📘 Bankruptcy Prediction for American Companies

Final Data Science Project Report

🔗 Dataset Source:

Kaggle: Bankruptcy Prediction

🧾 Objective

Build predictive models to classify whether a public company will go bankrupt in the following year based on financial/accounting ratios.

```
# 🔹 Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, RobustScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
```

Data Preprocessing & Balancing

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

# Label Encoding
le = LabelEncoder()
data['status_label'] = le.fit_transform(data['status_label'])

# Drop unnecessary columns
X = data.drop(columns=['status_label', 'company_name', 'year'])
y = data['status_label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# SMOTE Balancing
smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train, y_train)
```

Model Training & Comparison

```
models = {
    'DecisionTree': DecisionTreeClassifier(),
    'RandomForest': RandomForestClassifier(),
    'GradientBoosting': GradientBoostingClassifier(),
    'KNN': KNeighborsClassifier(),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss')
}

param_grid = {
    'DecisionTree': {'model__max_depth': [5, 10, 20]},
    'RandomForest': {'model__n_estimators': [100]},
    'GradientBoosting': {'model__n_estimators': [100]},
    'KNN': {'model__n_neighbors': [5, 7]},
    'XGBoost': {'model__n_estimators': [100]}
}

results = []
best_models = {}

for name in models:
    pipeline = Pipeline([('scaler', RobustScaler()), ('model', models[name])])
    grid = GridSearchCV(pipeline, param_grid[name], cv=3, scoring='f1', n_jobs=-1)
    grid.fit(X_train_bal, y_train_bal)
    best_model = grid.best_estimator_
    best_models[name] = best_model
    y_pred = best_model.predict(X_test)
    y_prob = best_model.predict_proba(X_test)[:, 1]
    results.append({
        'Model': name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1-Score': f1_score(y_test, y_pred),
        'ROC AUC': roc_auc_score(y_test, y_prob)
    })

results_df = pd.DataFrame(results).set_index('Model')
results_df
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [15:15:14] WARNING: /workspace/src/learner.cc
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
DecisionTree	0.814069	0.179932	0.506705	0.265562	0.678165
RandomForest	0.917964	0.402832	0.490421	0.442333	0.857835
GradientBoosting	0.689649	0.122790	0.598659	0.203782	0.713521
KNN	0.804283	0.210302	0.707854	0.324265	0.818558
XGBoost	0.814641	0.204107	0.618774	0.306961	0.811582

🏆 Best Model & Classification Report

```
best_model = best_models['KNN']
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.81      0.89     14693
           1       0.21      0.71      0.32      1044

    accuracy                           0.80     15737
   macro avg       0.59      0.76      0.60     15737
weighted avg       0.92      0.80      0.85     15737
```

```
from IPython.display import display, FileLink
display(FileLink("/content/US Bankruptcy Prediction Model Analysis.pdf"))
```

/content/US Bankruptcy Prediction Model Analysis.pdf

```
Start coding or generate with AI.
```