|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Kingdom of Saudi Arabia**  **Ministry of Education**  **University of Jeddah**  **College of Computer Science and Engineering**  **Department of Computer Science and Artificial Intelligence** | Logo, company name  Description automatically generated | **المملكة العربية السعودية**  **وزارة التعليم**  **جامعة جدّة**  **كلية علوم وهندسة الحاسب**  **قسم علوم الحاسب والذكاء الاصطناعي** | |  |  |

|  |
| --- |
| **Lab 6** |
| **CCAI 312 Pattern Recognition** |
| **Third Trimester 2023**   |  |  | | --- | --- | | **Lab Date/Time:**  **Lab assignment submission Date/Time:** |  | | **Student Name: \_\_\_\_\_\_\_\_\_bushra dajam\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Student ID: \_\_\_\_\_\_\_\_\_2110054\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |

|  |  |
| --- | --- |
| **Instructor Name** | **Section** |
|  |  |

**Instructions**:

The lab assignments must be submitted before the allocated Date/Time.

The lab assignments must by uploaded on LMS / sent by email to teacher@uj.edu.sa.

Plagiarism will be punished according to university rules.

|  |  |
| --- | --- |
| **PLO/CLO** | **SO** |
| **PLO S2 (CLO 2):** **Implement** a suitable pattern recognition technique for a given problem using Python | **SO 2:** Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Max Score** | **Student Score** |
| **PLO S2 / CLO 2 / SO 2** | **Task 1** | **2** |  |
| **PLO S2 / CLO 2 / SO 2** | **Task 2** | **2** |  |
| **Total** | |  |  |

# Lab Description

In this lab, we will explore the concept of synthetic data and how it can be generated using Scikit-Learn. We will focus on a binary classification problem and demonstrate how to train and test four different classifiers - KNN, Naive Bayes, Decision Trees, and Random Forest - on the synthetic dataset. The performance of each classifier will be evaluated using a range of metrics, including confusion matrix, precision, recall, fscore, and ROC curve. By the end of the lab, you will have a better understanding of how synthetic data can be used to simulate real-world scenarios and evaluate the effectiveness of machine learning algorithms in different contexts.

# Objectives

# • Understand the concept of synthetic data and its importance in machine learning.

# • Learn how to generate synthetic data using Scikit-Learn for binary classification problems.

# • Train and test four different classifiers - KNN, Naive Bayes, Decision Trees, and Random Forest - on a synthetic dataset.

# • Evaluate the performance of each classifier using various metrics, including confusion matrix, precision, recall, fscore, and ROC curve.

# • Compare the effectiveness of each classifier in predicting the binary classification problem using the synthetic dataset.

# Lab Tool(s)

<https://www.kaggle.com/>

or

<https://colab.research.google.com/>

# Lab Deliverables

Submit A notebook to Blackboard containing your solution to the lab assessment at the end of this document.

Other Resources:

<https://mostly.ai/synthetic-data/what-is-synthetic-data>

<https://www.turing.com/kb/synthetic-data-generation-techniques>

Model Evaluation

**Synthetic data**

In machine learning, synthetic data refers to artificially generated data that is created using algorithms or other methods rather than being collected from real-world sources. Synthetic data can be used for a variety of purposes, including:

1. **Data augmentation**: Synthetic data can be used to augment existing datasets, allowing machine learning models to be trained on larger and more diverse datasets.
2. **Privacy protection**: Synthetic data can be used to protect sensitive information by generating synthetic data that is statistically similar to the original data but does not contain any sensitive information.
3. **Simulation:** Synthetic data can be used to simulate scenarios that are difficult or impossible to reproduce in the real world, allowing machine learning models to be trained on a wider range of scenarios.
4. **Testing:** Synthetic data can be used to test machine learning models in a controlled environment, allowing for more comprehensive testing and evaluation.

There are many methods for generating synthetic data, including generative adversarial networks (GANs), variational autoencoders (VAEs), and other deep learning techniques. These methods can be used to generate data that is statistically similar to real-world data while maintaining privacy and protecting sensitive information.

