Integrating Data Mining, Statistics, and Machine Learning for Enhanced Credit Risk Scoring

MSc Research Project
Msc Data Analytics

Donnal Benzon
Student ID: 23216531

School of Computing
National College of Ireland

Supervisor: Shubham Shubhnil

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Donnal Benzon
Student ID:	23216531
Programme:	Msc Data Analytics	**Year:** 2024-2025
Module:	Research project
Supervisor: **Submission Due Date:**	29/01/2025
Project Title:	Integrating Data Mining, Statistics, and Machine Learning for Enhanced Credit Risk Scoring
Word Count:	……………………..09……… **Page Count**…………………417………………………….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Donnal Benzon

Date: 11/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Donnal Benzon
Student ID: 23216531

1 Introduction

All the informations regarding python libraries,tools and technology is included in the document and while delveloping it for the research and all the things to be included in the project its self all this things help to find the highest accuracy.

2 Latest version of python

Python 3.12.3 is the most recent version . The specifics of this version may be found on the official Python website (Python, 2024). This is
Official link: https://docs.python.org/3/whatsnew/3.12.html

1.1 Python lib:

Python libraries that are required, use the terminal or the command prompt to get the libraries' functionality to be used in the project. The libraries that are used here are pandas, numpy, matplotlib.pyplot, seaborn, matplotlib.patches, sklearn.experimental.enable_hist_gradient_boosting, sklearn.ensemble.HistGradientBoostingRegressor, sklearn.model_selection.KFold, sklearn.model_selection.train_test_split, sklearn.metrics.median_absolute_error, sklearn.metrics.roc_auc_score, xgboost.XGBClassifier, catboost.CatBoostClassifier.

2 Visual studio code

Here the Visual Studio Code editor has been used (Garage, 2021). It is fluent in running Python scripts and is fully compatible with integrating with Python libraries for fast project results. This is best experienced in the latest version of Visual Studio Code, king of which is 2024 (version 1.84 or later). Make sure you have the right Python and its extensions to enable it run properly.

3 Summary version table

Software

Software name	Version	Download Link
Python	3.10.0	https://docs.python.org/3/whatsnew/3.10.html
Visual Studio Code	October 2024 (version 1.84)	https://code.visualstudio.com/download

Libraries

Library Category	Library Name	Purpose
Numerical Computing	numpy	Fundamental package for scientific computing in Python
Data Visualization	matplotlib.pyplot	Plotting library for creating static, animated, and interactive visualizations
matplotlib.patches	Patch drawing utilities for	

	matplotlib	
seaborn	Statistical data visualization based on matplotlib	
Machine Learning Libraries	sklearn.experimental.enable_hist_gradient_boosting	Enables histogram-based gradient boosting in scikit-learn
sklearn.ensemble.HistGradientBoostingRegressor	Histogram-based gradient boosting regression model	
sklearn.model_selection.KFold	Cross-validation technique	
sklearn.model_selection.train_test_split	Data splitting utility	
sklearn.metrics.median_absolute_error	Evaluation metric for regression models	
sklearn.metrics.roc_auc_score	Performance metric for classification models	
Gradient Boosting Classifiers	xgboost.XGBClassifier	XGBoost gradient boosting classifier
catboost.CatBoostClassifier	CatBoost gradient boosting classifier	

4 Implementation

Step 1: Importing the libraries.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.metrics import median_absolute_error
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from matplotlib.patches import ConnectionPatch

from sklearn.metrics import roc_auc_score
from sklearn.model_selection import KFold
from xgboost import XGBClassifier
from catboost import CatBoostClassifier          Chat (CTRL + I) / Edit (CTRL + L)
```

Step 2: Data Loading

```
# Load datasets
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
df_sub = pd.read_csv('sample_submission.csv')
df_origi = pd.read_csv('credit_risk_dataset.csv')
```

Step 3: Initial data exploration Training Set: 58,645 rows, 13 columns Test Set: 39,098 rows,12columns

```
columns = [
    'person_age', 'person_income', 'person_home_ownership',
    'person_emp_length', 'loan_intent', 'loan_grade',
    'loan_amnt', 'loan_int_rate', 'loan_status',
    'loan_percent_income', 'cb_person_default_on_file',
    'cb_person_cred_hist_length'
]
```

Step 4: Data preprocessing

```
# Combine training and original datasets
df_train = pd.concat([df_train, df_origi], axis=0)
```

Step 5: Handling Missing values

```
# Check missing values
missing_values = df_train.isnull().sum()

# Fill missing values
df_train['person_emp_length'].fillna(df_train['person_emp_length'].mean(), inplace=True)
df_train['loan_int_rate'].fillna(df_train['loan_int_rate'].mean(), inplace=True)
```

Step 6: PerformingExploratoryData Analysis

