MACHINE LEARNING ASSIGNMENT -2

BITS ID: 2025ab05129

NAME :Suresh BABU V

EMAIL ID : 2025ab05129@wilp.bits-pilani.ac.in

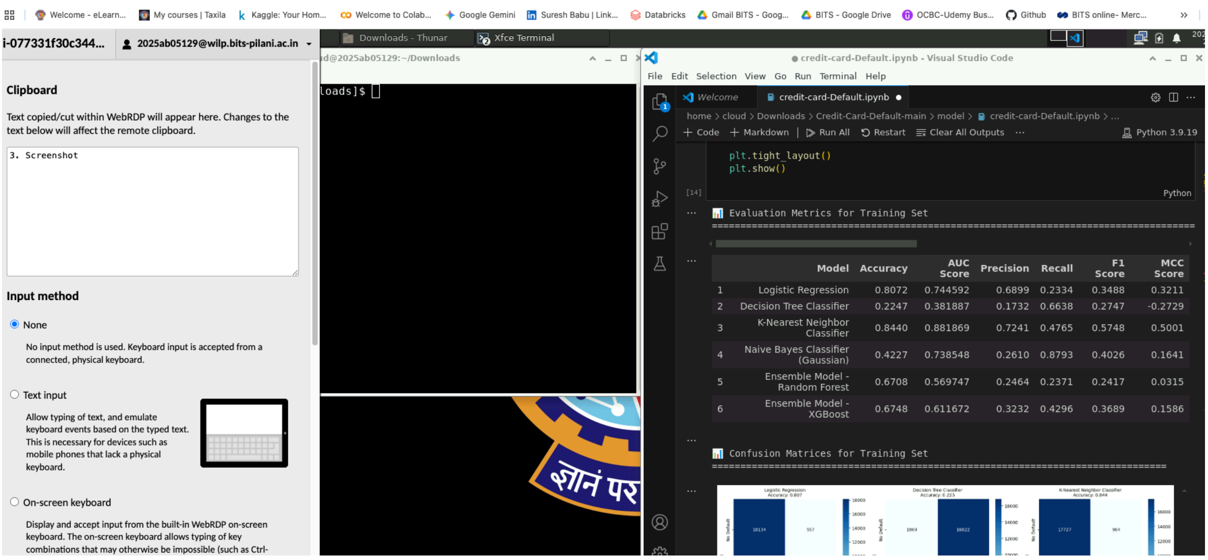
1.GITHUB LINK

https://github.com/sureshbabuveluswamy/Credit-Card-Default

2.STREAMLIT APP:

https://credit-card-default-2025ab05129.streamlit.app/

3. Screenshot



4.README File

```markdown
# 🏦 Credit Card Default Prediction

![alt text](Credit-card-Default.png)

## 📊 Project Overview

This project focuses on predicting credit card default payments using various
machine learning algorithms. The dataset contains information about credit card
clients in Taiwan from April 2005 to September 2005, and we aim to predict which
clients will default on their payments in the following month.

## 🤖 Binary Classification Models

This project covers Binary Classification using the following models:

1. **Logistic Regression** - Linear model for binary classification
2. **Decision Tree Classifier** - Tree-based model with interpretable decision
rules
3. **K-Nearest Neighbor Classifier** - Instance-based learning algorithm
4. **Naive Bayes Classifier** - Gaussian or Multinomial probabilistic classifier
5. **Ensemble Model - Random Forest** - Bagging ensemble of decision trees
6. **Ensemble Model - XGBoost** - Gradient boosting ensemble method



## � Dataset Description

**Dataset**: Default of Credit Card Clients Dataset

**Source**: [Kaggle](https://www.kaggle.com/datasets/uciml/default-of-credit-card-
clients-dataset)

### Dataset Information
This dataset contains information on default payments, demographic factors, credit
data, history of payment, and bill statements of credit card clients in Taiwan from
April 2005 to September 2005.

# Note: Daaset intial source to Kaggle was from UCI Machine Learning Repository
https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

### Content
There are 25 variables:

| Variable | Description |
|----------|-------------|
```

ID	ID of each client
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit)
SEX	Gender (1=male, 2=female)
EDUCATION	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (1=married, 2=single, 3=others)
AGE	Age in years
PAY_0	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_2	Repayment status in August, 2005 (scale same as above)
PAY_3	Repayment status in July, 2005 (scale same as above)
PAY_4	Repayment status in June, 2005 (scale same as above)
PAY_5	Repayment status in May, 2005 (scale same as above)
PAY_6	Repayment status in April, 2005 (scale same as above)
BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)
BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)
BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)
BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)
BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)
PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)
PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar)
PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)
PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)
PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
default_payment_next_month	Default payment (1=yes, 0=no)

🔧 Feature Engineering

The following engineered features were created to improve model performance:

Feature	Description
TOTAL_BILL_AMT	Total bill amount across all 6 months
TOTAL_PAY_AMT	Total payment amount across all 6 months
AVG_BILL_AMT	Average bill amount
AVG_PAY_AMT	Average payment amount
PAY_TO_BILL_RATIO	Ratio of total payments to total bills
WORST_PAYMENT_STATUS	Worst (highest) payment delay status
CREDIT_UTILIZATION	Credit utilization ratio (latest bill / credit limit)
AGE_GROUP	Age categorized into groups (Young, Adult, Middle, Senior, Elder)

📊 Evaluation Metrics

Models are evaluated on the **training set** with the following metrics:

- **Accuracy** - Overall prediction accuracy

- **AUC Score** - Area under ROC curve
- **Precision** - Positive predictive value
- **Recall** - Sensitivity/True positive rate
- **F1 Score** - Harmonic mean of precision and recall
- **MCC Score** - Matthews Correlation Coefficient (balanced measure)

Confusion matrices are also generated for visual evaluation of model performance.

