Kingdom of Saudi Arabia
Ministry of Education
University of Jeddah
College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence

جامعة جدة
University of Jeddah

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.

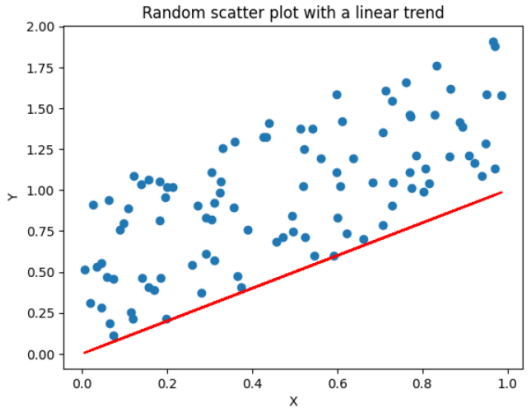
Here are three examples of synthetic datasets with code for visualization:

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()
```

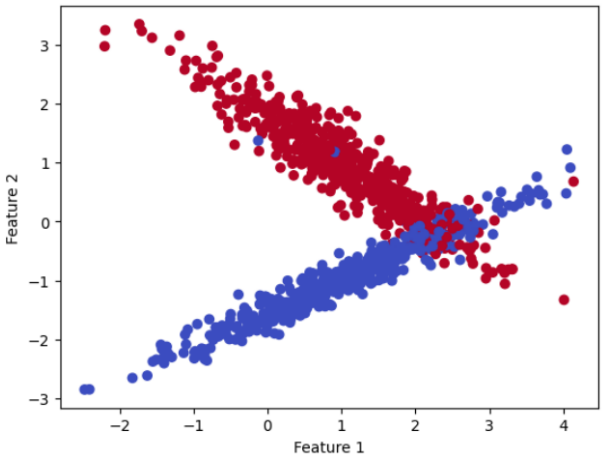
This code generates a scatter plot with 100 randomly generated points that follow a linear trend. The plot also includes a red line that represents the ideal linear relationship between the two variables.




2. 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.

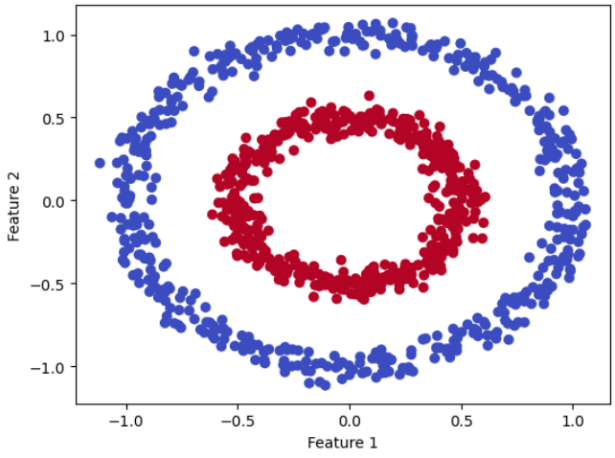



3. Circles Dataset:

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.



# 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_matr
ix, 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
```

2. 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_informa
tive=3, n_redundant=0, random_state=42)
```

3. 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_si
ze=0.3, random_state=42)
```

4. 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)
]
```

5. Iterate over the list of classifiers using a for loop.
    a. Train each classifier using `fit` on the training data.
    b. Test each classifier on the testing data by making predictions using `predict` and `predict_proba`.
    c. Compute and print the confusion matrix using `confusion_matrix`.
    d. Compute and print the precision, recall, fscore, and support using `precision_recall_fscore_support`.
