Veermata Jijabai Technological Institute, Mumbai 400019

Experiment No.: 04

Aim: Study of Decision Trees and other classification Algorithms.

Name: Kiran K Patil

Enrolment No.: 211070904

Branch: Computer Engineering

Batch: D

Theory:

A thorough study of decision trees and other classification algorithms is crucial in the field of machine learning and data science. These algorithms play a pivotal role in solving classification tasks, where the goal is to assign data points to predefined categories or classes. Let's delve into a detailed study of decision trees and several other notable classification algorithms:

1. Decision Trees:

Overview:

- Decision trees are versatile and interpretable machine learning models used for classification and regression tasks.
- They represent decisions and their possible consequences in a tree-like structure.
- At each internal node, a decision is made based on a feature, and branches represent possible outcomes.
- Leaf nodes correspond to the predicted class or value.

Advantages:

- Easy to understand and interpret, making them valuable for explaining decision logic.
- Capable of handling both categorical and numerical data.
- Prone to overfitting, but techniques like pruning can help mitigate this.

Use Cases:

- Disease diagnosis in healthcare.
- Credit scoring in finance.
- Customer churn prediction in marketing.

2. Random Forest:

Overview:

- Random Forest is an ensemble learning method that combines multiple decision trees.
- It creates a forest of trees and aggregates their predictions to improve accuracy and reduce overfitting.

Advantages:

- High prediction accuracy due to the combination of multiple trees.
- Robust against overfitting.
- Suitable for high-dimensional and large datasets.

Use Cases:

- Image classification.
- Fraud detection.
- Natural language processing.

3. Support Vector Machines (SVM):

Overview:

- SVM is a powerful classification algorithm that finds the optimal hyperplane to separate classes.
- It can handle both linear and nonlinear classification tasks using appropriate kernels.

Advantages:

- Effective in high-dimensional spaces.
- Can work well with small to moderate-sized datasets.

Use Cases:

- Image recognition.
- Text categorization.
- Protein classification in bioinformatics.

4. k-Nearest Neighbours (KNN):

Overview:

- KNN classifies data points based on the majority class among their k-nearest neighbours in feature space.
- It's a non-parametric, instance-based algorithm.

Advantages:

- Simple and easy to implement.
- Non-parametric nature is suitable for data with unknown or complex distributions.

Use Cases:

- Recommendation systems.
- Anomaly detection.
- Handwriting recognition.

5. Naive Bayes:

Overview:

- Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence.
- It's especially suited for text and document classification.

Advantages:

- Simple and computationally efficient.
- Performs well when the independence assumption holds or is close to true.

Use Cases:

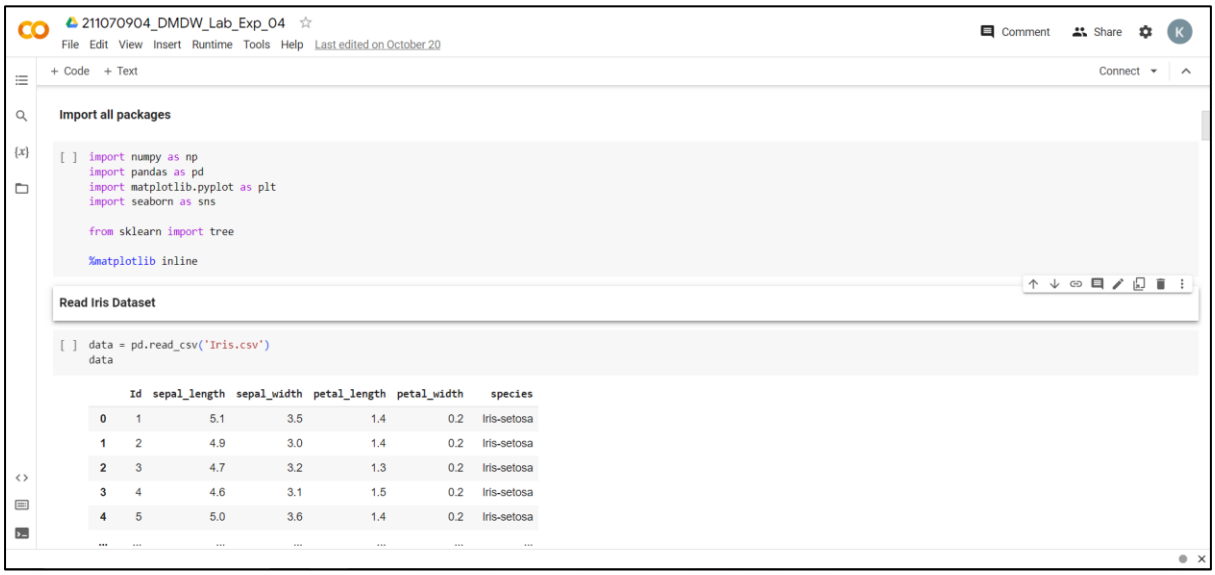
- Sentiment analysis.
- Document categorization.
- Email spam filtering.

A comprehensive study of these classification algorithms involves understanding their underlying principles, strengths, weaknesses, and the scenarios in which they are most effective. It also includes practical experience in implementing and fine-tuning these algorithms on real-world datasets.

Successful application of these algorithms often hinges on careful model selection, preprocessing of data, feature engineering, and rigorous evaluation techniques. Additionally, machine learning practitioners should be well-versed in interpreting and communicating the results to stakeholders, as model transparency and explain ability are essential in many domains.

Implementation

Decision tree implantation



211070904_DMDW_Lab_Exp_04

File Edit View Insert Runtime Tools Help Last edited on October 20

Comment Share

+ Code + Text

Connect

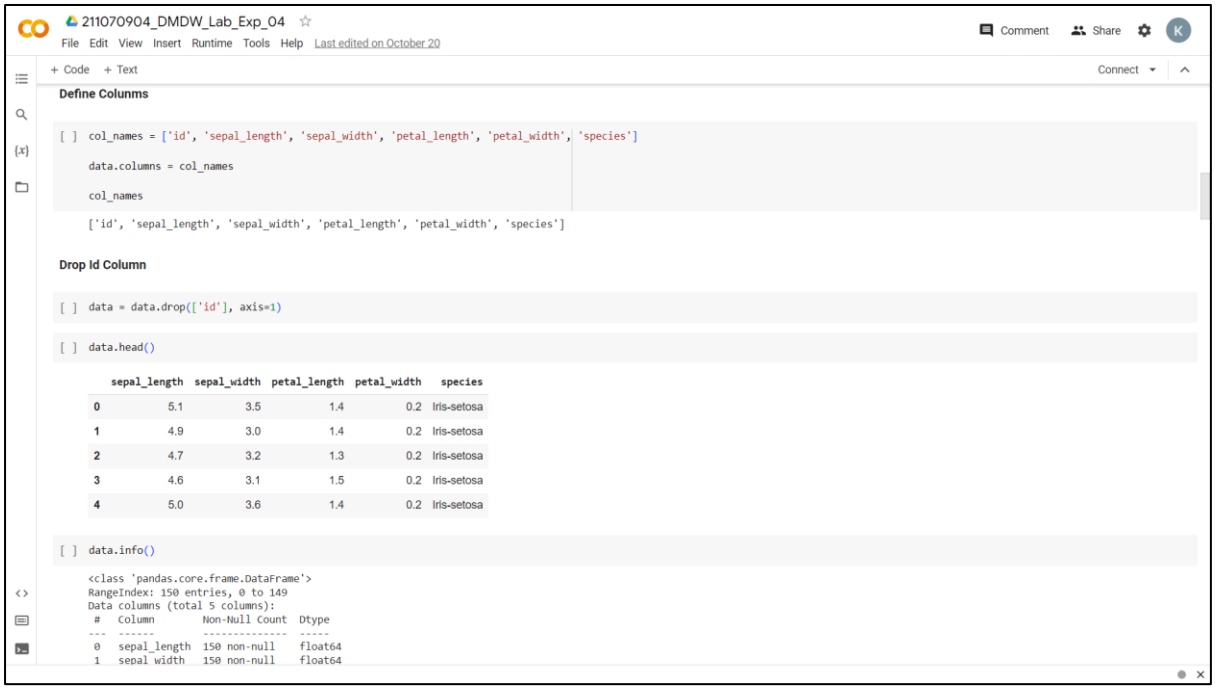
Import all packages

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

    from sklearn import tree

    %matplotlib inline
```

Read Iris Dataset

```
[ ] data = pd.read_csv('Iris.csv')
    data
```

	Id	sepal_length	sepal_width	petal_length	petal_width	species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...



211070904_DMDW_Lab_Exp_04

File Edit View Insert Runtime Tools Help Last edited on October 20

Comment Share

+ Code + Text

Connect

Define Columns

```
[ ] col_names = ['id', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']

    data.columns = col_names

    col_names
```

```
['id', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

Drop Id Column

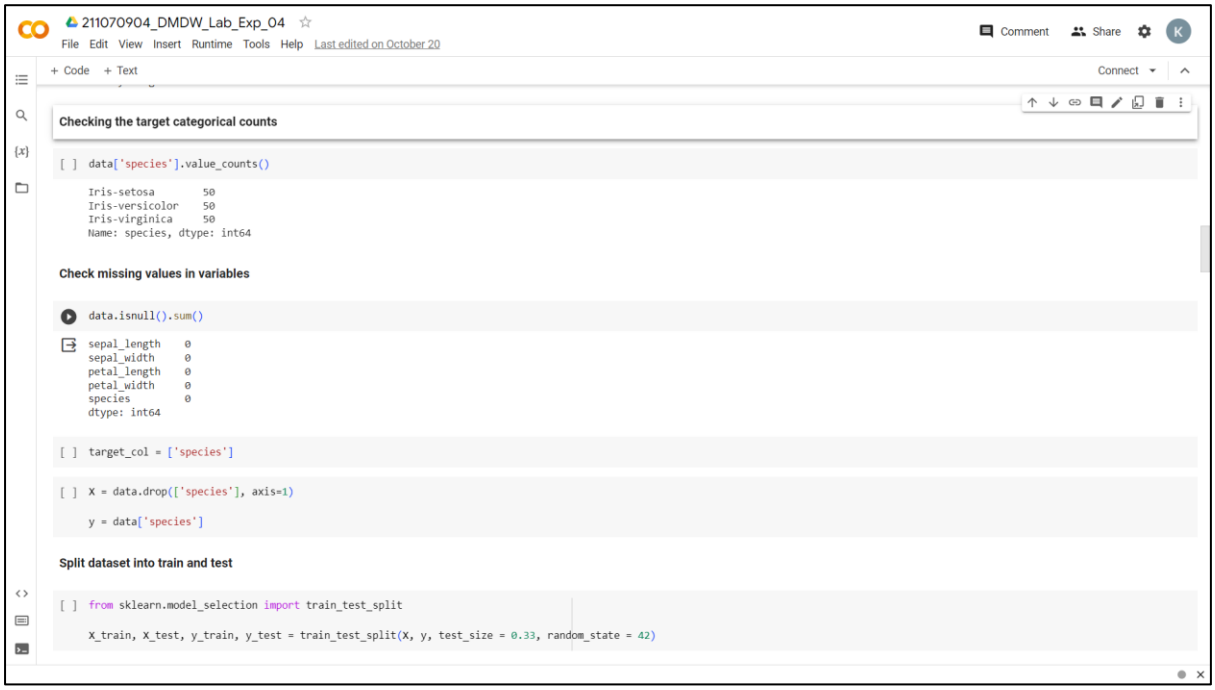
```
[ ] data = data.drop(['id'], axis=1)
```