In summary, synthetic data provides a valuable tool for machine learning researchers and practitioners to enhance and extend their datasets in a variety of ways.

Here are three examples of synthetic datasets with code for visualization:

1. Random scatter plot with a linear trend:

import numpy as np  
import matplotlib.pyplot as plt  
  
# Generate random data  
np.random.seed(42)  
x = np.random.rand(100)  
y = x + np.random.rand(100)  
  
# Create scatter plot  
plt.scatter(x, y)  
plt.plot(x, x, color='red')  
  
# Add labels and title  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.title('Random scatter plot with a linear trend')  
  
# Show plot  
plt.show()

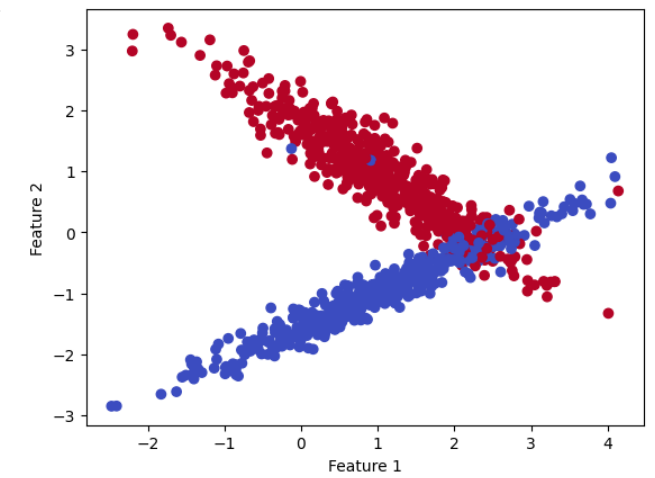
This code generates a scatter plot with 100 randomly generated points that follow a linear trend. The plot also includes a red line that represents the ideal linear relationship between the two variables.



1. Two-class classification dataset:

from sklearn.datasets import make\_classification  
import matplotlib.pyplot as plt  
  
# Generate classification dataset  
X, y = make\_classification(n\_samples=1000, n\_features=2, n\_redundant=0,  
 n\_informative=2, n\_clusters\_per\_class=1, class\_sep=1.0,  
 random\_state=42)  
  
# Plot dataset  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()

This code generates a two-dimensional classification dataset with 1000 samples and 2 features. The plot shows the distribution of the data points, with each class represented by a different color.



1. Circles Dataset:

The circles dataset is a synthetic dataset consisting of two concentric circles. This dataset is useful for testing classification algorithms that can handle non-linearly separable data. Here's an example of how to generate and visualize this dataset using scikit-learn:

from sklearn.datasets import make\_circles  
import matplotlib.pyplot as plt  
  
# Generate circles dataset  
X, y = make\_circles(n\_samples=1000, noise=0.05, factor=0.5, random\_state=42)  
  
# Plot dataset  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()

This code generates a circles dataset with 1000 samples and some added noise using the make\_circles() function from scikit-learn. The resulting dataset is plotted using the

scatter() function from matplotlib, with the color of each point corresponding to its label.

Chart, scatter chart

Description automatically generated

Part1

In this section we will trains and tests four classifiers - KNN, Naive Bayes, Decision Trees, and Random Forest - on a synthetic dataset for binary classification. We will computes various evaluation metrics, including confusion matrix, precision, recall, fscore, and ROC curve, to assess the performance of each classifier.

