# Bankruptcy Prevention Code Explanation

## 1. Importing Libraries

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import joblib  
  
- \*\*Pandas (`pd`)\*\*: Used for data manipulation and analysis, especially working with tabular data like DataFrames.  
- \*\*NumPy (`np`)\*\*: Provides support for efficient numerical and array operations.  
- \*\*Matplotlib (`plt`)\*\*: A visualization library for plotting graphs like bar charts, scatter plots, and line graphs.  
- \*\*Seaborn (`sns`)\*\*: A high-level visualization library based on Matplotlib, useful for statistical graphics.  
- \*\*Joblib\*\*: Used to save and load serialized Python objects, such as machine learning models or data scalers.  
  
import warnings  
warnings.filterwarnings('ignore')  
  
- Suppresses warnings in the notebook to make the output cleaner and more readable.

## 2. Loading the Dataset

df = pd.read\_excel('Bankruptcy (2).xlsx')  
  
- Reads an Excel file (`Bankruptcy (2).xlsx`) into a Pandas DataFrame named `df`.  
- The dataset contains risk factors and their corresponding bankruptcy outcomes.  
  
print("Dataset Shape:", df.shape)  
  
- Displays the number of rows and columns in the dataset, helping to understand its size.  
  
print("First 5 rows of the dataset:\n", df.head())  
  
- Prints the first five rows of the dataset to give an overview of its structure and contents.

## 3. Basic Dataset Information

df.info()  
  
- Provides a summary of the dataset, including column names, data types, and non-null value counts.  
  
df.describe()  
  
- Generates descriptive statistics, such as mean, standard deviation, and percentiles, for numerical columns in the dataset.

## 4. Data Cleaning

df.isnull().sum()  
  
- Checks for missing values in each column of the dataset.  
  
df.dropna(inplace=True)  
  
- Removes rows with missing values, ensuring the dataset is complete for further analysis.

## 5. Exploratory Data Analysis (EDA)

### Visualizing the Distribution of Features  
plt.figure(figsize=(10, 6))  
sns.histplot(df['class'], bins=3, kde=False)  
plt.title('Class Distribution')  
plt.xlabel('Class')  
plt.ylabel('Count')  
plt.show()  
  
- Plots the distribution of the target variable `class`, showing the balance between bankruptcy and non-bankruptcy cases.  
  
### Pairplot for Risk Factors  
sns.pairplot(df, hue='class', diag\_kind='kde')  
plt.show()  
  
- Generates pairwise scatter plots for all features, colored by the `class`, and overlays kernel density estimates for diagonal plots.  
  
### Heatmap of Correlations  
plt.figure(figsize=(12, 8))  
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')  
plt.title('Correlation Heatmap')  
plt.show()  
  
- Displays the correlation between numerical features using a heatmap, helping to identify strong relationships.

## 6. Data Preprocessing

### Encoding Target Variable  
df['class'] = df['class'].map({'non-bankruptcy': 0, 'bankruptcy': 1})  
  
- Converts the `class` column into a binary format where:  
 - `non-bankruptcy` = 0  
 - `bankruptcy` = 1  
  
### Splitting Features and Target  
X = df.drop(columns=['class'])  
y = df['class']  
  
- \*\*Features (`X`)\*\*: All columns except `class`.  
- \*\*Target (`y`)\*\*: The `class` column representing bankruptcy status.  
  
### Scaling Features  
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
- \*\*StandardScaler\*\*: Standardizes features by removing the mean and scaling to unit variance, ensuring all features contribute equally.

## 7. Splitting Data

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
- Splits the dataset into training (80%) and testing (20%) subsets, ensuring the model has unseen data for evaluation.

## 8. Building Machine Learning Models

### Logistic Regression  
from sklearn.linear\_model import LogisticRegression  
lr\_model = LogisticRegression()  
lr\_model.fit(X\_train, y\_train)  
  
- Trains a logistic regression model using the training data.  
  
### Random Forest  
from sklearn.ensemble import RandomForestClassifier  
rf\_model = RandomForestClassifier(random\_state=42)  
rf\_model.fit(X\_train, y\_train)  
  
- Trains a Random Forest classifier, which uses multiple decision trees for predictions.  
  
### Support Vector Machine (SVM)  
from sklearn.svm import SVC  
svm\_model = SVC(probability=True, random\_state=42)  
svm\_model.fit(X\_train, y\_train)  
  
- Trains an SVM classifier with probability estimates.  
  
### K-Nearest Neighbors (KNN)  
from sklearn.neighbors import KNeighborsClassifier  
knn\_model = KNeighborsClassifier()  
knn\_model.fit(X\_train, y\_train)  
  
- Trains a KNN model to classify based on the majority label of neighboring points.  
  
### XGBoost  
from xgboost import XGBClassifier  
xgb\_model = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42)  
xgb\_model.fit(X\_train, y\_train)  
  
- Trains an XGBoost classifier, a gradient boosting algorithm optimized for structured data.

## 9. Model Evaluation

### Accuracy Score  
from sklearn.metrics import accuracy\_score  
y\_pred\_lr = lr\_model.predict(X\_test)  
print("Logistic Regression Accuracy:", accuracy\_score(y\_test, y\_pred\_lr))  
  
- Evaluates the logistic regression model using accuracy as the metric.  
  
### Confusion Matrix  
from sklearn.metrics import confusion\_matrix  
cm\_lr = confusion\_matrix(y\_test, y\_pred\_lr)  
sns.heatmap(cm\_lr, annot=True, fmt='d')  
plt.title('Confusion Matrix - Logistic Regression')  
plt.show()  
  
- Displays the confusion matrix for logistic regression predictions.  
  
### Classification Report  
from sklearn.metrics import classification\_report  
print(classification\_report(y\_test, y\_pred\_lr))  
  
- Generates a detailed report showing precision, recall, F1-score, and support for each class.