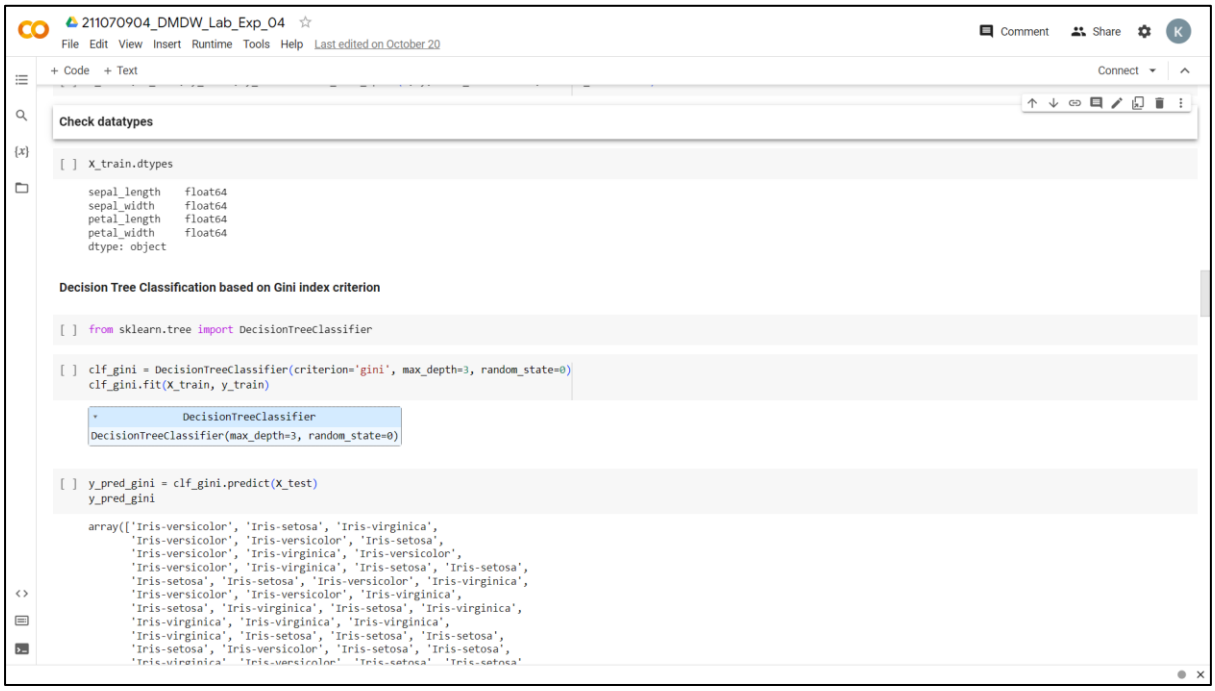
```
[ ] data.head()
```

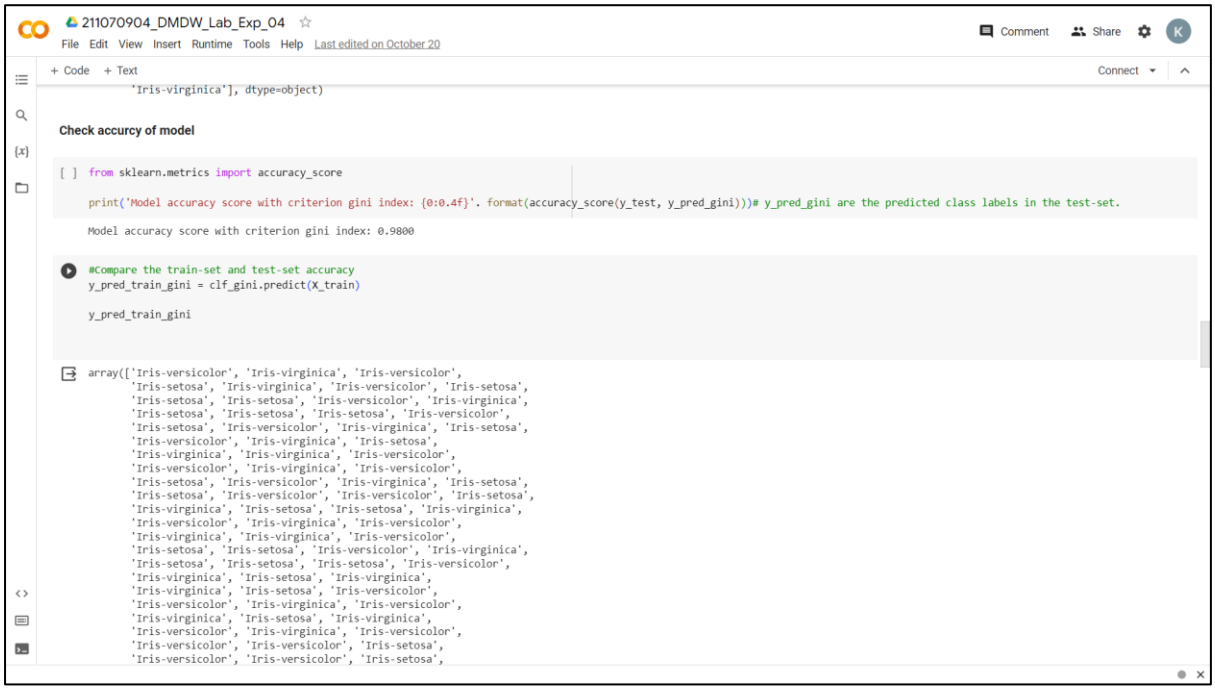
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