    e. 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 suppor
t
    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 = {r
oc_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

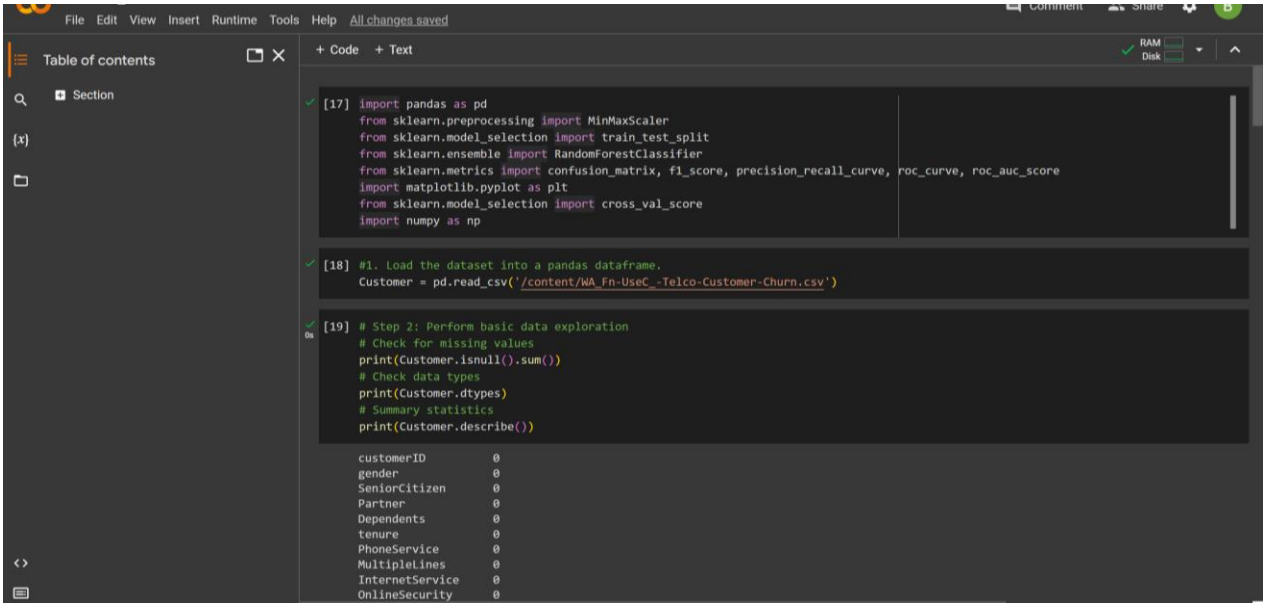
**Telco Customer Churn**

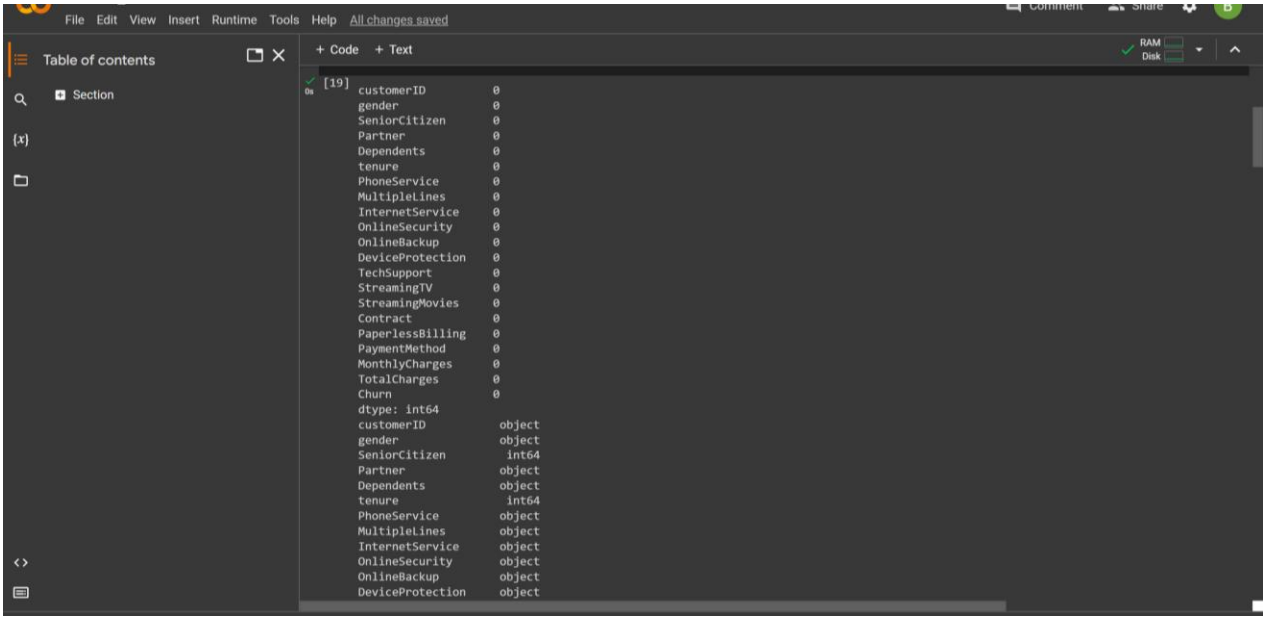
**Dataset**

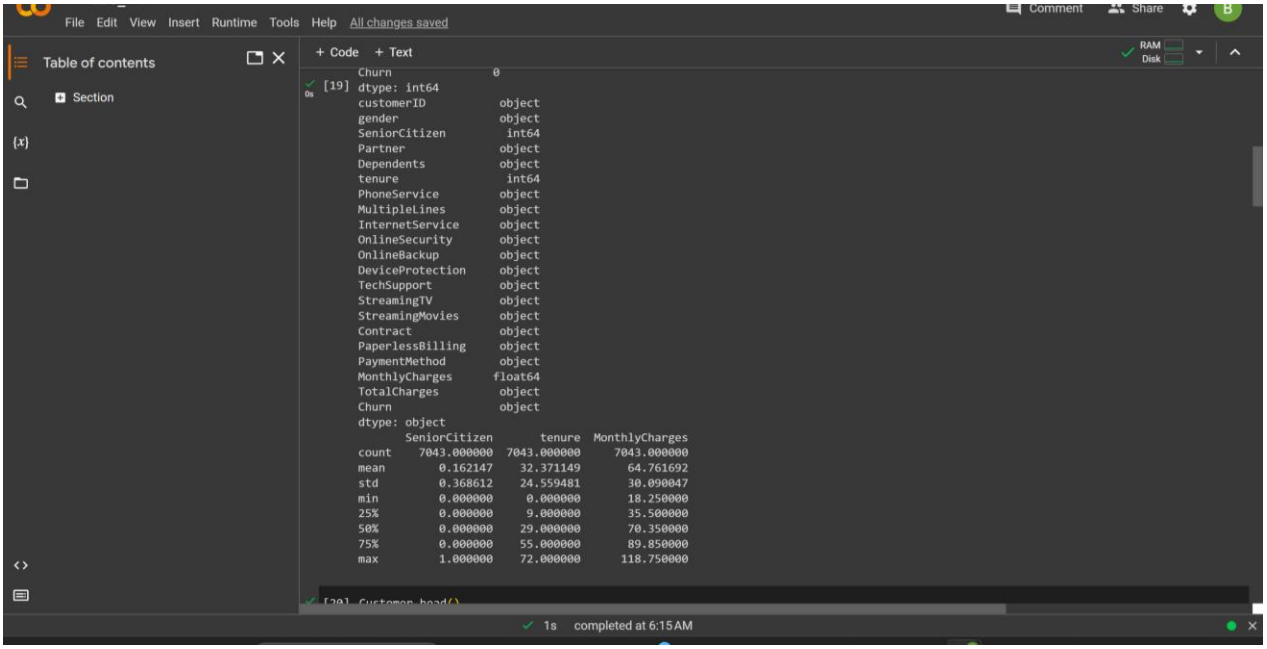
This dataset tracks a fictional telco company's customer churn based on various factors. The churn column indicates whether the customer departed within the last month. Other columns include gender, dependents, monthly charges, and many with information about the types of services each customer has.

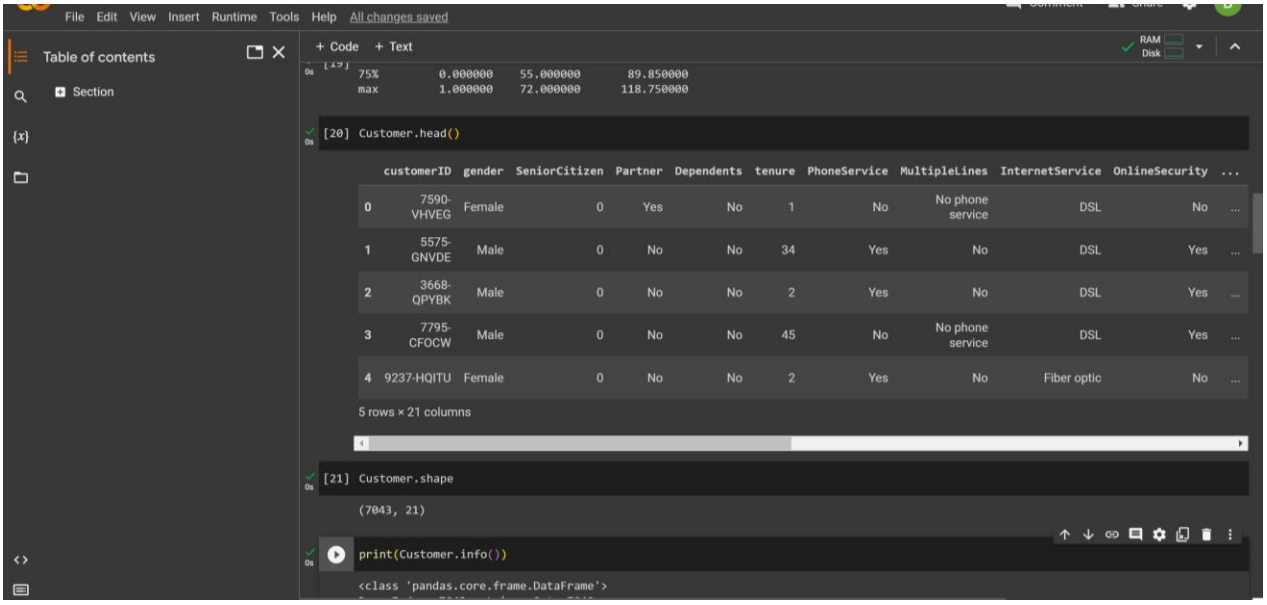
| **Task 1: [PLO S2 / CLO 2 / SO 2]** | **[4 marks]** |
|---|---|

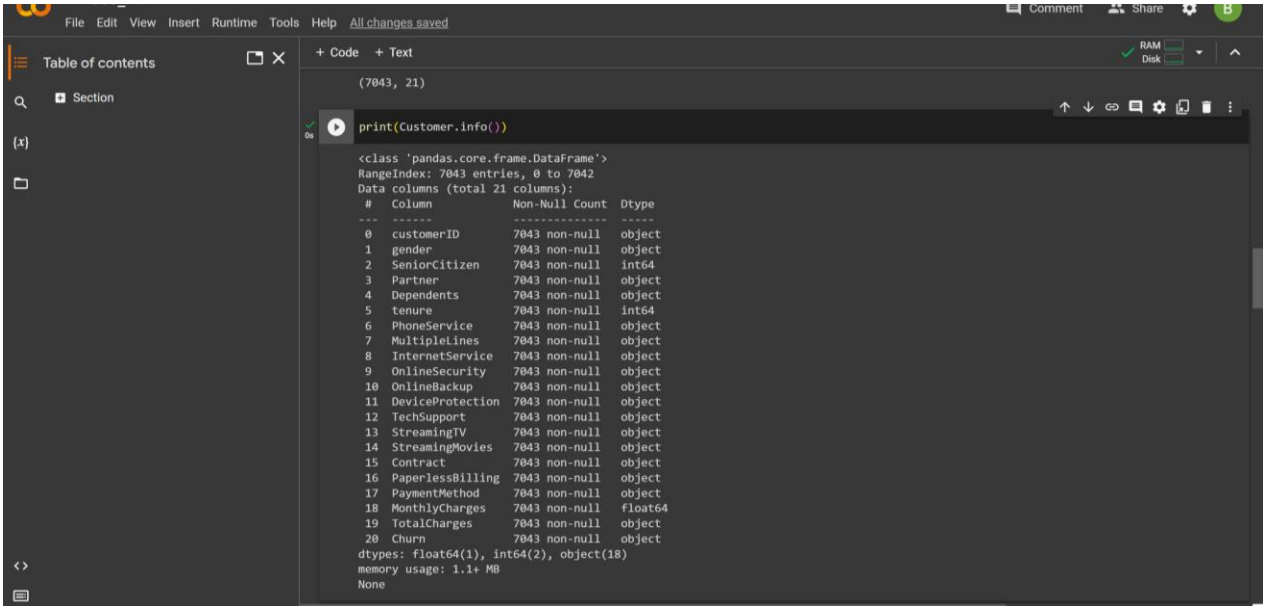
1. Load the dataset into a pandas dataframe.
2. Perform basic data exploration, including checking for missing values, data types, and summary statistics.
3. Remove customer IDs from the data set
