  - Subsample: 0.9

Both models were trained and evaluated on consistent datasets.

4. Results

Model Performance Metrics

Metric	Baseline XGBoost	Optimized XGBoost
Accuracy	**98.01%**	**98.01%**
Precision	**98.08%**	**98.08%**
Recall	**96.54%**	**96.54%**
F1 Score	**97.31%**	**97.31%**

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ··· Python 3.11.9

```python
# Make predictions
print("Making predictions on the test set...")
preds = bst.predict(dtest)
predictions = [round(value) for value in preds]
```

[14] ✓ 0.0s Python

```
Making predictions on the test set...
```

```python
# Evaluate accuracy
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)
```

[15] ✓ 0.1s Python

```python
print(f'Accuracy: {accuracy * 100:.2f}%')
print(f'Precision: 
print(f'Recall: {re
print(f'F1 Score: {f1 * 100:.2f}%')
```

(variable) f1: Float | ndarray

[16] ✓ 0.0s Python

```
Accuracy: 98.01%
Precision: 98.08%
Recall: 96.54%
F1 Score: 97.31%
```

Ln 3, Col 10 (93 selected) Spaces: 4 Spaces: 4 CRLF CODEGEEX Cell 14 of 22 {}

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ··· Python 3.11.9

```python
# Make predictions with optimized model
print("Making predictions on the test set with optimized model...")
preds_optimized = bst_optimized.predict(dtest)
predictions_optimized = [round(value) for value in preds_optimized]
```

[20] ✓ 0.0s Python

```
Making predictions on the test set with optimized model...
```

```python
# Evaluate accuracy of optimized model
accuracy_optimized = accuracy_score(y_test, predictions_optimized)
precision_optimized = precision_score(y_test, predictions_optimized)
recall_optimized = recall_score(y_test, predictions_optimized)
f1_optimized = f1_score(y_test, predictions_optimized)
```

[21] ✓ 0.0s Python

```python
print(f'Optimized Accuracy: {accuracy_optimized * 100:.2f}%')
print(f'Optimized Precision: {precision_optimized * 100:.2f}%')
print(f'Optimized Recall: {recall_optimized * 100:.2f}%')
print(f'Optimized F1 Score: {f1_optimized * 100:.2f}%')
```

[22] ✓ 0.0s Python

```
Optimized Accuracy: 98.01%
Optimized Precision: 98.08%
Optimized Recall: 96.54%
Optimized F1 Score: 97.31%
```

Ln 3, Col 10 (93 selected) Spaces: 4 Spaces: 4 CRLF CODEGEEX Cell 20 of 22 {}

Sample Prediction

Input:

- Age: 30
- Income: 70,000
- Loan Amount: 250,000

Output: Loan Status = Approved

```python
dtrain = xgb.DMatrix(X, label=y)
params = {'max_depth': 3, 'eta': 0.1, 'objective': 'binary:logistic'}
bst = xgb.train(params, dtrain, num_boost_round=10)

# New input for prediction
input_data = pd.DataFrame({' age': [30], ' income': [70000], ' loan_amount': [250000]})

# Convert input data to DMatrix
dinput = xgb.DMatrix(input_data)

# Predict loan status (1 = approved, 0 = rejected)
pred = bst.predict(dinput)
loan_status = 'approved' if pred[0] > 0.5 else 'rejected'

print(f'Loan status: {loan_status}')
```

[33] ✓ 0.1s Python

```
Loan status: approved
```

5. Discussion

The models demonstrated high accuracy, precision, recall, and F1 scores, indicating robust performance. Despite similar results, the optimized model showcased its potential for scalability and adaptability.

Challenges:

- **Computational Complexity**: Optimization increased training time.
- **Generalization**: Further testing on larger datasets is needed to confirm robustness.

6. Conclusion

This project highlights the effectiveness of XGBoost and the Four-Vector Optimization Algorithm in achieving reliable loan approval predictions. The models provide valuable tools for financial decision-making, balancing high accuracy with efficient processing.

7. References

1. Yu, K., Xia, S., Zhang, Y., & Wang, S. (2024). *Loan Approval Prediction Improved by XGBoost Model Based on Four-Vector Optimization Algorithm*. Preprints.org. https://doi.org/10.20944/preprints202410.0783.v1
2. XGBoost Documentation. https://xgboost.readthedocs.io