The steps required are:

1. Import the necessary libraries, including make\_classification from sklearn.datasets, train\_test\_split from sklearn.model\_selection, and various classifiers, evaluation metrics, and plotting functions from sklearn.

from sklearn.datasets import make\_classification  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report, confusion\_matrix, precision\_recall\_fscore\_support, roc\_curve, auc  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestClassifier  
import matplotlib.pyplot as plt

1. Generate synthetic data for binary classification using make\_classification.

# Generate synthetic data for binary classification  
X, y = make\_classification(n\_samples=500, n\_features=4, n\_informative=3, n\_redundant=0, random\_state=42)

1. Split the data into training and testing sets using train\_test\_split.

# Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

1. Define a list of classifiers to train and test, including KNN, Naive Bayes, Decision Trees, and Random Forest.

# Define a list of classifiers to train and test  
classifiers = [  
    KNeighborsClassifier(n\_neighbors=5),  
    GaussianNB(),  
    DecisionTreeClassifier(random\_state=42),  
    RandomForestClassifier(n\_estimators=100, random\_state=42)  
]

1. Iterate over the list of classifiers using a for loop.
   1. Train each classifier using fit on the training data.
   2. Test each classifier on the testing data by making predictions using predict and predict\_proba.
   3. Compute and print the confusion matrix using confusion\_matrix.
   4. Compute and print the precision, recall, fscore, and support using precision\_recall\_fscore\_support.
   5. Compute and plot the ROC curve and AUC using roc\_curve and auc.

# Train and test each classifier in the list using a for loop  
for clf in classifiers:  
    # Train the classifier  
    clf.fit(X\_train, y\_train)  
  
    # Test the classifier and make predictions  
    y\_pred = clf.predict(X\_test)  
    proba = clf.predict\_proba(X\_test)[:, 1]  
  
    # Compute and print the confusion matrix  
    cm = confusion\_matrix(y\_test, y\_pred)  
    print(f"Confusion Matrix for {clf.\_\_class\_\_.\_\_name\_\_}:")  
    print(cm)  
  
    # Compute and print the precision, recall, fscore, and support  
    prf = precision\_recall\_fscore\_support(y\_test, y\_pred, average='binary')  
    print(f"Precision, Recall, F-score, Support for {clf.\_\_class\_\_.\_\_name\_\_}:")  
    print(f"Precision: {prf[0]:.2f}")  
    print(f"Recall: {prf[1]:.2f}")  
    print(f"F-score: {prf[2]:.2f}")  
    print(f"Support: {prf[3]}")  
  
    # Compute and plot the ROC curve and AUC  
    fpr, tpr, thresholds = roc\_curve(y\_test, proba)  
    roc\_auc = auc(fpr, tpr)  
    plt.plot(fpr, tpr, label=f"{clf.\_\_class\_\_.\_\_name\_\_} (AUC = {roc\_auc:.2f})")  
    # Plot the ROC curve for all classifiers  
    plt.plot([0, 1], [0, 1], 'k--', lw=2)  
    plt.xlim([0.0, 1.0])  
    plt.ylim([0.0, 1.05])  
    plt.xlabel('False Positive Rate')  
    plt.ylabel('True Positive Rate')  
    plt.title('Receiver operating characteristic')  
    plt.legend(loc="lower right")  
    plt.show()