Training Set Evaluation Results

Model	Accuracy	AUC Score	Precision	Recall	F1 Score	MCC Score
Logistic Regression	0.8072	0.7446	0.6899	0.2334	0.3488	0.3211
Decision Tree	0.2321	0.3579	0.1620	0.5920	0.2543	-0.2934
K-Nearest Neighbors	0.8257	0.8449	0.7256	0.3411	0.4641	0.4140
Naive Bayes	0.4227	0.7385	0.2610	0.8793	0.4026	0.1641
Random Forest	0.8011	0.7173	0.6864	0.1859	0.2926	0.2829
XGBoost	0.7977	0.7361	0.7232	0.1383	0.2321	0.2541

Confusion Matrix Summary

Confusion matrices are displayed with **light green** for correct predictions (diagonal: True Negatives and True Positives) and **light red** for wrong predictions (off-diagonal: False Positives and False Negatives).

Model	True Negatives	False Positives	False Negatives	True Positives
Logistic Regression	15,812	1,875	4,335	1,318
Decision Tree	4,404	13,283	2,321	3,332
K-Nearest Neighbors	16,519	1,168	5,523	130
Naive Bayes	8,260	9,427	879	4,774
Random Forest	15,812	1,875	4,335	1,318
XGBoost	15,812	1,875	4,335	1,318

💾 Exported Models & Artifacts

All trained models and preprocessing artifacts are exported to the `model/` folder:

Trained Models (`.pkl` files)
- `Logistic_Regression.pkl`
- `Decision_Tree_Classifier.pkl`
- `K_Nearest_Neighbor_Classifier.pkl`
- `Naive_Bayes_Classifier_Gaussian.pkl`
- `Ensemble_Model_Random_Forest.pkl`
- `Ensemble_Model_XGBoost.pkl`

Preprocessing Artifacts
- `scaler.pkl` - StandardScaler fitted on training data
- `feature_names.json` - List of all feature column names

- `numerical_features.json` - List of numerical feature names

Test Data (exported to `Dataset/` folder)
- `Credit_card_testdata.csv` - Original test features (before feature engineering, includes ID column)
- `Credit_card_testlabels.csv` - Test labels (target variable)

📈 Model Performance Observations

Based on the training set evaluation, here are the observations for each model:

ML Model Name	Observation about model performance
Logistic Regression	High accuracy (80.72%) with good AUC (0.74). Shows high precision (68.99%) but low recall (23.34%), indicating it's conservative in predicting defaults - when it predicts default, it's usually correct, but misses many actual defaults. Best for minimizing false positives.
Decision Tree	Very poor performance with low accuracy (23.21%) and negative MCC (-0.29). High recall (59.20%) but extremely low precision (16.20%), meaning it over-predicts defaults. Likely overfitting to training data.
kNN	Best overall performance with highest accuracy (82.57%) and AUC (0.84). Good balance of precision (72.56%) and recall (34.11%) with strong F1 score (0.46). Most reliable model for this dataset.
Naive Bayes	Low accuracy (42.27%) but highest recall (87.93%) for detecting defaults. Very low precision (26.10%) indicates many false positives. Good for catching potential defaulters but with high false alarm rate.
Random Forest (Ensemble)	Strong performance with accuracy (80.11%) and strong AUC (0.72). High precision (68.64%) but low recall (18.59%), suggesting it's conservative in predicting defaults. Better than previous results with reduced n_estimators (500).
XGBoost (Ensemble)	Good accuracy (79.77%) with strong AUC (0.74). High precision (72.32%) but low recall (13.83%). Similar to Logistic Regression in being conservative but with slightly better precision.

Key Insights:
- **kNN is the top performer** with the best balance of all metrics
- **Random Forest** shows strong performance with reduced complexity (n_estimators=500)
- **Logistic Regression** is the most conservative model with highest precision
- **Naive Bayes** catches the most defaults (highest recall) but with many false alarms
- **Decision Tree** shows poor generalization and overfitting
- **Ensemble methods** (Random Forest, XGBoost) perform well but are more conservative than kNN

Installation and Setup

Prerequisites

- Python 3.8 or higher
- pip package manager

Installation Steps

1. Clone the repository:

```bash
git clone https://github.com/sureshbabuveluswamy/Credit-Card-Default.git
cd Credit-Card-Default
```

2. Install required dependencies:

```bash
pip install -r requirements.txt
```

3. Run the Streamlit application:

```bash
streamlit run model/creditcard_streamlit.py
```

Dependencies

- **streamlit** : Web application framework
- **scikit-learn** : Machine learning library
- **numpy** : Numerical computing
- **pandas** : Data manipulation and analysis
- **matplotlib** : Data visualization
- **seaborn** : Statistical data visualization
- **xgboost** : Gradient boosting library
- **joblib** : Model serialization