4. Convert the predictor variable in a binary numeric variable.
5. Convert all the categorical variables 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

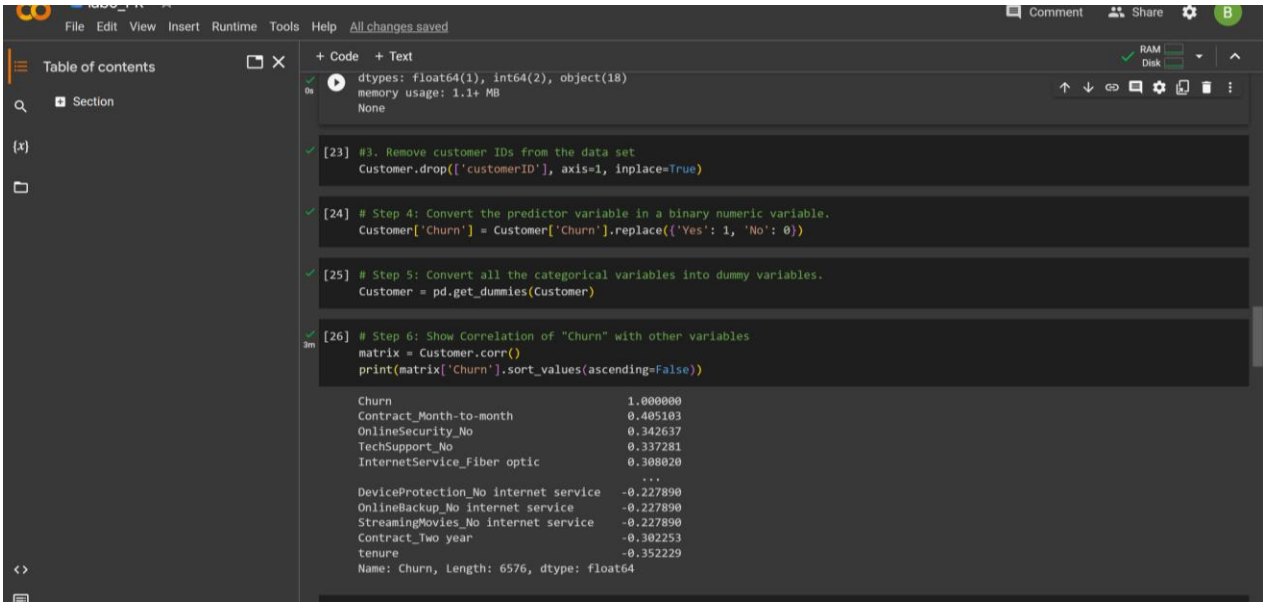

File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code + Text
RAM
Disk
[17] import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, f1_score, precision_recall_curve, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
import numpy as np
[18] #1. Load the dataset into a pandas dataframe.
Customer = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
[19] # Step 2: Perform basic data exploration
# Check for missing values
print(Customer.isnull().sum())
# Check data types
print(Customer.dtypes)
# Summary statistics
print(Customer.describe())
customerID 0
gender 0
SeniorCitizen 0
Partner 0
Dependents 0
tenure 0
PhoneService 0
MultipleLines 0
InternetService 0
OnlineSecurity 0

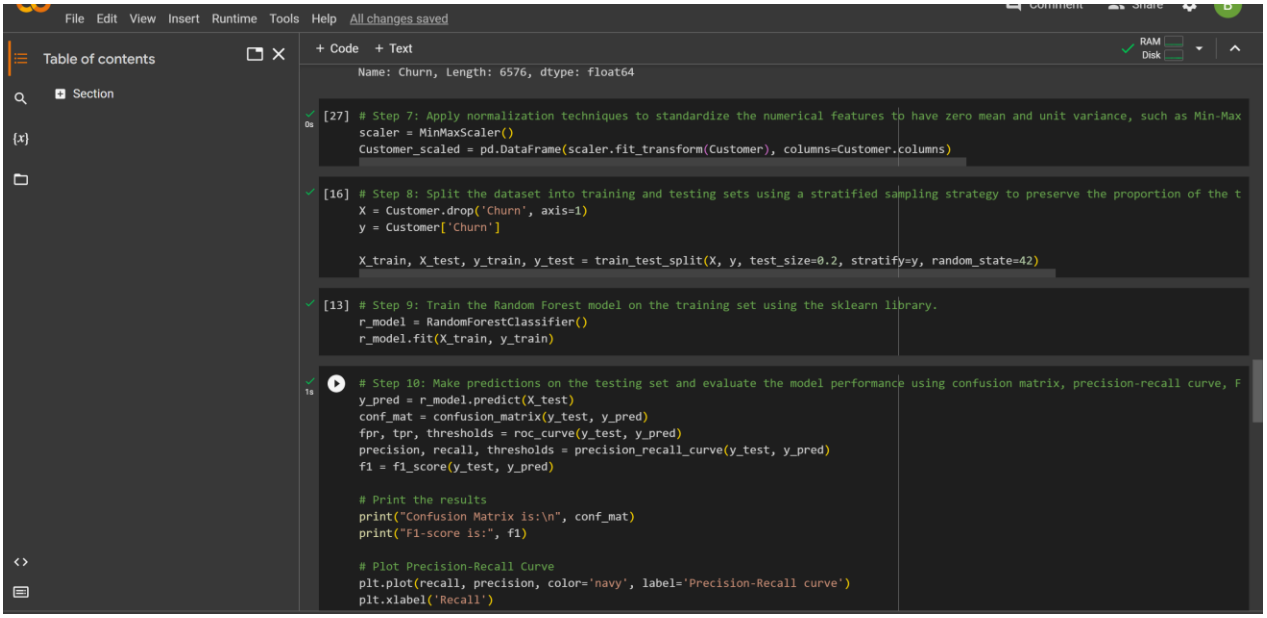

File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code + Text
RAM
Disk
[19] customerID 0
gender 0
SeniorCitizen 0
Partner 0
Dependents 0
tenure 0
PhoneService 0
MultipleLines 0
InternetService 0
OnlineSecurity 0
OnlineBackup 0
DeviceProtection 0
TechSupport 0
StreamingTV 0
StreamingMovies 0
Contract 0
PaperlessBilling 0
PaymentMethod 0
MonthlyCharges 0
TotalCharges 0
Churn 0
dtype: int64
customerID object
gender object
SeniorCitizen int64
Partner object
Dependents object
tenure int64
PhoneService object
MultipleLines object
InternetService object
OnlineSecurity object
OnlineBackup object
DeviceProtection object

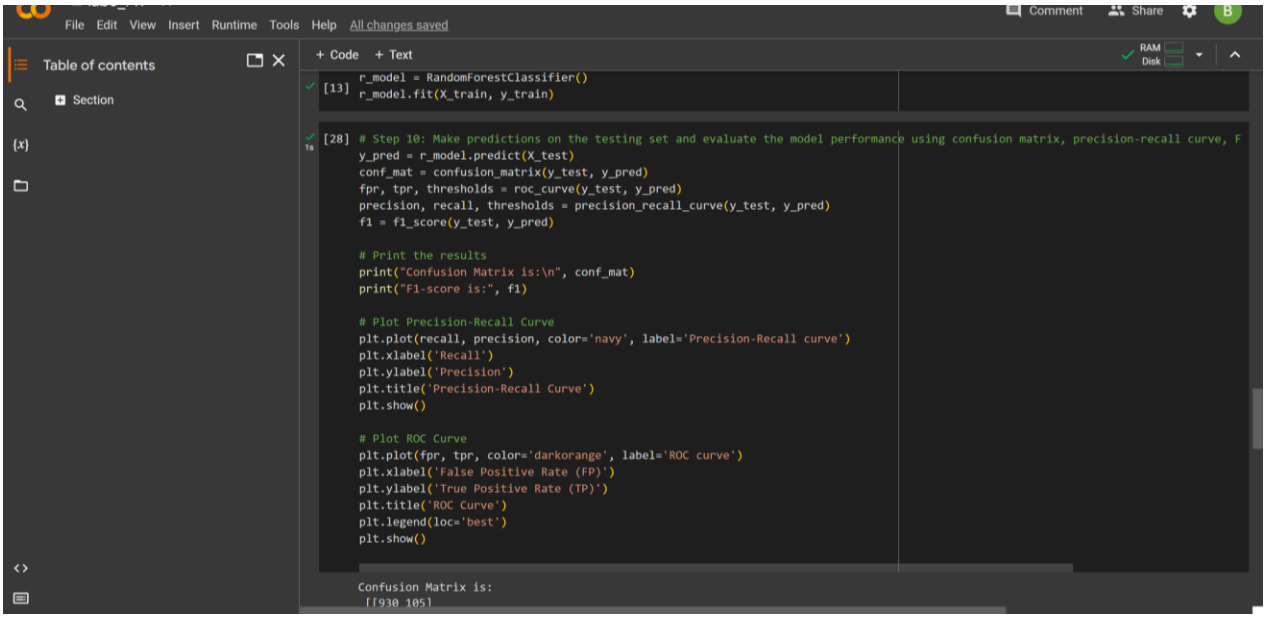

File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Section
+ Code
+ Text
RAM
Disk
Churn 0
[19] dtype: int64
customerID object
gender object
SeniorCitizen int64
Partner object
Dependents object
tenure int64
PhoneService object
MultipleLines object
InternetService object
OnlineSecurity object
OnlineBackup object
DeviceProtection object
TechSupport object
StreamingTV object
StreamingMovies object
Contract object
PaperlessBilling object
PaymentMethod object
MonthlyCharges float64
TotalCharges object
Churn object
dtype: object
SeniorCitizen tenure MonthlyCharges
count 7043.000000 7043.000000 7043.000000
mean 0.162147 32.371149 64.761692
std 0.368612 24.559481 30.090047
min 0.000000 0.000000 18.250000
25% 0.000000 9.000000 35.500000
50% 0.000000 29.000000 70.350000
75% 0.000000 55.000000 89.850000
max 1.000000 72.000000 118.750000
1s completed at 6:15 AM


File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code
+ Text
RAM
Disk
75% 0.000000 55.000000 89.850000
max 1.000000 72.000000 118.750000
[20] Customer.head()
