**Lab 5**

**Credit card defaulter prediction**

1.Financial threats are displaying a trend about the credit risk of commercial banks as the incredible improvement in the financial industry has arisen. In this way, one of the biggest threats faces by commercial banks is the risk prediction of credit clients. The goal is to predict the probability of credit default based on credit card owner's characteristics and payment history.

**Approach:** The classical machine learning tasks like Data Exploration, Data Cleaning,

Feature Engineering, Model Building and Model Testing. Try out different machine

learning algorithms that’s best fit for the above case.

**Results:** You have to build a solution that should able to predict the probability of credit

default based on credit card owner’s characteristics and payment history.

**2. Dataset Description**:

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.

Content

There are 25 variables:

* **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)

3. Dataset link:

<https://drive.google.com/file/d/1-9Yy0vnyeH_en6hyWykFqASGHY_HsVLS/view?usp=share_link>

4. Exercise

1. Data cleaning

2. Feature selection

3. Apply all classification models with cross validation and grid search

4. Compare precision, recall, accuracy, F1-score

5. Conclude

Sample code:

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import SelectKBest, f\_classif

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification\_report

# Load the dataset

file\_path = 'UCI\_Credit\_Card.csv' # Update this path if necessary

data = pd.read\_csv(file\_path)

# Data exploration

print(data.info()) # To get info about the dataset

print(data.head()) # To view the first few rows

# Define feature matrix X and target vector y

X = data.drop(columns=['ID', 'default.payment.next.month']) # Assuming 'default.payment.next.month' is the target

y = data['default.payment.next.month']

# Split the data into train and test sets

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

# Feature selection: SelectKBest to choose top k features

k = 10 # You can choose the value of k based on your preference

select\_kbest = SelectKBest(score\_func=f\_classif, k=k)

# Standardization

scaler = StandardScaler()

# Define classifiers

models = {

'KNN': KNeighborsClassifier(),

'DecisionTree': DecisionTreeClassifier(random\_state=42),

'SVM': SVC(),

'LogisticRegression': LogisticRegression(max\_iter=1000, random\_state=42)

}

# Hyperparameter grids

param\_grids = {

'KNN': {

'knn\_\_n\_neighbors': [3, 5, 7, 9],

'knn\_\_weights': ['uniform', 'distance'],

'knn\_\_metric': ['euclidean', 'manhattan']

},

'DecisionTree': {

'decisiontree\_\_max\_depth': [5, 10, 15, 20],

'decisiontree\_\_min\_samples\_split': [2, 5, 10]

},

'SVM': {

'svm\_\_C': [0.1, 1, 10],

'svm\_\_kernel': ['linear', 'rbf']

},

'LogisticRegression': {

'lr\_\_C': [0.1, 1, 10],

'lr\_\_penalty': ['l2']

}

}

# Loop through models and apply GridSearchCV

for model\_name, model in models.items():

print(f"\nRunning GridSearchCV for {model\_name}")

# Create pipeline

pipeline = Pipeline([

('scaler', scaler),

('select\_kbest', select\_kbest),

(model\_name.lower(), model) # Add classifier to the pipeline

])

# Define parameter grid for the current model

param\_grid = param\_grids[model\_name]

# Perform grid search

grid\_search = GridSearchCV(pipeline, param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)

# Fit the model

grid\_search.fit(X\_train, y\_train)

# Output best parameters and cross-validation score

print(f"Best hyperparameters for {model\_name}: {grid\_search.best\_params\_}")

print(f"Best cross-validation score for {model\_name}: {grid\_search.best\_score\_:.4f}")

# Predict on the test set

y\_pred = grid\_search.predict(X\_test)

# Output classification report

print(f"Classification Report for {model\_name}:\n{classification\_report(y\_test, y\_pred)}")