Part2

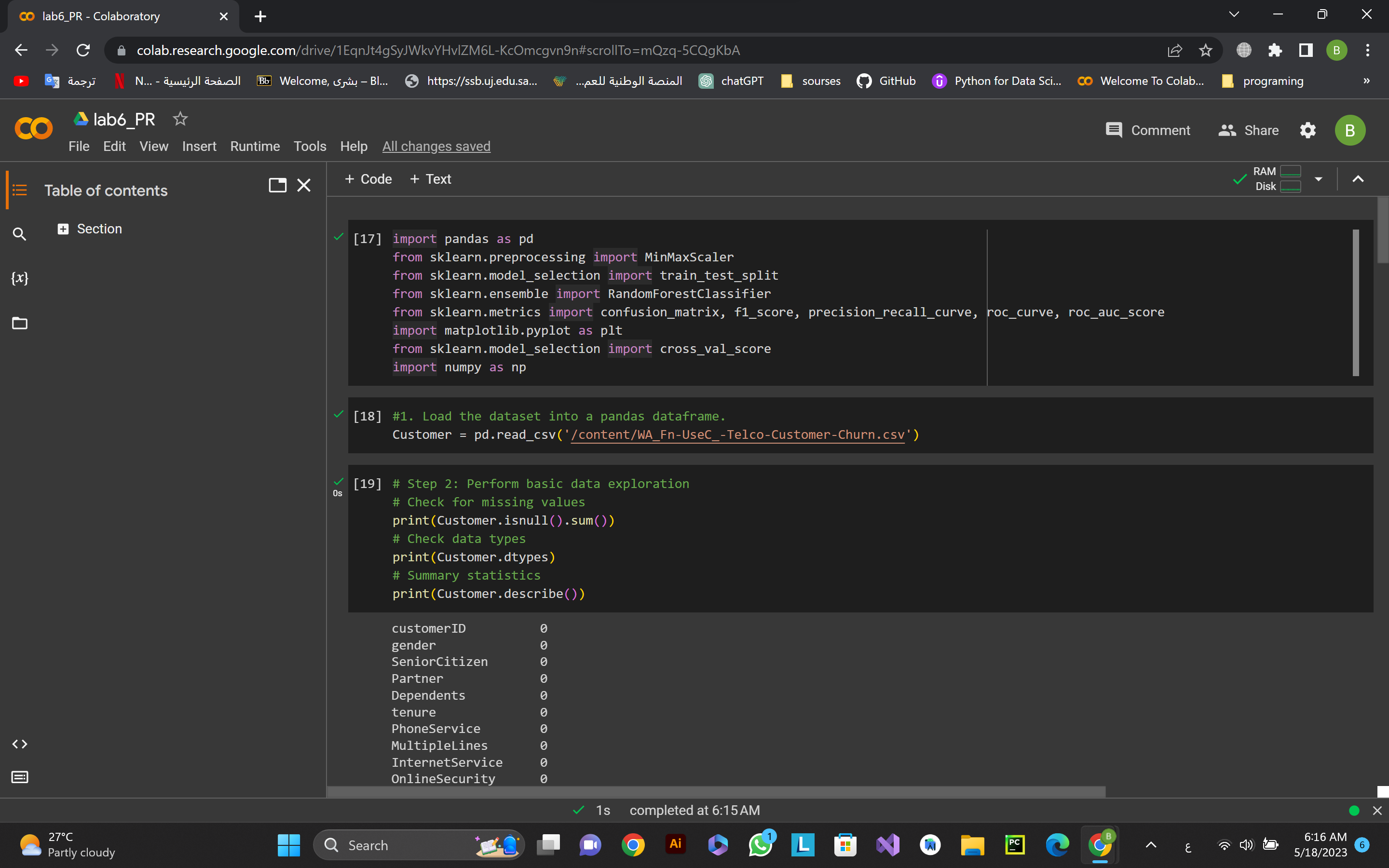
## **Telco Customer Churn**

## **Dataset**

**This dataset tracks a fictional telco company's customer churn based on various factors. The churn column indicates whether the customer departed within the last month. Other columns include gender, dependents, monthly charges, and many with information about the types of services each customer has.**

Task **Task 1: [PLO S2 / CLO 2 / SO 2] [4 marks]**

1. Load the dataset into a pandas dataframe.
2. Perform basic data exploration, including checking for missing values, data types, and summary statistics.
3. Remove customer IDs from the data set
4. Convert the predictor variable in a binary numeric variable.
5. Convert all the categorical variables into dummy variables
6. Show Correlation of "Churn" with other variables
7. Apply normalization techniques to standardize the numerical features to have zero mean and unit variance, such as Min-Max scaling, Z-score normalization, or Robust scaling.
8. Split the dataset into training and testing sets using a stratified sampling strategy to preserve the proportion of the target variable in each set.
9. Train the Random Forest model on the training set using the sklearn library.
10. Make predictions on the testing set and evaluate the model performance using confusion matrix, precision-recall curve, F1-score, and ROC curve.
11. Perform 5-fold cross-validation and calculate the mean accuracy score



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated