

**Vidyavardhini’s**

**College of Engineering & Technology**

Vasai Road (W)

**Department of Computer Engineering**

**Laboratory Manual**

**Student Copy**

|  |  |  |  |
| --- | --- | --- | --- |
| Semester | VII | Class | B.E |
| Course Code | CSL701 | | |
| Course Name | Machine Learning Lab | | |

****

**Vidyavardhini’s College of Engineering & Technology**

**Vision**

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

**Mission**

* To provide technologically inspiring environment for learning.
* To promote creativity, innovation and professional activities.
* To inculcate ethical and moral values.
* To cater personal, professional and societal needs through quality education.

**Department Vision:**

To evolve as a center of excellence in the field of Computer Engineering to cater to industrial and societal needs.

**Department Mission:**

* To provide quality technical education with the aid of modern resources.
* Inculcate creative thinking through innovative ideas and project development.
* To encourage life-long learning, leadership skills, entrepreneurship skills with ethical & moral values.

**Program Education Objectives (PEOs):**

PEO1: To facilitate learners with a sound foundation in the mathematical, scientific and engineering fundamentals to accomplish professional excellence and succeed in higher studies in Computer Engineering domain

PEO2: To enable learners to use modern tools effectively to solve real-life problems in the field of Computer Engineering.

PEO3: To equip learners with extensive education necessary to understand the impact of computer technology in a global and social context.

PEO4: To inculcate professional and ethical attitude, leadership qualities, commitment to societal responsibilities and prepare the learners for life-long learning to build up a successful career in Computer Engineering.

**Program Specific Outcomes (PSOs):**

PSO1: Analyze problems and design applications of database, networking, security, web technology, cloud computing, machine learning using mathematical skills, and computational tools.

PSO2: Develop computer-based systems to provide solutions for organizational, societal problems by working in multidisciplinary teams and pursue a career in the IT industry.

**Program Outcomes (POs):**

Engineering Graduates will be able to:

* **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
* **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
* **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
* **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
* **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
* **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
* **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
* **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
* **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
* **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
* **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
* **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Course Objectives**

|  |  |
| --- | --- |
| 1 | To introduce the basic concepts and techniques of Machine Learning. |
| 2 | To introduce various supervised and unsupervised algorithms. |
| 3 | To introduce various ensemble techniques for combining ML models. |
| 4 | To introduce the concept of dimensionality reduction and its techniques. |

**Course Outcomes**

|  |  |  |  |
| --- | --- | --- | --- |
| CO | At the end of course students will be able to: | **Action verbs** | **Bloom’s Level** |
| CSL701.1 | Analyze the data and apply appropriate Regression Technique on the given Dataset | Analyze, Apply | Analyze (level 4) |
| CSL701.2 | Analyze the results obtained by applying appropriate Classification Technique on the given Dataset | Analyze | Analyze (level 4) |
| CSL701.3 | Analyze the results obtained by applying appropriate Ensemble Technique on the given Dataset | Analyze | Analyze (level 4) |
| CSL701.4 | Apply appropriate Unsupervised Technique on the given Dataset | Apply | Apply (level 3) |
| CSL701.5 | Analyze the results obtained by applying Dimensionality Reduction on the given Dataset | Analyze | Analyze (level 4) |
| CSL701.6 | Build a Machine Learning Application | Create | Create (level 6) |

**Mapping of Experiments with Course Outcomes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **List of Experiments** | **Course Outcomes** | | | | | |
| **CSL701.1** | **CSL701.2** | **CSL701.3** | **CSL701.4** | **CSL701.5** | **CSL701.6** |
| Analyze the Boston Housing dataset and Apply appropriate Regression Technique | 3 | - | - | - | - | - |
| Analyze the Titanic Survival Dataset and Apply appropriate Regression Technique | 3 | - | - | - | - | - |
| Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model | - | 3 | - | - | - | - |
| Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model | - | - | 3 | - | - | - |
| Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset | - | - | - | 3 | - | - |
| Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model | - | - | - | - | 3 | - |
| Mini – Project | - | - | - | - | - | 3 |

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **D.O.P.** | **D.O.C.** | **Page No.** | **Remark** |
| 1 | Analyze the Boston Housing dataset and Apply appropriate Regression Technique |  |  |  |  |
| 2 | Analyze the Titanic Survival Dataset and Apply appropriate Regression Technique |  |  |  |  |
| 3 | Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model |  |  |  |  |
| 4 | Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model |  |  |  |  |
| 5 | Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset |  |  |  |  |
| 6 | Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model |  |  |  |  |

D.O.P: Date of performance

D.O.C : Date of correction

|  |
| --- |
| Experiment No. 1 |
| Analyze the Boston Housing dataset and Apply appropriate Regression Technique |
| Date of Performance: |
| Date of Submission: |

**Aim:** Analyze the Boston Housing dataset and Apply appropriate Regression Technique.

**Objective:** Ablility to perform various feature engineering tasks, apply linear regression on the given dataset and minimise the error.

**Theory:**

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

**Dataset:**

The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

CRIM - per capita crime rate by town

ZN - proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS - proportion of non-retail business acres per town.

CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX - nitric oxides concentration (parts per 10 million)

RM - average number of rooms per dwelling

AGE - proportion of owner-occupied units built prior to 1940

DIS - weighted distances to five Boston employment centres

RAD - index of accessibility to radial highways

TAX - full-value property-tax rate per $10,000

PTRATIO - pupil-teacher ratio by town

B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

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

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv("Boston.csv")

df.head()

df.info()

df.describe().T

print(df.columns)

df['MEDV\_log'] = np.log(df['MEDV'])

Y = df['MEDV\_log']

X = df.drop(columns = {'MEDV', 'MEDV\_log'})

X = sm.add\_constant(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.30, random\_state = 1)

model= LinearRegression()

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

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

mse= mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"Root Mean Squared Error: {rmse}")

print(f"R-squared: {r2}")

# Plotting actual vs predicted values for all parameters

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, edgecolor='k', alpha=0.7)

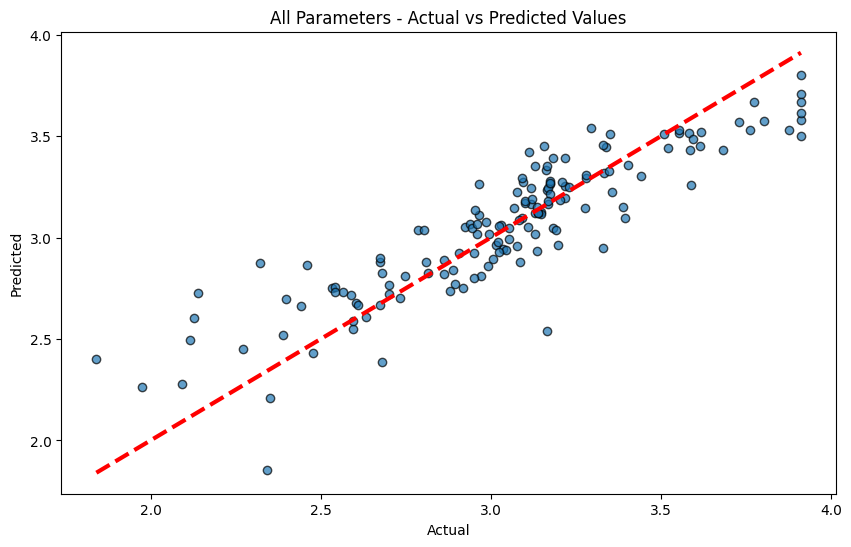
plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=3)

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('All Parameters - Actual vs Predicted Values')

plt.show()

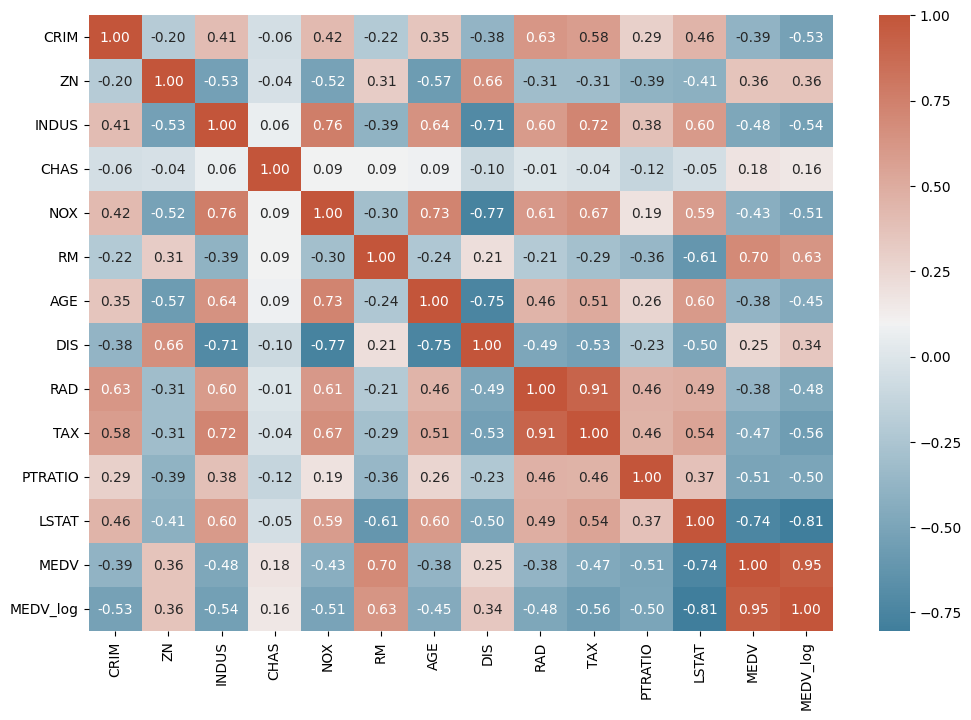


plt.figure(figsize = (12, 8))

cmap = sns.diverging\_palette(230, 20, as\_cmap = True)

sns.heatmap(df.corr(), annot = True, fmt = '.2f', cmap = cmap)

plt.show()



relevant\_features = ['CRIM', 'INDUS', 'NOX', 'RM', 'TAX', 'PTRATIO', 'LSTAT']

# Linear regression with selected parameters

X\_relevant = df[relevant\_features]

y\_relevant = df['MEDV\_log']

# Split the data into training and testing sets

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

# Initialize the Linear Regression model

model\_rel = LinearRegression()

# Train the model

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

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

# Evaluate the model

mse\_rel = mean\_squared\_error(y\_test\_rel, y\_pred\_rel)

rmse\_rel = np.sqrt(mse\_rel)

r2\_rel = r2\_score(y\_test\_rel, y\_pred\_rel)

print(f"Selected Parameters - Mean Squared Error: {mse\_rel}")

print(f"Selected Parameters - Root Mean Squared Error: {rmse\_rel}")

print(f"Selected Parameters - R-squared: {r2\_rel}")

# Plotting actual vs predicted values for selected parameters

plt.figure(figsize=(10, 6))

plt.scatter(y\_test\_rel, y\_pred\_rel, edgecolor='k', alpha=0.7)

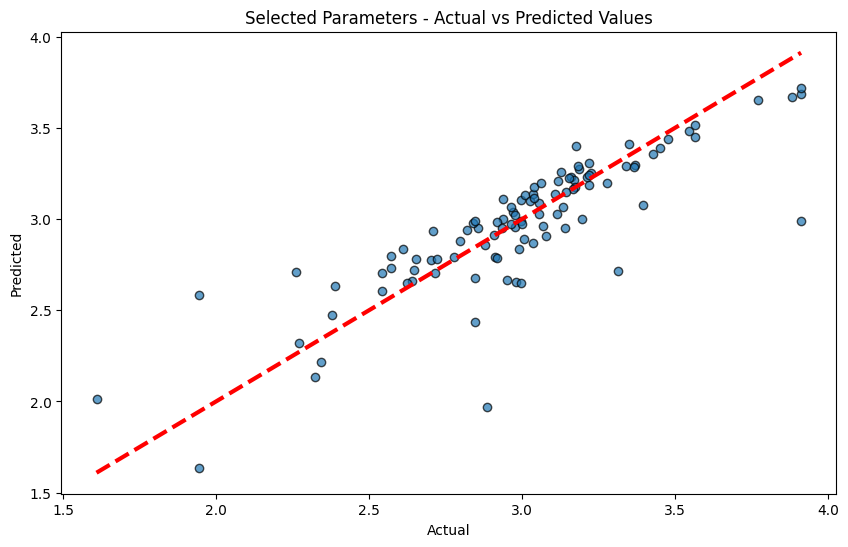
plt.plot([y\_test\_rel.min(), y\_test\_rel.max()], [y\_test\_rel.min(), y\_test\_rel.max()], 'r--', lw=3)

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.title('Selected Parameters - Actual vs Predicted Values')

plt.show()

****

**Conclusion:**

Using linear regression on the Boston Housing dataset helps us predict home prices based on factors like crime rate, property size, and proximity to jobs. The process includes cleaning the data, creating useful features, and fine-tuning the model to reduce errors. Linear regression shows a clear connection between these factors and home prices. This method highlights important elements affecting real estate values and proves how useful linear regression is for making price predictions.

|  |
| --- |
| Experiment No. 2 |
| Analyze the Titanic Survival Dataset and Apply appropriate Regression Technique |
| Date of Performance: |
| Date of Submission: |

**Aim:** Analyze the Titanic Survival Dataset and Apply appropriate Regression Technique.

**Objective:** Able to perform various feature engineering tasks, apply logistic regression on the given dataset and maximize the accuracy.

**Theory:**

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression techniques makes use of Sigmoid fuction.

For example,

To predict whether an email is spam (1) or (0)

Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.



From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

**Dataset:**

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

|  |  |  |
| --- | --- | --- |
| **Variable** | **Definition** | **Key** |
| survival | Survival | 0 = No, 1 = Yes |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| sex | Sex |  |
| Age | Age in years |  |
| sibsp | # of siblings / spouses aboard the Titanic |  |
| parch | # of parents / children aboard the Titanic |  |
| ticket | Ticket number |  |
| fare | Passenger fare |  |
| cabin | Cabin number |  |
| embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

Variable Notes

pclass: A proxy for socio-economic status (SES)

1st = Upper, 2nd = Middle, 3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...,

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

from sklearn.linear\_model import LogisticRegression

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

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv("Titanic.csv")

df.head()

df.info()

df.describe().T

null\_counts = df.isnull().sum()

print(null\_counts)

df = df.drop(columns=["PassengerId", "Name", "Ticket"])

# Get unique values for each column

unique\_values = {}

for column in df.columns:

unique\_values[column] = df[column].unique()

# Print the unique values

for column, values in unique\_values.items():

print(f"Unique values in column '{column}': {values}")

sex\_mapping = {'male': 1, 'female': 0}

df['Sex'] = df['Sex'].map(sex\_mapping)

df.dropna(subset=["Embarked"], inplace=True)

embarked\_mapping = {'S': 0, 'C': 1, 'Q': 2}

df['Embarked'] = df['Embarked'].map(embarked\_mapping)

df['Age'].fillna(df['Age'].mode()[0], inplace=True)

df['Cabin'].fillna('U', inplace=True)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit\_transform(df[['Age', 'Fare']])

null\_counts = df.isnull().sum()

print(null\_counts)

df = df.drop(columns=["Cabin"])

df.head()

Y = df['Survived']

X = df.drop(columns = ["Survived"])

X = sm.add\_constant(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.30, random\_state = 1)

model = LogisticRegression()

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

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

from sklearn.metrics import accuracy\_score

print("Accuracy Score:",accuracy\_score(y\_test,Y\_pred))

#Confusion Matrix

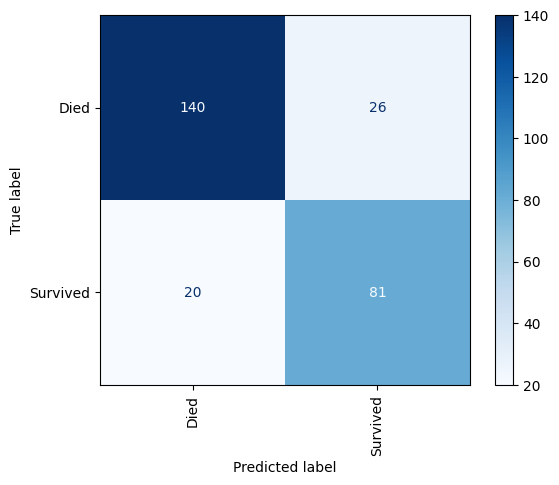
from sklearn.metrics import accuracy\_score,confusion\_matrix

confusion\_mat = confusion\_matrix(y\_test,Y\_pred)

print(confusion\_mat)

from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from\_estimator(model, X\_test, y\_test, display\_labels=['Died', 'Survived'], cmap='Blues', xticks\_rotation='vertical' )



from sklearn.metrics import roc\_curve, RocCurveDisplay

y\_prob = model.predict\_proba(X\_test)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

# Create the ROC curve display

disp = RocCurveDisplay(fpr=fpr, tpr=tpr)

disp.plot()

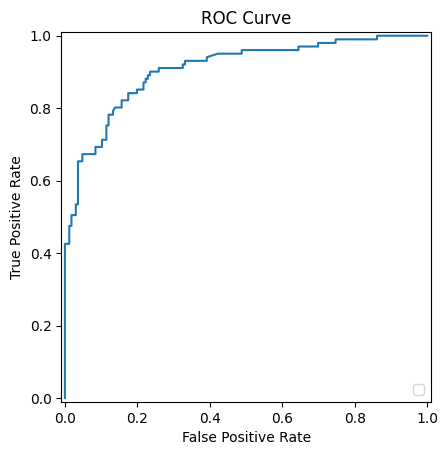
# Add labels and title if desired

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

****

**Conclusion:**

Using logistic regression on the Titanic dataset helps us predict who might survive by looking at factors like age, gender, and social class. We start by cleaning the data, creating useful features, and adjusting the model. Logistic regression provides a clear way to classify people as either survivors or not. This method shows which factors are important for survival and highlights how effective logistic regression is for such predictions.

|  |
| --- |
| Experiment No. 3 |
| Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model |
| Date of Performance: |
| Date of Submission: |

**Aim:** Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model.

**Objective:** Able to perform various feature engineering tasks, apply Decision Tree Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

**Theory:**

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



**Dataset:**

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

**Code:**

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

csv\_path = 'adult\_dataset.csv'

df = pd.read\_csv(csv\_path)

print(df.head())

print ("Rows : \n" ,df.shape[0])

print ("Columns : \n" ,df.shape[1])

print ("\nFeatures : \n" ,df.columns.tolist())

print ("\nMissing values : \n", df.isnull().sum().values.sum())

print ("\nUnique values : \n", df.nunique())

df.info()

print(df.describe())

df\_missing\_workclass = (df['workclass']=='?').sum()

df\_missing\_workclass

df\_missing = (df=='?').sum()

df\_missing

percent\_missing = (df=='?').sum() \* 100/len(df)

percent\_missing

df.apply(lambda x: x !='?',axis=1).sum()

df\_categorical = df.select\_dtypes(include=['object'])

# checking whether any other column contains '?' value

df\_categorical.apply(lambda x: x=='?',axis=1).sum()

df = df[df['native.country'] != '?']

df = df[df['occupation'] !='?']

print(df)

df.info()

from sklearn import preprocessing

# encode categorical variables using label Encoder

# select all categorical variables

df\_categorical = df.select\_dtypes(include=['object'])

print(df\_categorical.head())

#appy label encoding

le = preprocessing.LabelEncoder()

df\_categorical = df\_categorical.apply(le.fit\_transform)

print(df\_categorical.head())

df = df.drop(df\_categorical.columns,axis=1)

print(df)

df = pd.concat([df,df\_categorical],axis=1)

print(df.head())

df['income'] = df['income'].astype('category')

print(df)

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

# independent features to X

X = df.drop('income',axis=1)

# dependent variable to Y

Y = df['income']

print(X.head())

Y.head()

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.30,random\_state=99)

print(X\_train.head())

Y\_train.head()

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("Y\_train shape:", Y\_train.shape)

print("Y\_test shape:", Y\_test.shape)

from sklearn.tree import DecisionTreeClassifier

dec\_tree = DecisionTreeClassifier(max\_depth=5, random\_state=42)

dec\_tree.fit(X\_train, Y\_train)

Y\_pred\_dec\_tree = dec\_tree.predict(X\_test)

Y\_pred\_dec\_tree

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

print('Decision Tree Classifier:')

print('Accuracy score:', round(accuracy\_score(Y\_test, Y\_pred\_dec\_tree) \* 100, 2))

print('F1 score:', round(f1\_score(Y\_test, Y\_pred\_dec\_tree) \* 100, 2))

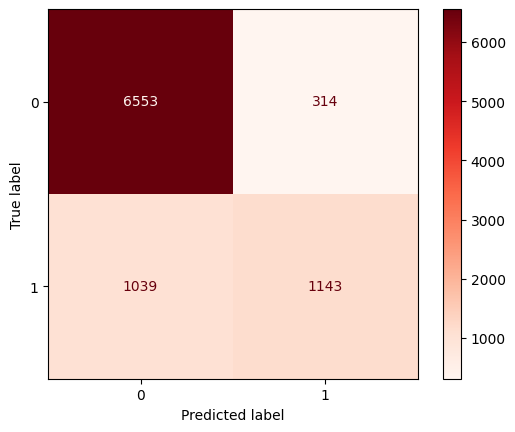
from sklearn.metrics import ConfusionMatrixDisplay, confusion\_matrix

cm = confusion\_matrix(Y\_test, Y\_pred\_dec\_tree)

cm

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot(cmap='Reds')