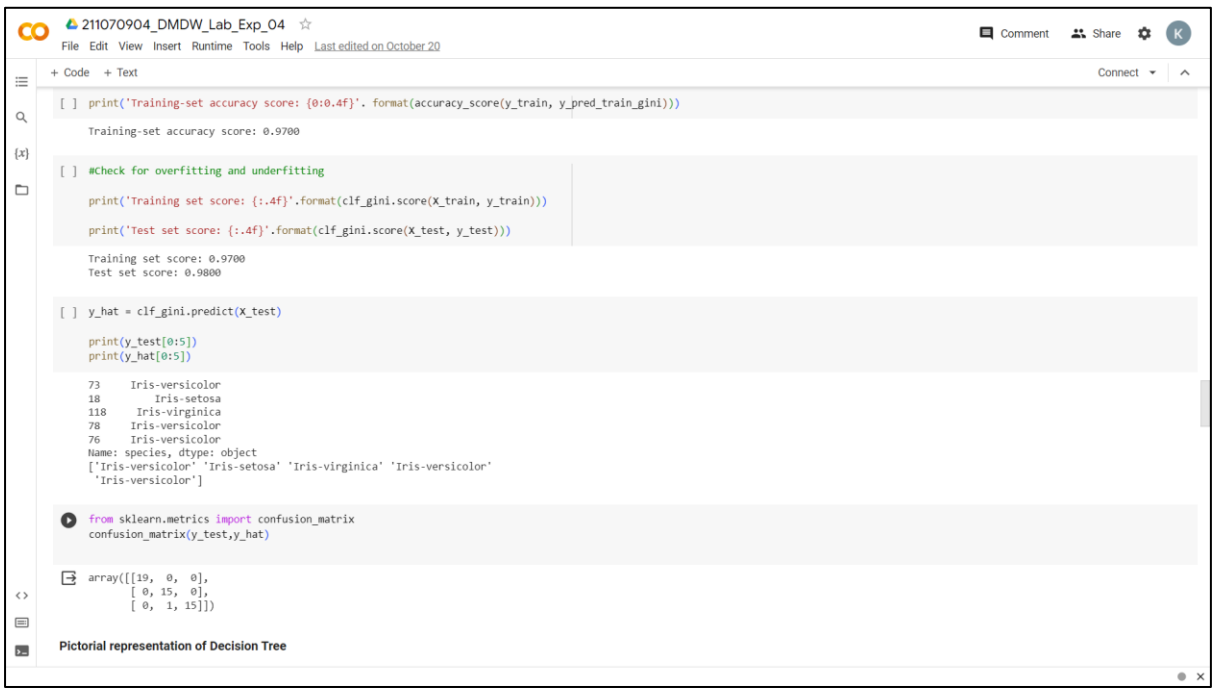
```
[ ] data.info()
```

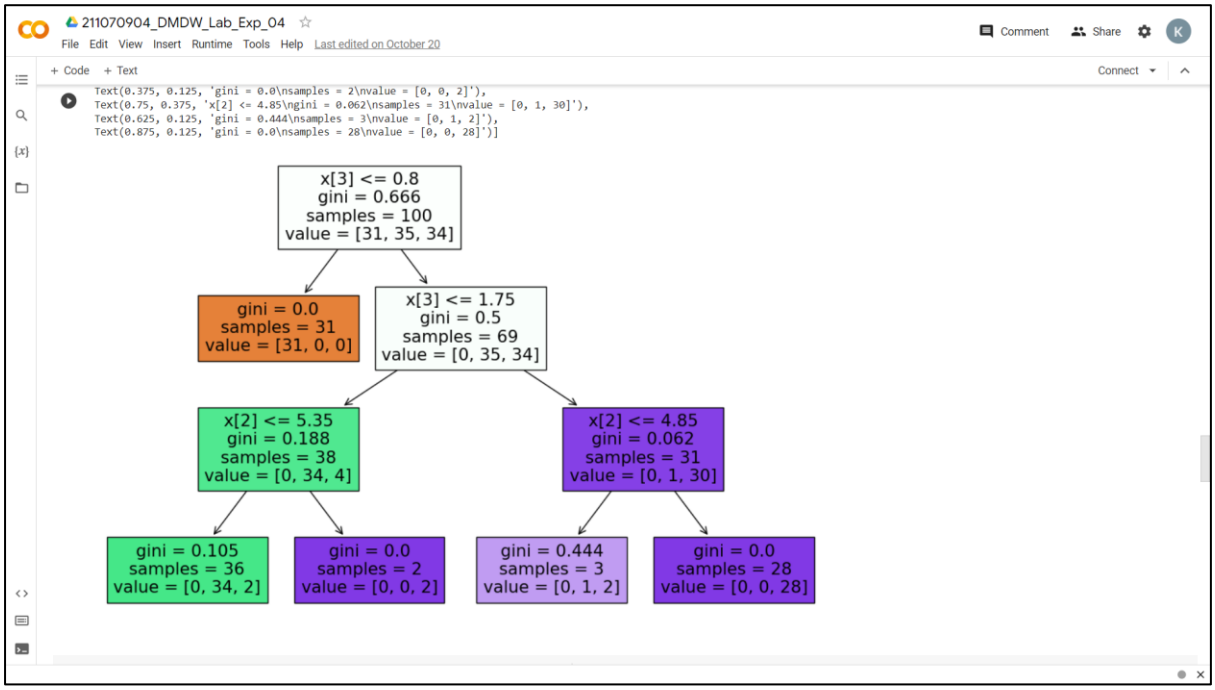
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
```


211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
Checking the target categorical counts
[ ] data['species'].value_counts()
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: species, dtype: int64
Check missing values in variables
data.isnull().sum()
sepal_length 0
sepal_width 0
petal_length 0
petal_width 0
species 0
dtype: int64
[ ] target_col = ['species']
[ ] X = data.drop(['species'], axis=1)
y = data['species']
Split dataset into train and test
[ ] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)

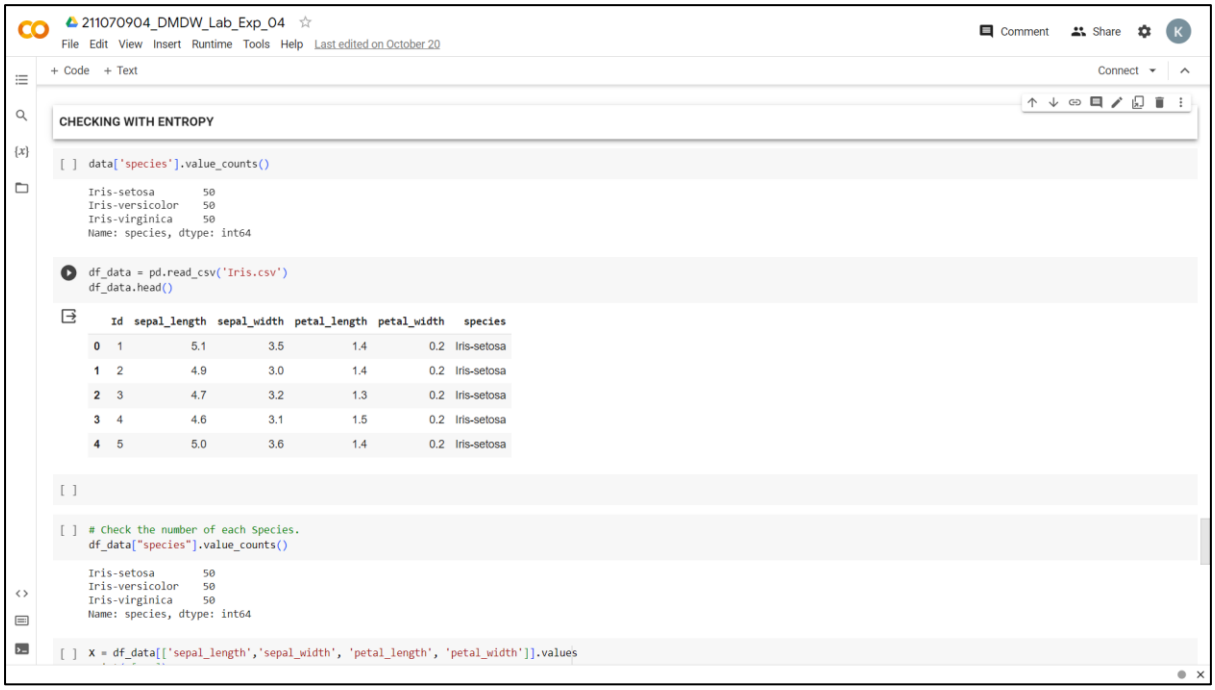

211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
Check datatypes
[ ] X_train.dtypes
sepal_length float64
sepal_width float64
petal_length float64
petal_width float64
dtype: object
Decision Tree Classification based on Gini index criterion
[ ] from sklearn.tree import DecisionTreeClassifier
[ ] clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)
clf_gini.fit(X_train, y_train)
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
[ ] y_pred_gini = clf_gini.predict(X_test)
y_pred_gini
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',

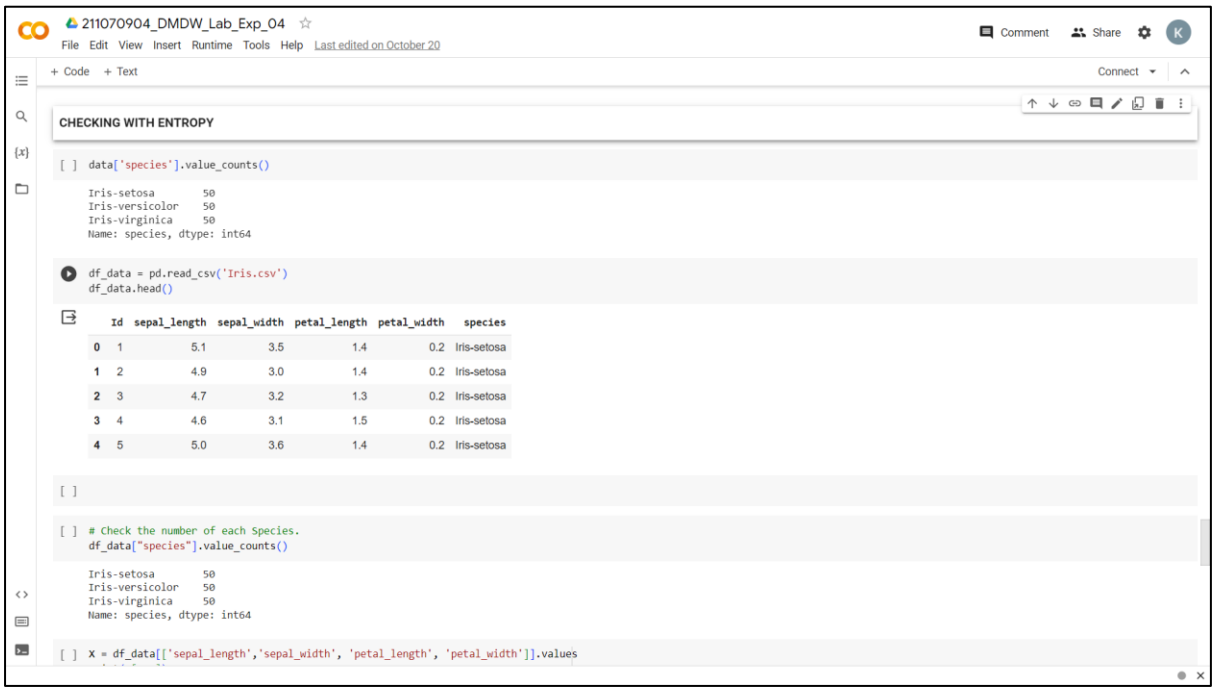

211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