```
def stacked_bar_plot(df, feature, target='loan_status'):
    # Create stacked bar plot for categorical features
    crosstab = pd.crosstab(df[feature], df[target], normalize='index')
    crosstab.plot(kind='bar', stacked=True)
    plt.title(f'Stacked Bar Plot of {feature} vs {target}')
    plt.show()

def plot_boxplots(df, columns):
    # Create box plots for numerical features
    plt.figure(figsize=(12, 6))
    for i, col in enumerate(columns, 1):
        plt.subplot(1, len(columns), i)
        sns.boxplot(y=df[col], color='lightblue')
        plt.title(f'Box Plot of {col}')
    plt.tight_layout()
    plt.show()
```

Step 7: Data Preprocessing

```
def preprocess_data(df_train, df_test):
    # Label Encoding for categorical variables
    label_enc = LabelEncoder()
    label_cols = [
        'person_home_ownership',
        'loan_grade',
        'cb_person_default_on_file'
    ]
    for col in label_cols:
        df_train[col] = label_enc.fit_transform(df_train[col])
        df_test[col] = label_enc.transform(df_test[col])

    # One-Hot Encoding for loan_intent
    df_train = pd.get_dummies(df_train, columns=['loan_intent'], drop_first=True)
    df_test = pd.get_dummies(df_test, columns=['loan_intent'], drop_first=True)

    return df_train, df_test
```

Step 8: Feature Engineering

```
def feature_engineering(df):
    # Create new features to capture more complex relationships
    df['loan_to_income_ratio'] = df['loan_amnt'] / df['person_income']
    df['financial_burden'] = df['loan_amnt'] * df['loan_int_rate']
    df['income_per_year_emp'] = df['person_income'] / (df['person_emp_length'])
    df['cred_hist_to_age_ratio'] = df['cb_person_cred_hist_length'] / df['person_age']
    df['int_to_loan_ratio'] = df['loan_int_rate'] / df['loan_amnt']
    df['loan_int_emp_interaction'] = df['loan_int_rate'] * df['person_emp_length']
    df['debt_to_credit_ratio'] = df['loan_amnt'] / df['cb_person_cred_hist_length']
    df['int_to_cred_hist'] = df['loan_int_rate'] / df['cb_person_cred_hist_length']
    df['int_per_year_emp'] = df['loan_int_rate'] / (df['person_emp_length'])
    df['loan_amt_per_emp_year'] = df['loan_amnt'] / (df['person_emp_length'])
    df['income_to_loan_ratio'] = df['person_income'] / df['loan_amnt']

    return df
```

Step 9: Correlation Analysis

```
# Create and visualize correlation matrix
correlation_matrix = df_train.corr()
plt.figure(figsize=(15, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".1f", linewidths=0.2)
plt.title('Correlation Matrix')
plt.show()
```

Step 10:Model Preperation

```
# Create and visualize correlation matrix
correlation_matrix = df_train.corr()
plt.figure(figsize=(15, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".1f", linewidths=0.2)
plt.title('Correlation Matrix')
plt.show()
```

Step 11: Hyperparameter Tuning with RandomizedSearchCV

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint

# Define hyperparameter search space
param_dist = {
    'n_estimators': randint(100, 5000),
    'learning_rate': uniform(0.01, 0.3),
    'max_depth': randint(3, 20),
    'num_leaves': randint(10, 100),
    'min_child_samples': randint(1, 50),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5)
}

# Create LightGBM model
lgb_model = LGBMClassifier(random_state=42)

# Randomized Search
random_search = RandomizedSearchCV(
    estimator=lgb_model,
    param_distributions=param_dist,
    n_iter=100,  # Number of parameter settings sampled
    cv=5,
    scoring='roc_auc',
    random_state=42,
    n_jobs=-1
)

# Fit RandomizedSearchCV
random_search.fit(X, y)

# Best parameters and score
print("Best Parameters:", random_search.best_params_)
print("Best CV Score:", random_search.best_score_)
```

Step 12: Feature Importance Analysis

```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_feature_importance(model, X):
    # Get feature importances
    feature_importance = model.feature_importances_

    # Create dataframe of features and their importance
    feature_imp = pd.DataFrame({
        'feature': X.columns,
        'importance': feature_importance
    })

    # Sort features by importance
    feature_imp = feature_imp.sort_values('importance', ascending=False)

    # Plot
    plt.figure(figsize=(10, 6))
    sns.barplot(x='importance', y='feature', data=feature_imp.head(15))
    plt.title('Top 15 Most Important Features')
    plt.xlabel('Feature Importance')
    plt.tight_layout()
    plt.show()

# Use the best model from RandomizedSearchCV
best_model = random_search.best_estimator_
plot_feature_importance(best_model, X)
```