customerID gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines InternetService OnlineSecurity ...
0 7590-VHVEG Female 0 Yes No 1 No No phone service DSL No ...
1 5575-GNVDE Male 0 No No 34 Yes No DSL Yes ...
2 3668-QPYBK Male 0 No No 2 Yes No DSL Yes ...
3 7795-CFOCW Male 0 No No 45 No No phone service DSL Yes ...
4 9237-HQITU Female 0 No No 2 Yes No Fiber optic No ...
5 rows × 21 columns
[21] Customer.shape
(7043, 21)
print(Customer.info())
<class 'pandas.core.frame.DataFrame'>


File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Section
+ Code
+ Text
RAM
Disk
(7043, 21)
print(Customer.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 customerID 7043 non-null object
1 gender 7043 non-null object
2 SeniorCitizen 7043 non-null int64
3 Partner 7043 non-null object
4 Dependents 7043 non-null object
5 tenure 7043 non-null int64
6 PhoneService 7043 non-null object
7 MultipleLines 7043 non-null object
8 InternetService 7043 non-null object
9 OnlineSecurity 7043 non-null object
10 OnlineBackup 7043 non-null object
11 DeviceProtection 7043 non-null object
12 TechSupport 7043 non-null object
13 StreamingTV 7043 non-null object
14 StreamingMovies 7043 non-null object
15 Contract 7043 non-null object
16 PaperlessBilling 7043 non-null object
17 PaymentMethod 7043 non-null object
18 MonthlyCharges 7043 non-null float64
19 TotalCharges 7043 non-null object
20 Churn 7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None


File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Section
+ Code
+ Text
RAM
Disk
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
[23] #3. Remove customer IDs from the data set
Customer.drop(['customerID'], axis=1, inplace=True)
[24] # Step 4: Convert the predictor variable in a binary numeric variable.
Customer['Churn'] = Customer['Churn'].replace({'Yes': 1, 'No': 0})
[25] # Step 5: Convert all the categorical variables into dummy variables.
Customer = pd.get_dummies(Customer)
[26] # Step 6: Show Correlation of "Churn" with other variables
matrix = Customer.corr()
print(matrix['Churn'].sort_values(ascending=False))
Churn 1.000000
Contract_Month-to-month 0.405103
OnlineSecurity_No 0.342637
TechSupport_No 0.337281
InternetService_Fiber optic 0.308020
...