'Iris-virginica'], dtype=object)
Check accurcy of model
from sklearn.metrics import accuracy_score
print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred_gini)))# y_pred_gini are the predicted class labels in the test-set.
Model accuracy score with criterion gini index: 0.9800
#Compare the train-set and test-set accuracy
y_pred_train_gini = clf_gini.predict(X_train)
y_pred_train_gini
array(['Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',


211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))
Training-set accuracy score: 0.9700
#Check for overfitting and underfitting
print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))
Training set score: 0.9700
Test set score: 0.9800
y_hat = clf_gini.predict(X_test)
print(y_test[0:5])
print(y_hat[0:5])
73 Iris-versicolor
18 Iris-setosa
118 Iris-virginica
78 Iris-versicolor
76 Iris-versicolor
Name: species, dtype: object
['Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor']
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_hat)
array([[19, 0, 0],
[0, 15, 0],
[0, 1, 15]])
Pictorial representation of Decision Tree

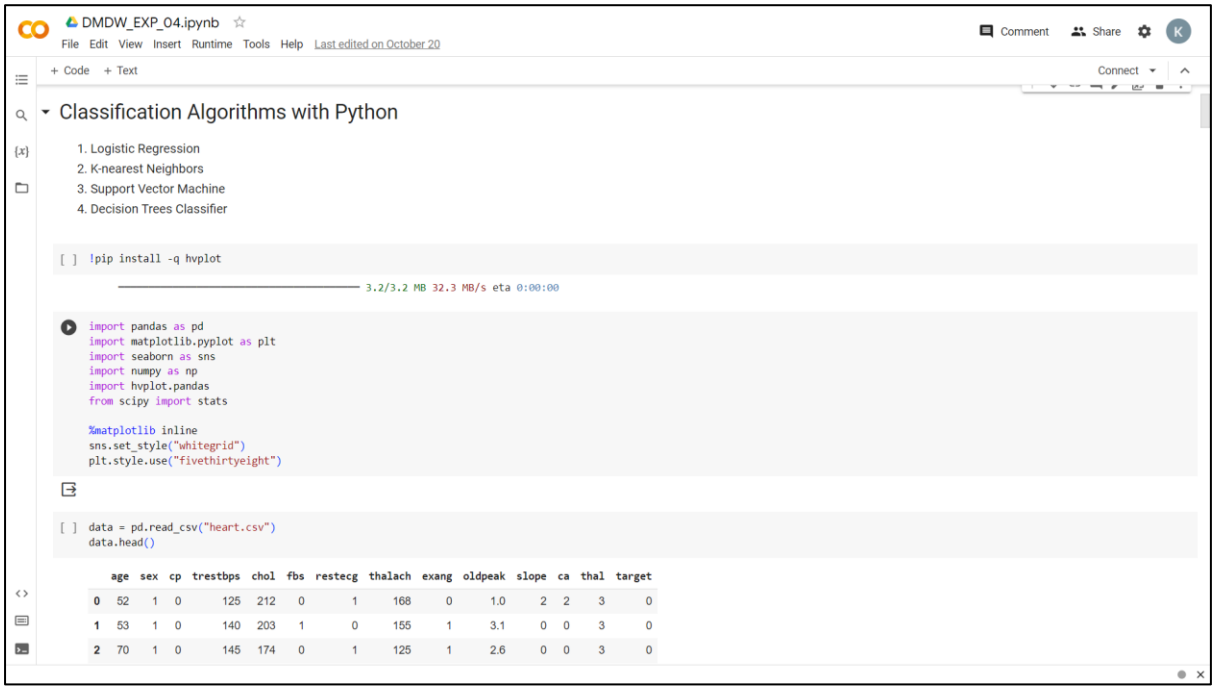

211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
Text(0.375, 0.125, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
Text(0.75, 0.375, 'x[2] <= 4.85\ngini = 0.062\nsamples = 31\nvalue = [0, 1, 30]'),