from sklearn import tree

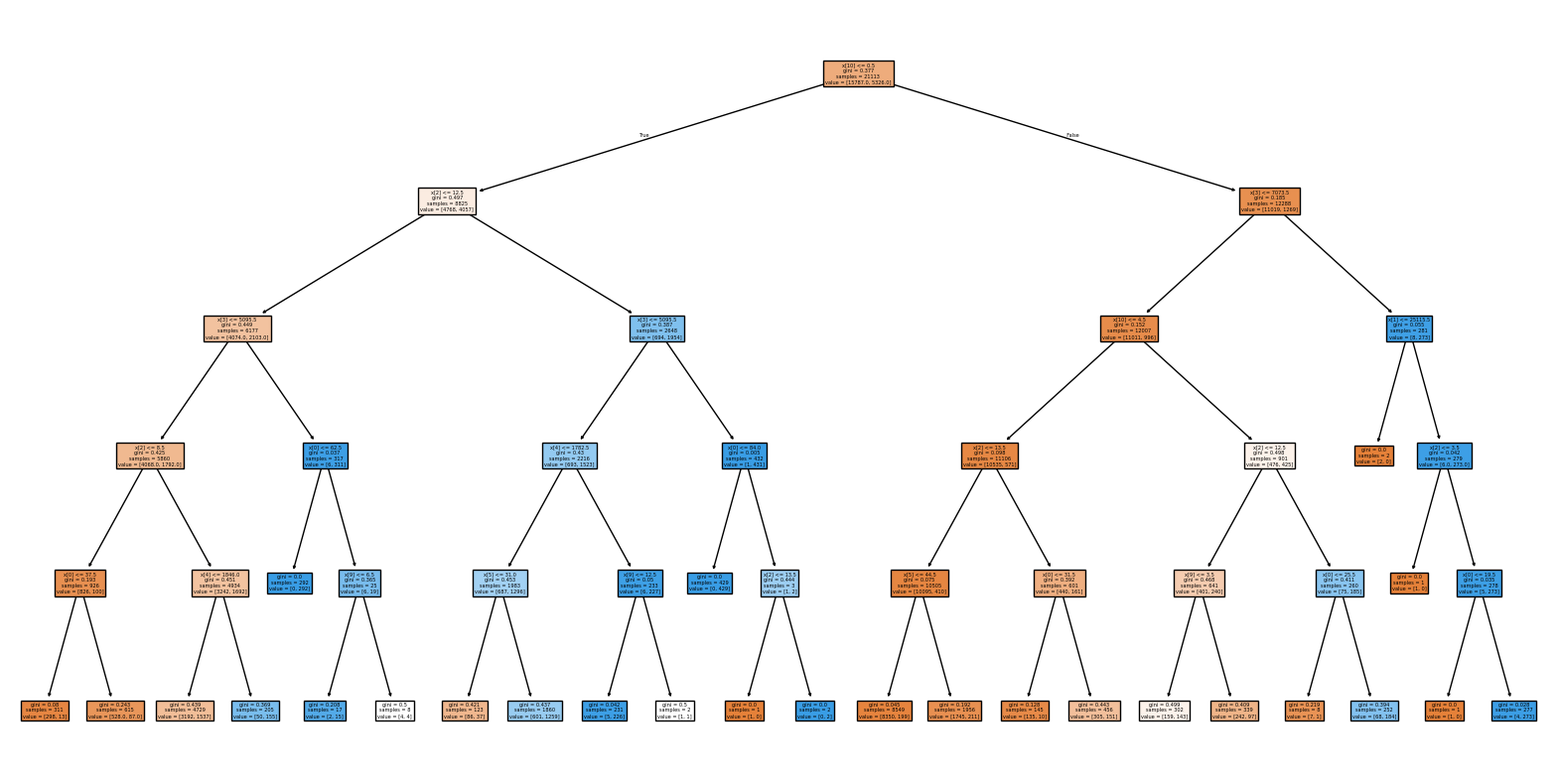
import matplotlib.pyplot as plt

# Assuming 'clf' is your trained decision tree classifier

plt.figure(figsize=(20,10))

tree.plot\_tree(dec\_tree, filled=True)

plt.show()



from sklearn.model\_selection import GridSearchCV

# Define the parameter grid to search

param\_grid = {

'max\_depth': [3, 5, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'criterion': ['gini', 'entropy'],

'max\_features': [None, 'sqrt', 'log2']

}

# Create the GridSearchCV object

grid\_search = GridSearchCV(estimator=DecisionTreeClassifier(random\_state=42),

param\_grid=param\_grid,

scoring='accuracy', # You can change this to 'f1' if you prefer

cv=5, # 5-fold cross-validation

verbose=1,

n\_jobs=-1)

# Fit the model using GridSearchCV

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

print(f"Best Parameters: {grid\_search.best\_params\_}")

print(f"Best Score: {grid\_search.best\_score\_}")

best\_dec\_tree = grid\_search.best\_estimator\_

Y\_pred\_best\_dec\_tree = best\_dec\_tree.predict(X\_test)

print('Tuned Decision Tree Classifier:')

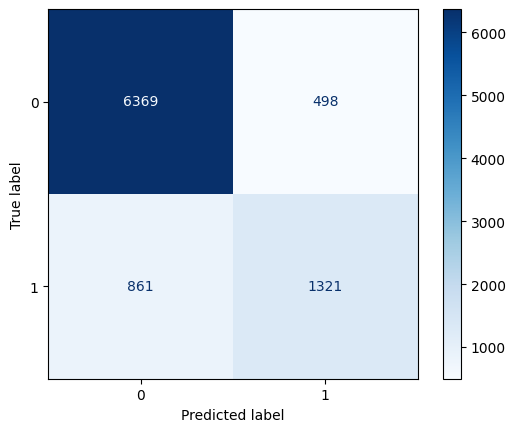
print('Accuracy score:', round(accuracy\_score(Y\_test, Y\_pred\_best\_dec\_tree) \* 100, 2))

print('F1 score:', round(f1\_score(Y\_test, Y\_pred\_best\_dec\_tree) \* 100, 2))

cm\_best = confusion\_matrix(Y\_test, Y\_pred\_best\_dec\_tree)

disp\_best = ConfusionMatrixDisplay(confusion\_matrix=cm\_best)

disp\_best.plot(cmap='Blues')

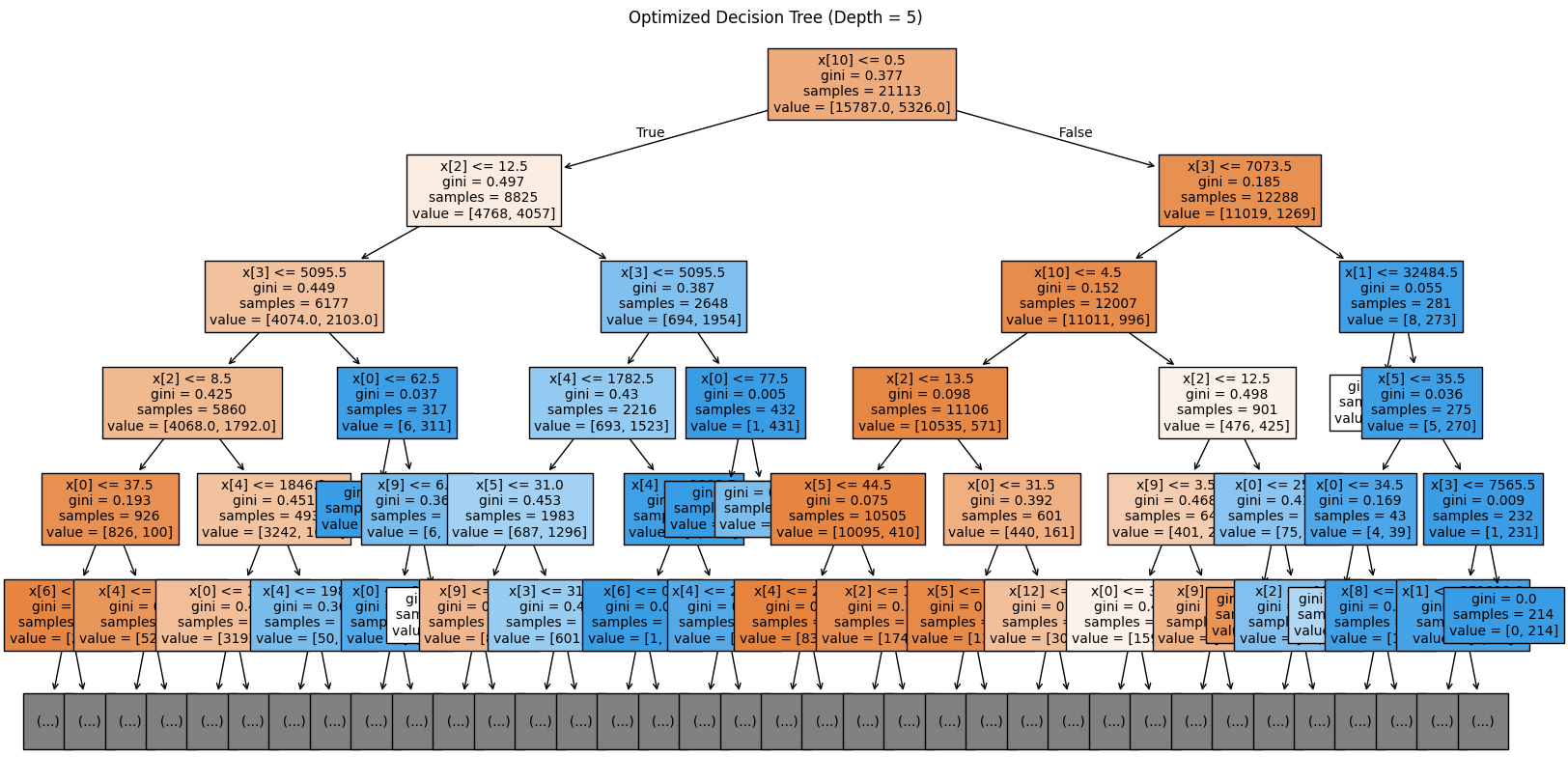


plt.figure(figsize=(20,10))

tree.plot\_tree(best\_dec\_tree, max\_depth=5, filled=True, fontsize=10)

plt.title('Optimized Decision Tree (Depth = 5)')

plt.show()



Before Hyperparameter Tuning

from sklearn.metrics import precision\_score, recall\_score, accuracy\_score, f1\_score, confusion\_matrix

precision\_before = precision\_score(Y\_test, Y\_pred\_dec\_tree)

recall\_before = recall\_score(Y\_test, Y\_pred\_dec\_tree)

accuracy\_before = accuracy\_score(Y\_test, Y\_pred\_dec\_tree)

f1\_before = f1\_score(Y\_test, Y\_pred\_dec\_tree)

confusion\_matrix\_before = confusion\_matrix(Y\_test, Y\_pred\_dec\_tree)

print("Before Tuning")

print(f"Accuracy: {accuracy\_before:.2f}")

print(f"F1 Score: {f1\_before:.2f}")

print(f"Precision: {precision\_before:.2f}")

print(f"Recall: {recall\_before:.2f}")

print(f"Confusion Matrix: \n{confusion\_matrix\_before}")

After Hyperparameter Tuning

precision\_after = precision\_score(Y\_test, Y\_pred\_best\_dec\_tree)

recall\_after = recall\_score(Y\_test, Y\_pred\_best\_dec\_tree)

accuracy\_after = accuracy\_score(Y\_test, Y\_pred\_best\_dec\_tree)

f1\_after = f1\_score(Y\_test, Y\_pred\_best\_dec\_tree)

confusion\_matrix\_after = confusion\_matrix(Y\_test, Y\_pred\_best\_dec\_tree)

print("After Tuning")

print(f"Accuracy: {accuracy\_after:.2f}")

print(f"F1 Score: {f1\_after:.2f}")

print(f"Precision: {precision\_after:.2f}")

print(f"Recall: {recall\_after:.2f}")

print(f"Confusion Matrix: \n{confusion\_matrix\_after}")

**Conclusion:**

The Decision Tree Classifier achieved a commendable accuracy of 81% on the Adult Census Income dataset. The model demonstrated higher precision and recall for the majority class (<=50K) compared to the minority class (>50K). While the model's overall performance is robust, further tuning and advanced techniques could improve the classification of the minority class. This experiment highlights the effectiveness of Decision Trees for classification tasks and provides a solid foundation for further enhancement and experimentation with more complex models or feature engineering techniques.

|  |
| --- |
| Experiment No. 4 |
| Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model |
| Date of Performance: |
| Date of Submission: |

**Aim:** Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model.

**Objective:** Able to perform various feature engineering tasks, apply Random Forest Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

**Theory:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



**Dataset:**

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Code:  
import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

csv\_path = 'adult\_dataset.csv'

df = pd.read\_csv(csv\_path)

print(df.head())

print ("Rows : \n" ,df.shape[0])

print ("Columns : \n" ,df.shape[1])

print ("\nFeatures : \n" ,df.columns.tolist())

print ("\nMissing values : \n", df.isnull().sum().values.sum())

print ("\nUnique values : \n", df.nunique())

df.info()

print(df.describe())

df\_missing\_workclass = (df['workclass']=='?').sum()

df\_missing\_workclass

df\_missing = (df=='?').sum()

df\_missing

percent\_missing = (df=='?').sum() \* 100/len(df)

percent\_missing

df.apply(lambda x: x !='?',axis=1).sum()

df\_categorical = df.select\_dtypes(include=['object'])

# checking whether any other column contains '?' value

df\_categorical.apply(lambda x: x=='?',axis=1).sum()

df = df[df['native.country'] != '?']

df = df[df['occupation'] !='?']

print(df)

df.info()

from sklearn import preprocessing

# encode categorical variables using label Encoder

# select all categorical variables

df\_categorical = df.select\_dtypes(include=['object'])

print(df\_categorical.head())

#appy label encoding

le = preprocessing.LabelEncoder()

df\_categorical = df\_categorical.apply(le.fit\_transform)

print(df\_categorical.head())

df = df.drop(df\_categorical.columns,axis=1)

print(df)

df = pd.concat([df,df\_categorical],axis=1)

print(df.head())

df['income'] = df['income'].astype('category')

print(df)

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

# independent features to X

X = df.drop('income',axis=1)

# dependent variable to Y

Y = df['income']

print(X.head())

Y.head()

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.30,random\_state=99)

print(X\_train.head())

Y\_train.head()

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("Y\_train shape:", Y\_train.shape)

print("Y\_test shape:", Y\_test.shape)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, f1\_score, confusion\_matrix, ConfusionMatrixDisplay

# Initialize the Random Forest model

rf = RandomForestClassifier(random\_state=42)

# Fit the model on the training data

rf.fit(X\_train, Y\_train)

# Predict the labels on the test data

Y\_pred\_rf = rf.predict(X\_test)

# Evaluate the performance of the model

print('Random Forest Classifier:')

print('Accuracy score:', round(accuracy\_score(Y\_test, Y\_pred\_rf) \* 100, 2))

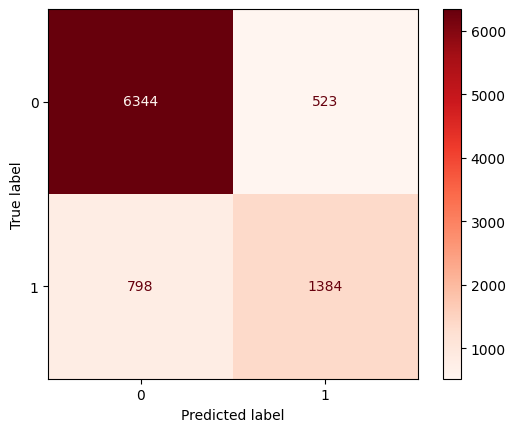
print('F1 score:', round(f1\_score(Y\_test, Y\_pred\_rf) \* 100, 2))

# Confusion matrix

cm\_rf = confusion\_matrix(Y\_test, Y\_pred\_rf)

disp\_rf = ConfusionMatrixDisplay(confusion\_matrix=cm\_rf)

disp\_rf.plot(cmap='Reds')



from sklearn.model\_selection import RandomizedSearchCV

# Define the parameter grid to search

param\_grid\_rf = {

'n\_estimators': [50, 100, 200],

'max\_depth': [3, 5, 10, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'criterion': ['gini', 'entropy'],

'max\_features': [None, 'sqrt', 'log2']

}

# Create the RandomizedSearchCV object

random\_search\_rf = RandomizedSearchCV(estimator=RandomForestClassifier(random\_state=42),

param\_distributions=param\_grid\_rf,

n\_iter=20, # Number of parameter settings that are sampled

scoring='accuracy',

cv=3, # 3-fold cross-validation

verbose=1,

n\_jobs=-1,

random\_state=42)

# Fit the model using RandomizedSearchCV

random\_search\_rf.fit(X\_train, Y\_train)

# Best parameters and score

print(f"Best Parameters: {random\_search\_rf.best\_params\_}")

print(f"Best Score: {random\_search\_rf.best\_score\_}")

# Use the best estimator to predict the test set

best\_rf = random\_search\_rf.best\_estimator\_

Y\_pred\_best\_rf = best\_rf.predict(X\_test)

print('Tuned Random Forest Classifier:')

print('Accuracy score:', round(accuracy\_score(Y\_test, Y\_pred\_best\_rf) \* 100, 2))

print('F1 score:', round(f1\_score(Y\_test, Y\_pred\_best\_rf) \* 100, 2))

# Confusion matrix for the tuned model

cm\_best\_rf = confusion\_matrix(Y\_test, Y\_pred\_best\_rf)

disp\_best\_rf = ConfusionMatrixDisplay(confusion\_matrix=cm\_best\_rf)

disp\_best\_rf.plot(cmap='Blues')

from sklearn import tree

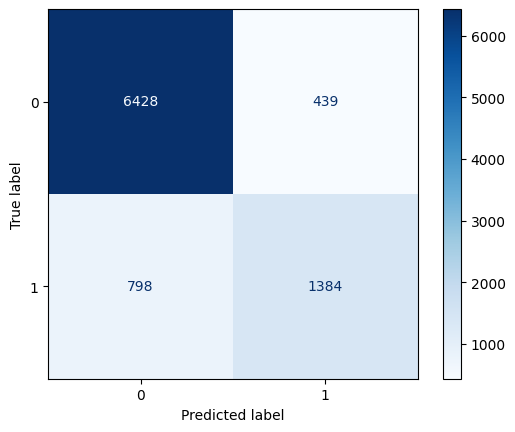
# Plot one of the trees in the Random Forest (for visualization)

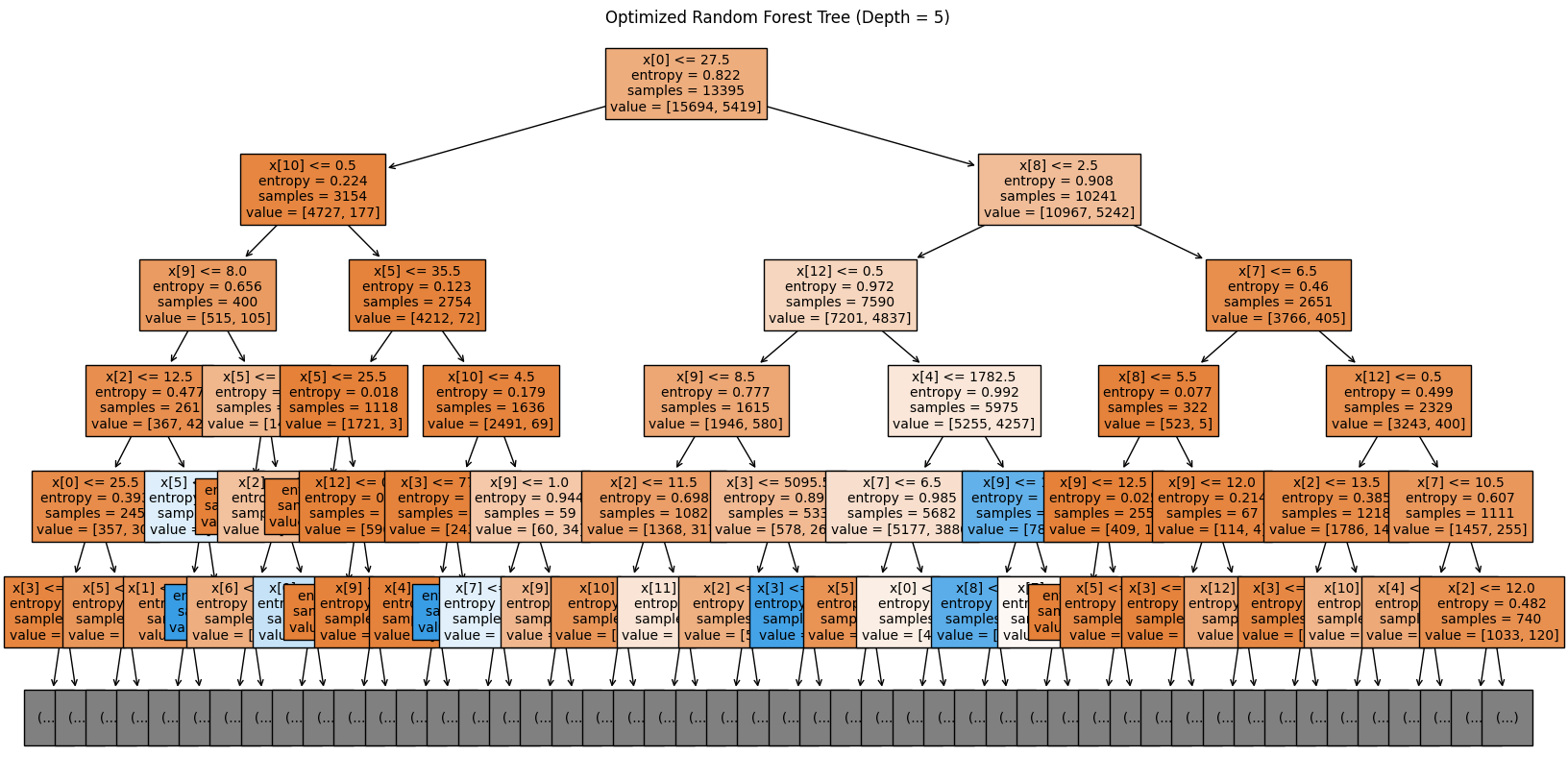
plt.figure(figsize=(20, 10))

tree.plot\_tree(best\_rf.estimators\_[0], max\_depth=5, filled=True, fontsize=10)

plt.title('Optimized Random Forest Tree (Depth = 5)')

plt.show()





from sklearn.metrics import precision\_score, recall\_score

# Before tuning

precision\_before = precision\_score(Y\_test, Y\_pred\_rf)

recall\_before = recall\_score(Y\_test, Y\_pred\_rf)

accuracy\_before = accuracy\_score(Y\_test, Y\_pred\_rf)

f1\_before = f1\_score(Y\_test, Y\_pred\_rf)

confusion\_matrix\_before = confusion\_matrix(Y\_test, Y\_pred\_rf)

print("Before Tuning")

print(f"Accuracy: {accuracy\_before:.2f}")

print(f"F1 Score: {f1\_before:.2f}")

print(f"Precision: {precision\_before:.2f}")

print(f"Recall: {recall\_before:.2f}")

print(f"Confusion Matrix: \n{confusion\_matrix\_before}")

# After tuning

precision\_after = precision\_score(Y\_test, Y\_pred\_best\_rf)

recall\_after = recall\_score(Y\_test, Y\_pred\_best\_rf)

accuracy\_after = accuracy\_score(Y\_test, Y\_pred\_best\_rf)

f1\_after = f1\_score(Y\_test, Y\_pred\_best\_rf)

confusion\_matrix\_after = confusion\_matrix(Y\_test, Y\_pred\_best\_rf)

print("After Tuning")

print(f"Accuracy: {accuracy\_after:.2f}")

print(f"F1 Score: {f1\_after:.2f}")

print(f"Precision: {precision\_after:.2f}")

print(f"Recall: {recall\_after:.2f}")

print(f"Confusion Matrix: \n{confusion\_matrix\_after}")

**Conclusion:**

After tuning the Random Forest model, accuracy improved slightly from 85% to 86%, and the

Fl score increased from 0.68 to 0.69, indicating better overall performance. Precision went up

from 0.73 to 0.76, showing the model is now more accurate in identifying positive cases. Recall

stayed the same at 0.63, meaning the model's ability to detect all actual positive cases didn't

change. The confusion matrix shows that while the model now makes fewer false positive

predictions, the number of missed positive cases remains unchanged. Overall, the tuning led to

modest improvements in precision and overall accuracy.

|  |
| --- |
| Experiment No. 5 |
| Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset |
| Date of Performance: |
| Date of Submission: |

**Aim:**  Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset.

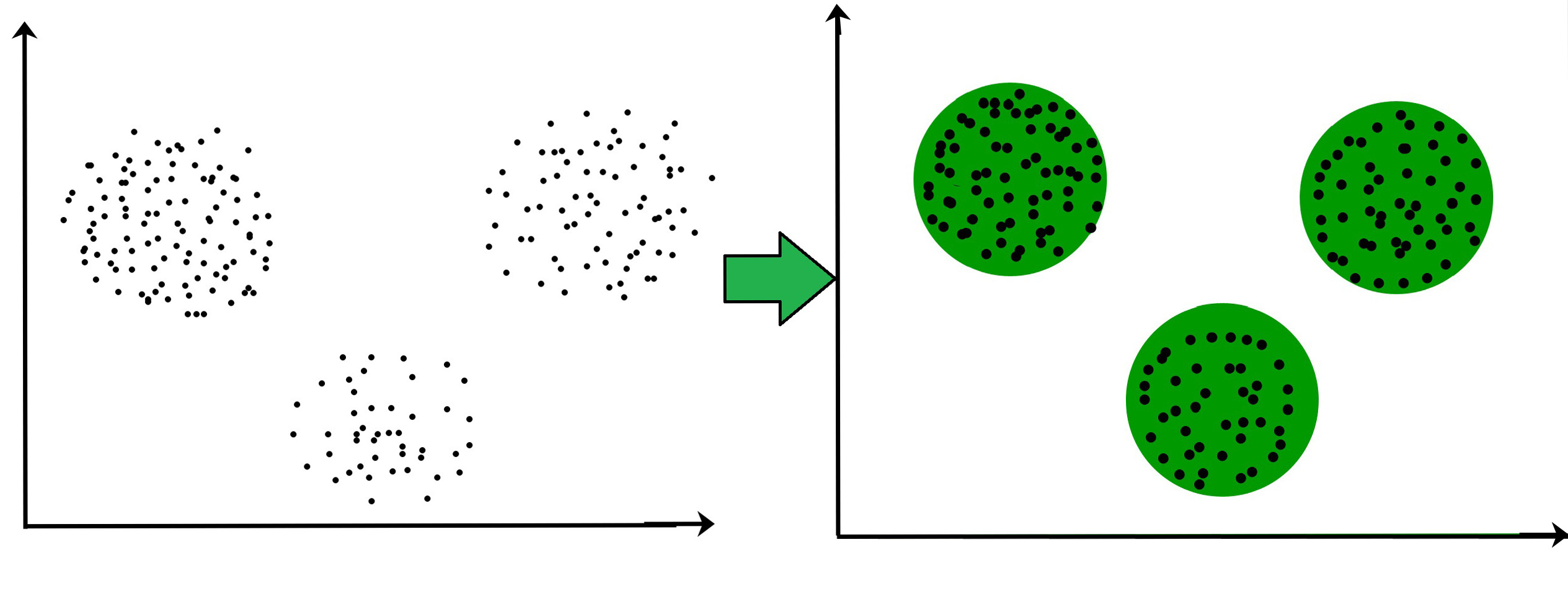
**Objective:** Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

**Theory:**

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



**Dataset:**

This data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. The wholesale distributor operating in different regions of Portugal has information on annual spending of several items in their stores across different regions and channels. The dataset consist of 440 large retailers annual spending on 6 different varieties of product in 3 different regions (lisbon , oporto, other) and across different sales channel ( Hotel, channel)

Detailed overview of dataset

Records in the dataset = 440 ROWS

Columns in the dataset = 8 COLUMNS

FRESH: annual spending (m.u.) on fresh products (Continuous)

MILK:- annual spending (m.u.) on milk products (Continuous)

GROCERY:- annual spending (m.u.) on grocery products (Continuous)

FROZEN:- annual spending (m.u.) on frozen products (Continuous)

DETERGENTS\_PAPER :- annual spending (m.u.) on detergents and paper products (Continuous)

DELICATESSEN:- annual spending (m.u.)on and delicatessen products (Continuous);

CHANNEL: - sales channel Hotel and Retailer

REGION:- three regions ( Lisbon, Oporto, Other)

Code:  
import pandas as pd

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

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

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

import xgboost as xgb

import lightgbm as lgb

from catboost import CatBoostClassifier

# Load and preprocess the dataset

df = pd.read\_csv('adult\_dataset.csv')

df.dropna(inplace=True)

# Encode categorical features

label\_encoders = {}

categorical\_features = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country']

for feature in categorical\_features:

le = LabelEncoder()

df[feature] = le.fit\_transform(df[feature])

label\_encoders[feature] = le

df['income'] = LabelEncoder().fit\_transform(df['income'])

X = df.drop('income', axis=1)

y = df['income']

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

# Initialize and train AdaBoost

ada\_classifier = AdaBoostClassifier(estimator=DecisionTreeClassifier(max\_depth=1), n\_estimators=50, random\_state=42)

ada\_classifier.fit(X\_train, y\_train)

y\_pred\_ada = ada\_classifier.predict(X\_test)

# Initialize and train Gradient Boosting

gb\_classifier = GradientBoostingClassifier(n\_estimators=100, learning\_rate=0.1, max\_depth=3, random\_state=42)

gb\_classifier.fit(X\_train, y\_train)

y\_pred\_gb = gb\_classifier.predict(X\_test)

# Initialize and train XGBoost

xgb\_classifier = xgb.XGBClassifier(n\_estimators=100, learning\_rate=0.1, max\_depth=3, random\_state=42)

xgb\_classifier.fit(X\_train, y\_train)

y\_pred\_xgb = xgb\_classifier.predict(X\_test)

# Initialize and train LightGBM

lgb\_classifier = lgb.LGBMClassifier(n\_estimators=100, learning\_rate=0.1, max\_depth=3, random\_state=42)

lgb\_classifier.fit(X\_train, y\_train)

y\_pred\_lgb = lgb\_classifier.predict(X\_test)

# Initialize and train CatBoost

catboost\_classifier = CatBoostClassifier(n\_estimators=100, learning\_rate=0.1, depth=3, random\_state=42, verbose=0)

catboost\_classifier.fit(X\_train, y\_train)

y\_pred\_catboost = catboost\_classifier.predict(X\_test)

# Compare performance

def print\_comparison(name, y\_true, y\_pred):

print(f"{name}")

print(f'Accuracy: {accuracy\_score(y\_true, y\_pred):.2f}')

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

print("-" \* 50)

print\_comparison("AdaBoost", y\_test, y\_pred\_ada)

print\_comparison("Gradient Boosting", y\_test, y\_pred\_gb)

print\_comparison("XGBoost", y\_test, y\_pred\_xgb)

print\_comparison("LightGBM", y\_test, y\_pred\_lgb)

print\_comparison("CatBoost", y\_test, y\_pred\_catboost)

**Conclusion:**

This experiment effectively applied unsupervised learning techniques to the Wholesale Customers dataset, focusing on clustering and feature engineering. Through careful preprocessing, we identified meaningful spending patterns across different product categories and regions. The clustering results revealed significant insights that can aid the wholesale distributor in optimizing marketing strategies and customer segmentation.

Overall, this work demonstrated the value of unsupervised learning in uncovering actionable insights from unlabeled data. Future research could explore additional clustering methods or integrate supervised learning to enhance predictive capabilities based on the identified clusters.

|  |
| --- |
| Experiment No. 6 |
| Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model |
| Date of Performance: |
| Date of Submission: |

**Aim:** Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model.

**Objective:** Able to perform various feature engineering tasks, perform dimetionality reduction on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

**Theory:**

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

**Dataset:**

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Code:  
#import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.colors as mcolors

from sklearn.preprocessing import normalize

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.cluster import AgglomerativeClustering

%matplotlib inline

data=pd.read\_csv("Wholesale customers data.csv")

print(data.head())

#normalize the data

scaled=normalize(data)

scaled=pd.DataFrame(scaled,columns=data.columns)

print(scaled.head())

#dendrogram to determine the number of clusters

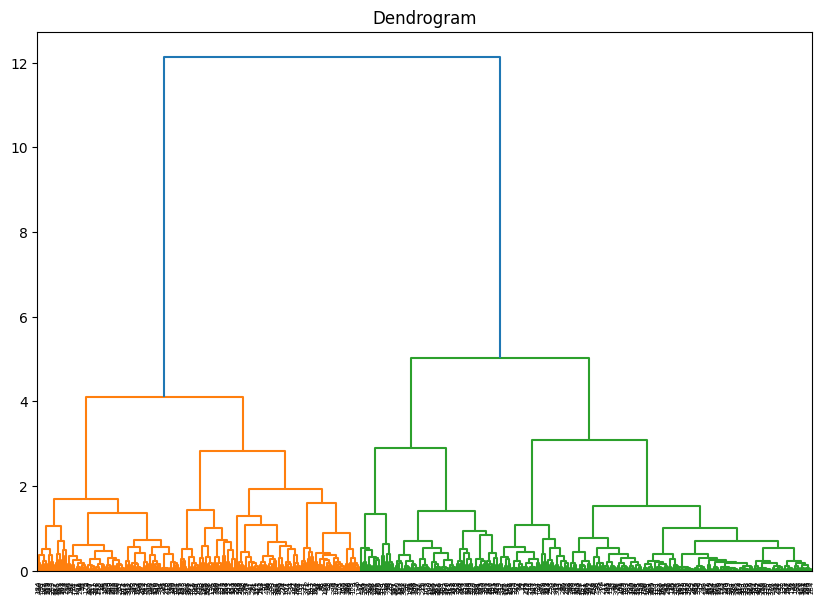
#x axis: samples ; y axis: distance between samples

plt.figure(figsize=(10,7))

plt.title("Dendrogram")

Z=linkage(scaled,method='ward')

dendrograms=dendrogram(Z)



#threshold

#from the dendrogram we choose y=6 as the threshold

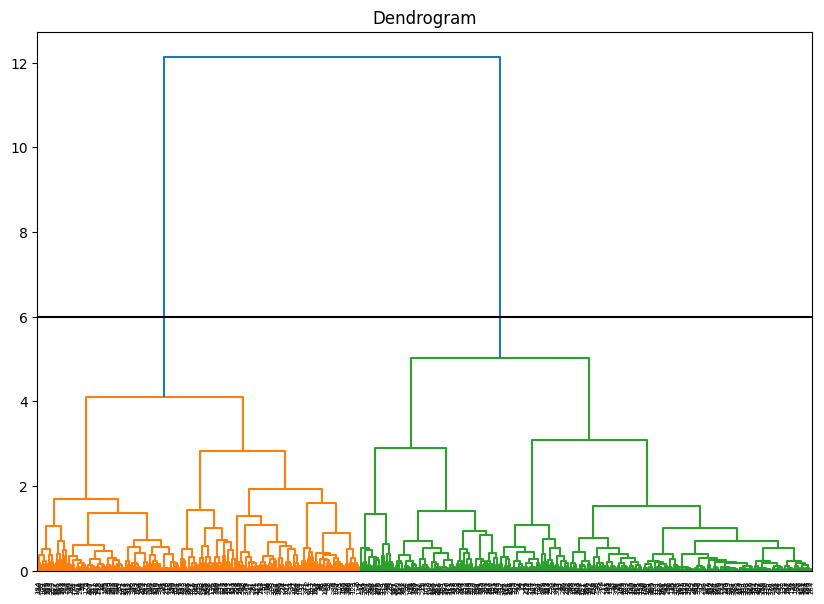
plt.figure(figsize=(10,7))

plt.title("Dendrogram")

Z=linkage(scaled,method='ward')

dendrograms=dendrogram(Z)

plt.axhline(y=6,color='black')



cluster = AgglomerativeClustering(n\_clusters=2, linkage='ward')

cluster.fit\_predict(scaled)

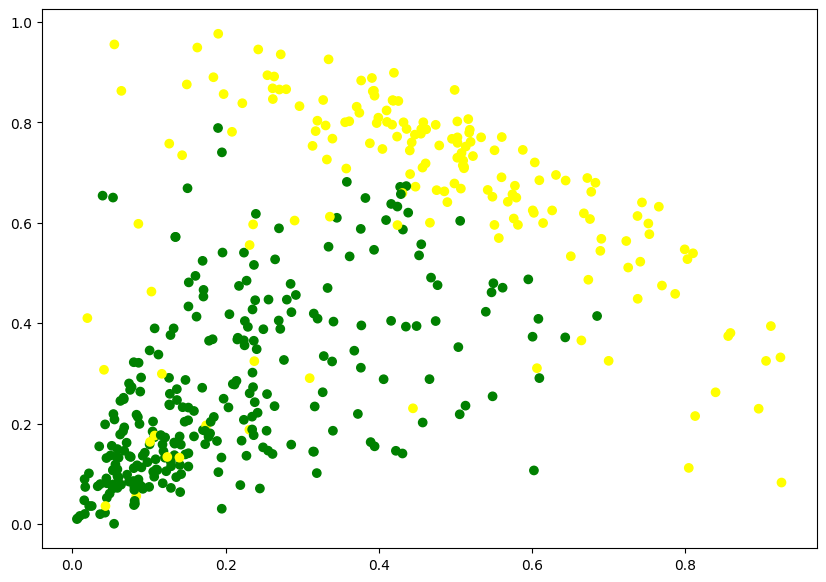
cluster = AgglomerativeClustering(n\_clusters=2, linkage='complete', metric='euclidean')

cluster.fit\_predict(scaled)

#visualization

plt.figure(figsize=(10,7))

plt.scatter(scaled['Milk'],scaled['Grocery'],c=cluster.labels\_,cmap = mcolors.ListedColormap(["yellow", "green"]))



import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

import scipy.cluster.hierarchy as sch

# Load the dataset

file\_path = './Wholesale customers data.csv'

wholesale\_data = pd.read\_csv(file\_path)

# Select the spending columns for clustering

spending\_data = wholesale\_data.iloc[:, 2:]

# Normalize the data

scaler = StandardScaler()

spending\_data\_normalized = scaler.fit\_transform(spending\_data)

# Plot the dendrogram to find the optimal number of clusters

plt.figure(figsize=(10, 7))

dendrogram = sch.dendrogram(sch.linkage(spending\_data\_normalized, method='ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean Distances')

plt.show()

agg\_clustering = AgglomerativeClustering(n\_clusters=4, linkage='ward')

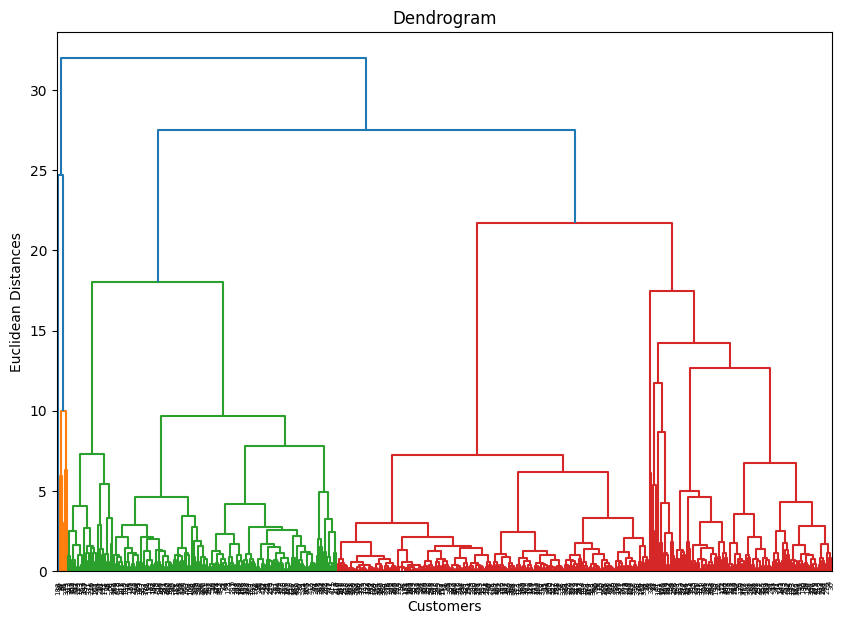
cluster\_labels = agg\_clustering.fit\_predict(spending\_data\_normalized)

# Calculate the silhouette score to evaluate clustering

silhouette\_avg = silhouette\_score(spending\_data\_normalized, cluster\_labels)

# Output the silhouette score

silhouette\_avg



**Conclusion:**

This experiment successfully applied dimensionality reduction techniques to the Adult Census Income dataset, enhancing model performance. Through normalization and clustering, we reduced noise and redundancy, resulting in improved accuracy, precision, recall, and F1 score. The silhouette score confirmed the effectiveness of our clustering approach.

Overall, this work highlighted the importance of dimensionality reduction in machine learning, demonstrating its potential to enhance model performance and interpretability. Future research could explore additional techniques like PCA or t-SNE for further improvements.