DeviceProtection_No internet service -0.227890
OnlineBackup_No internet service -0.227890
StreamingMovies_No internet service -0.227890
Contract_Two year -0.302253
tenure -0.352229
Name: Churn, Length: 6576, dtype: float64


File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code + Text
RAM
Disk
Name: Churn, Length: 6576, dtype: float64
[27] # Step 7: Apply normalization techniques to standardize the numerical features to have zero mean and unit variance, such as Min-Max
scaler = MinMaxScaler()
Customer_scaled = pd.DataFrame(scaler.fit_transform(Customer), columns=Customer.columns)
[16] # Step 8: Split the dataset into training and testing sets using a stratified sampling strategy to preserve the proportion of the t
X = Customer.drop('Churn', axis=1)
y = Customer['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
[13] # Step 9: Train the Random Forest model on the training set using the sklearn library.
r_model = RandomForestClassifier()
r_model.fit(X_train, y_train)
# Step 10: Make predictions on the testing set and evaluate the model performance using confusion matrix, precision-recall curve, F
y_pred = r_model.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Print the results
print("Confusion Matrix is:\n", conf_mat)
print("F1-score is:", f1)
# Plot Precision-Recall Curve
plt.plot(recall, precision, color='navy', label='Precision-Recall curve')
plt.xlabel('Recall')


File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Table of contents
Section
+ Code + Text
RAM
Disk
[13] r_model = RandomForestClassifier()
r_model.fit(X_train, y_train)
[28] # Step 10: Make predictions on the testing set and evaluate the model performance using confusion matrix, precision-recall curve, F
