Alex Spence

DSC 630 Project Milestone 4 Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, precision_score,
recall_score, f1_score, confusion_matrix
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import RFE

# Set random seed
np.random.seed(42)

# Load dataset
data = pd.read_csv('loan_default.csv')

# Step 1: Data Preparation
print("Checking for missing values:")
print(data.isnull().sum())

# Remove outliers
num_cols = ['Age', 'Income', 'LoanAmount', 'CreditScore',
'MonthsEmployed', 'NumCreditLines', 'InterestRate', 'LoanTerm',
'DTIRatio']
for col in num_cols:
    cap = data[col].quantile(0.99)
    data[col] = data[col].clip(upper=cap)

cat_cols = ['Education', 'EmploymentType', 'MaritalStatus',
'HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']
data = data.drop('LoanID', axis=1)

# Encode categorical variables
data = pd.get_dummies(data, columns=cat_cols, drop_first=True)

# Define features and target
X = data.drop('Default', axis=1)
y = data['Default']
```

```
# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Apply SMOTE
smote = SMOTE(k_neighbors=3, random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
print("After SMOTE, training set class distribution:",
np.bincount(y_train_smote))

# Feature selection with RFE
lr = LogisticRegression(random_state=42)
rfe = RFE(lr, n_features_to_select=10)
X_train_rfe = rfe.fit_transform(X_train_smote, y_train_smote)
X_test_rfe = rfe.transform(X_test)
selected_features = X.columns[rfe.support_].tolist()
print("Selected features:", selected_features)

# Step 2: Visualizations
# Sample data for scatterplot
sample_data = data.sample(1000, random_state=42)

# Histogram of default rate by credit score
data['CreditScore_bin'] = pd.cut(data['CreditScore'], bins=[0, 600,
700, 1000], labels=['<600', '600-700', '>700'])
plt.figure(figsize=(8, 6))
sns.barplot(x='CreditScore_bin', y='Default', data=data,
estimator=np.mean)
plt.title('Default Rate by Credit Score')
plt.ylabel('Default Rate')
plt.savefig('credit_score_default.png')
plt.close()

# Scatterplot: income vs. loan amount
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Income', y='LoanAmount', hue='Default',
data=sample_data, alpha=0.5)
plt.title('Income vs. Loan Amount by Default Status')
plt.savefig('income_loan_scatter.png')
plt.close()

# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data[num_cols].corr(), annot=True, cmap='coolwarm',
center=0)
plt.title('Feature Correlation Heatmap')
```

```
plt.savefig('correlation_heatmap.png')
plt.close()

# Step 3: Build and Evaluate Models
def evaluate_model(model, X_train, y_train, X_test, y_test, name,
threshold=0.4):
    model.fit(X_train, y_train)
    y_prob = model.predict_proba(X_test)[:, 1]
    y_pred = (y_prob >= threshold).astype(int)

    auc = roc_auc_score(y_test, y_prob)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    print(f"\n{name} Results (Threshold={threshold}):")
    print(f"AUC-ROC: {auc:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")

    # Confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.savefig(f'{name}_confusion_matrix.png')
    plt.close()

    return auc, precision, recall, f1

# Logistic Regression
lr = LogisticRegression(C=0.1, random_state=42)
lr_scores = evaluate_model(lr, X_train_rfe, y_train_smote, X_test_rfe,
y_test, "Logistic Regression", threshold=0.4)

# Random Forest
rf = RandomForestClassifier(max_depth=10, n_estimators=100,
random_state=42)
rf_scores = evaluate_model(rf, X_train_rfe, y_train_smote, X_test_rfe,
y_test, "Random Forest", threshold=0.4)

# XGBoost with cost-sensitive learning
xgb_params = {
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [6, 8, 10],
    'scale_pos_weight': [3, 4, 5]
}
```

```
xgb = XGBClassifier(random_state=42)
grid_search = GridSearchCV(xgb, xgb_params, cv=5, scoring='recall',
n_jobs=-1)
grid_search.fit(X_train_rfe, y_train_smote)
xgb_best = grid_search.best_estimator_
print("Best XGBoost params:", grid_search.best_params_)

# Apply sample weights for cost-sensitive learning
sample_weights = np.where(y_train_smote == 1, 5, 1)  # Higher weight
for defaults
xgb_best.fit(X_train_rfe, y_train_smote, sample_weight=sample_weights)
xgb_scores = evaluate_model(xgb_best, X_train_rfe, y_train_smote,
X_test_rfe, y_test, "XGBoost", threshold=0.35)

# Feature importance plot for XGBoost
feature_importance = pd.DataFrame({
    'feature': selected_features,
    'importance': xgb_best.feature_importances_
}).sort_values('importance', ascending=False)
plt.figure(figsize=(8, 6))
sns.barplot(x='importance', y='feature', data=feature_importance)
plt.title('XGBoost Feature Importance')
plt.savefig('xgboost_feature_importance.png')
plt.close()

# Save results
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest', 'XGBoost'],
    'AUC-ROC': [lr_scores[0], rf_scores[0], xgb_scores[0]],
    'Precision': [lr_scores[1], rf_scores[1], xgb_scores[1]],
    'Recall': [lr_scores[2], rf_scores[2], xgb_scores[2]],
    'F1-Score': [lr_scores[3], rf_scores[3], xgb_scores[3]]
})
results.to_csv('model_results.csv', index=False)
```

```
Checking for missing values:
LoanID            0
Age               0
Income            0
LoanAmount        0
CreditScore       0
MonthsEmployed    0
NumCreditLines    0
InterestRate      0
LoanTerm          0
DTIRatio          0
Education         0
EmploymentType    0
MaritalStatus     0
HasMortgage       0
```

```
HasDependents     0
LoanPurpose       0
HasCoSigner       0
Default           0
dtype: int64
After SMOTE, training set class distribution: [180524 180524]
Selected features: ['Age', 'Income', 'LoanAmount', 'CreditScore',
'MonthsEmployed', 'InterestRate', 'EmploymentType_Part-time',
'EmploymentType_Unemployed', 'HasDependents_Yes', 'HasCoSigner_Yes']

Logistic Regression Results (Threshold=0.4):
AUC-ROC: 0.75
Precision: 0.18
Recall: 0.80
F1-Score: 0.30

Random Forest Results (Threshold=0.4):
AUC-ROC: 0.73
Precision: 0.19
Recall: 0.75
F1-Score: 0.30
Best XGBoost params: {'learning_rate': 0.05, 'max_depth': 6,
'scale_pos_weight': 5}

XGBoost Results (Threshold=0.35):
AUC-ROC: 0.73
Precision: 0.13
Recall: 0.98
F1-Score: 0.23
```