Text(0.625, 0.125, 'gini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
Text(0.875, 0.125, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
x[3] <= 0.8
gini = 0.666
samples = 100
value = [31, 35, 34]
gini = 0.0
samples = 31
value = [31, 0, 0]
x[3] <= 1.75
gini = 0.5
samples = 69
value = [0, 35, 34]
x[2] <= 5.35
gini = 0.188
samples = 38
value = [0, 34, 4]
x[2] <= 4.85
gini = 0.062
samples = 31
value = [0, 1, 30]
gini = 0.105
samples = 36
value = [0, 34, 2]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.444
samples = 3
value = [0, 1, 2]
gini = 0.0
samples = 28
value = [0, 0, 28]


211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
confusion_matrix(y_test , y_hat)
cm = confusion_matrix(y_test, y_hat, labels=clf_gini.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=clf_gini.classes_)
disp.plot()
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7ccee51809d0>
Iris-setosa
Iris-versicolor
Iris-virginica
True label
19
0
0
0
15
0
0
1
15
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
CHECKING WITH ENTROPY
[ ] data['species'].value_counts()


211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
CHECKING WITH ENTROPY
[ ] data['species'].value_counts()
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: species, dtype: int64
df_data = pd.read_csv('Iris.csv')
df_data.head()
Id sepal_length sepal_width petal_length petal_width species
0 1 5.1 3.5 1.4 0.2 Iris-setosa