y_pred = r_model.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Print the results
print("Confusion Matrix is:\n", conf_mat)
print("F1-score is:", f1)
# Plot Precision-Recall Curve
plt.plot(recall, precision, color='navy', label='Precision-Recall curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
# Plot ROC Curve
plt.plot(fpr, tpr, color='darkorange', label='ROC curve')
plt.xlabel('False Positive Rate (FP)')
plt.ylabel('True Positive Rate (TP)')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
Confusion Matrix is:
[[930 105]

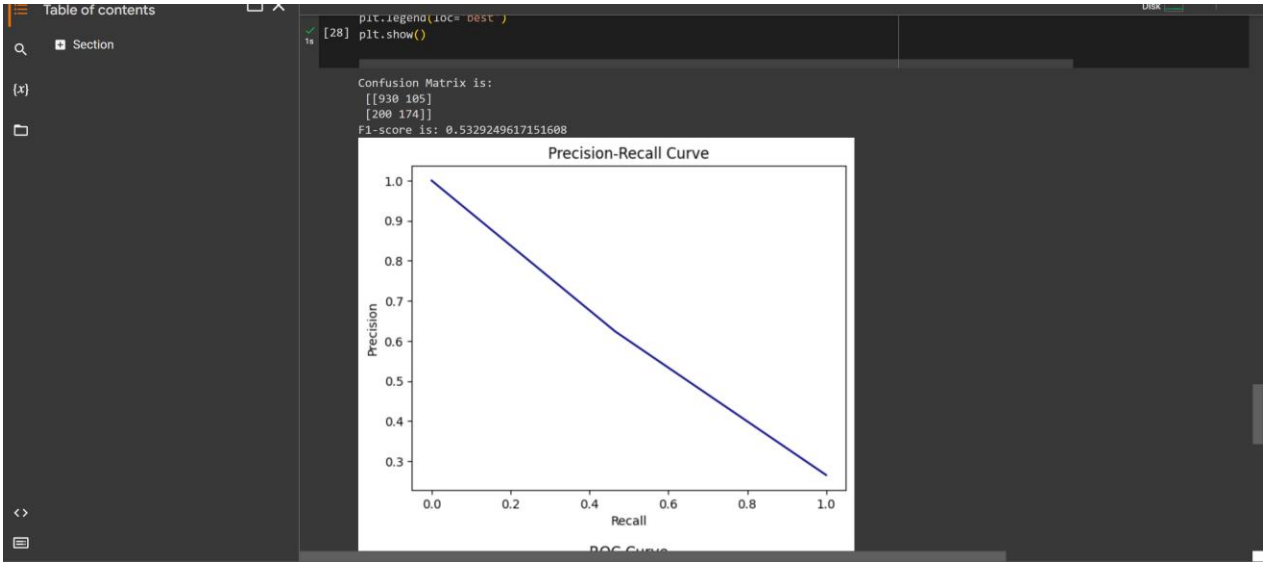
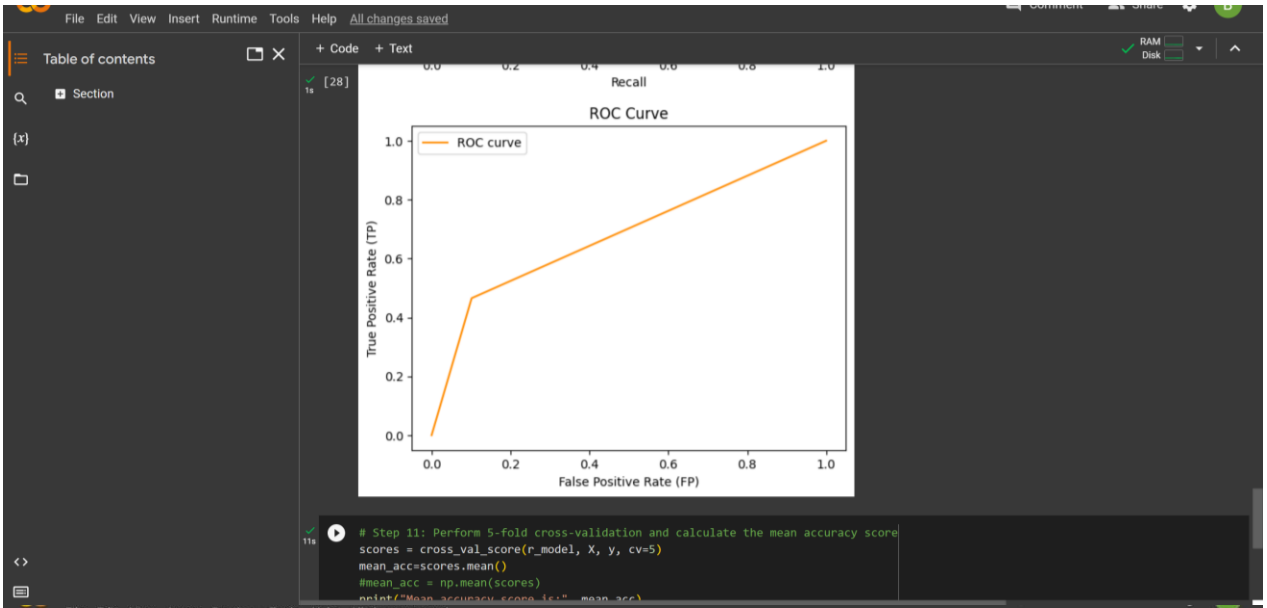
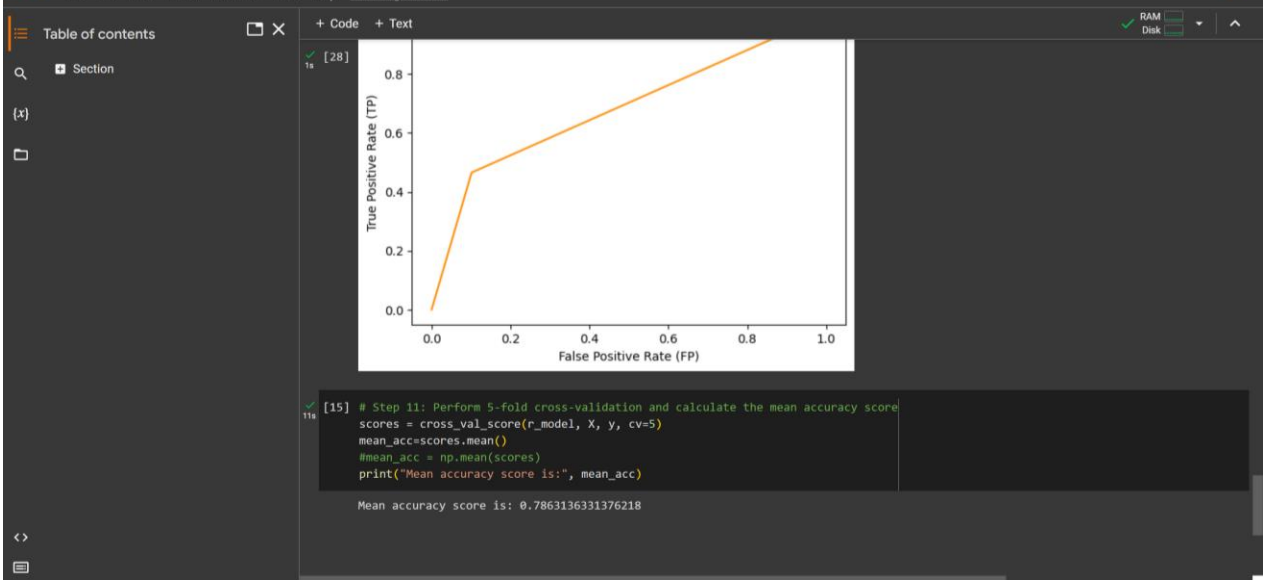


Table of contents
Section
plt.legend(loc='best')
[28] plt.show()
Confusion Matrix is:
[[930 105]
[200 174]]
F1-score is: 0.5329249617151608
Precision-Recall Curve
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0


File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code + Text
RAM
Disk
[28]
Recall
ROC Curve
ROC curve
True Positive Rate (TP)
False Positive Rate (FP)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
# Step 11: Perform 5-fold cross-validation and calculate the mean accuracy score
scores = cross_val_score(r_model, X, y, cv=5)
mean_acc=scores.mean()
#mean_acc = np.mean(scores)


File Edit View Insert Runtime Tools Help All changes saved
Table of contents
Section
+ Code + Text
RAM
Disk
[28]
True Positive Rate (TP)
False Positive Rate (FP)
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
[15] # Step 11: Perform 5-fold cross-validation and calculate the mean accuracy score
scores = cross_val_score(r_model, X, y, cv=5)
mean_acc=scores.mean()
#mean_acc = np.mean(scores)
print("Mean accuracy score is:", mean_acc)
Mean accuracy score is: 0.7863136331376218