# CREDIT SCORE PREDICTION REPORT

## ****1. Title Page****

**Project Title:** Credit Score Prediction: Data Cleaning & Model Training  
**Author:** Siddhi Gupta  
**Date:** 10/03/2025

**Institution Name**- KIET Group of Institutions

**Course Name**-Introduction to AI  
**Platform Used:** Google Colab  
**Language:** Python

## ****2. Introduction****

### ****Problem Statement****

Credit risk assessment is crucial for financial institutions to determine loan eligibility. This project focuses on **cleaning, transforming, and analyzing financial data** to predict a customer's credit score category.

### ****Objectives****

1. Clean and preprocess financial data  
2. Transform categorical and numerical features  
3. Train and evaluate a **RandomForestClassifier** model  
4. Save and deploy the trained model

## ****3. Methodology****

### ****Data Preprocessing****

1.Dropped unnecessary columns (ID, Customer\_ID, SSN, Name, Month)  
2. Converted object-type numerical columns to float  
3.Filled missing values (median for numerical, mode for categorical)  
4.Transformed **Credit\_History\_Age** to months  
5.Encoded categorical variables (Occupation, Credit Mix, etc.)  
6.Numerical features using **StandardScaler**

### ****Model Selection & Training****

**1.RandomForestClassifier** (handles categorical & numerical data efficiently)  
2. **Train-test split (80% train, 20% test)**  
3. **Evaluation Metrics:** Accuracy Score, Classification Report

1. **Code Typed**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load the dataset

df = pd.read\_csv('test.csv')

# Display the first few rows of the dataset

print(df.head())

# Check for missing values

print(df.isnull().sum())

# Handle missing values

# For numerical columns, fill missing values with the median

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

df[numerical\_cols] = df[numerical\_cols].fillna(df[numerical\_cols].median())

# For categorical columns, fill missing values with the mode

categorical\_cols = df.select\_dtypes(include=['object']).columns

df[categorical\_cols] = df[categorical\_cols].fillna(df[categorical\_cols].mode().iloc[0])

# Check for duplicate rows

print(df.duplicated().sum())

# Remove duplicate rows

df = df.drop\_duplicates()

# Convert categorical variables to numerical using Label Encoding

label\_encoder = LabelEncoder()

for col in categorical\_cols:

df[col] = label\_encoder.fit\_transform(df[col])

# Feature Scaling

scaler = StandardScaler()

df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

# Split the dataset into features and target variable

X = df.drop('Credit\_Mix', axis=1) # Assuming 'Credit\_Mix' is the target variable

y = df['Credit\_Mix']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the RandomForestClassifier

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

# Print classification report

print(classification\_report(y\_test, y\_pred))

# Feature Importance

importances = model.feature\_importances\_

feature\_importance\_df = pd.DataFrame({'Feature': X.columns, 'Importance': importances})

feature\_importance\_df = feature\_importance\_df.sort\_values(by='Importance', ascending=False)

print(feature\_importance\_df)

# Save the cleaned and transformed dataset

df.to\_csv('cleaned\_credit\_data.csv', index=False)

1. **ScreenShots Output**