1 2 4.9 3.0 1.4 0.2 Iris-setosa
2 3 4.7 3.2 1.3 0.2 Iris-setosa
3 4 4.6 3.1 1.5 0.2 Iris-setosa
4 5 5.0 3.6 1.4 0.2 Iris-setosa
[ ]
[ ] # Check the number of each Species.
df_data["species"].value_counts()
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: species, dtype: int64
[ ] X = df_data[['sepal_length','sepal_width', 'petal_length', 'petal_width']].values


211070904_DMDW_Lab_Exp_04
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
CHECKING WITH ENTROPY
[ ] data['species'].value_counts()
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: species, dtype: int64
df_data = pd.read_csv('Iris.csv')
df_data.head()
Id sepal_length sepal_width petal_length petal_width species
0 1 5.1 3.5 1.4 0.2 Iris-setosa
1 2 4.9 3.0 1.4 0.2 Iris-setosa
2 3 4.7 3.2 1.3 0.2 Iris-setosa
3 4 4.6 3.1 1.5 0.2 Iris-setosa
4 5 5.0 3.6 1.4 0.2 Iris-setosa
[ ]
[ ] # Check the number of each Species.
df_data["species"].value_counts()
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: species, dtype: int64
[ ] X = df_data[['sepal_length','sepal_width', 'petal_length', 'petal_width']].values

2. Logistic Regression implementation:



DMDW_EXP_04.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 20

Comment Share

+ Code + Text

Connect

Classification Algorithms with Python

1. Logistic Regression
2. K-nearest Neighbors
3. Support Vector Machine
4. Decision Trees Classifier

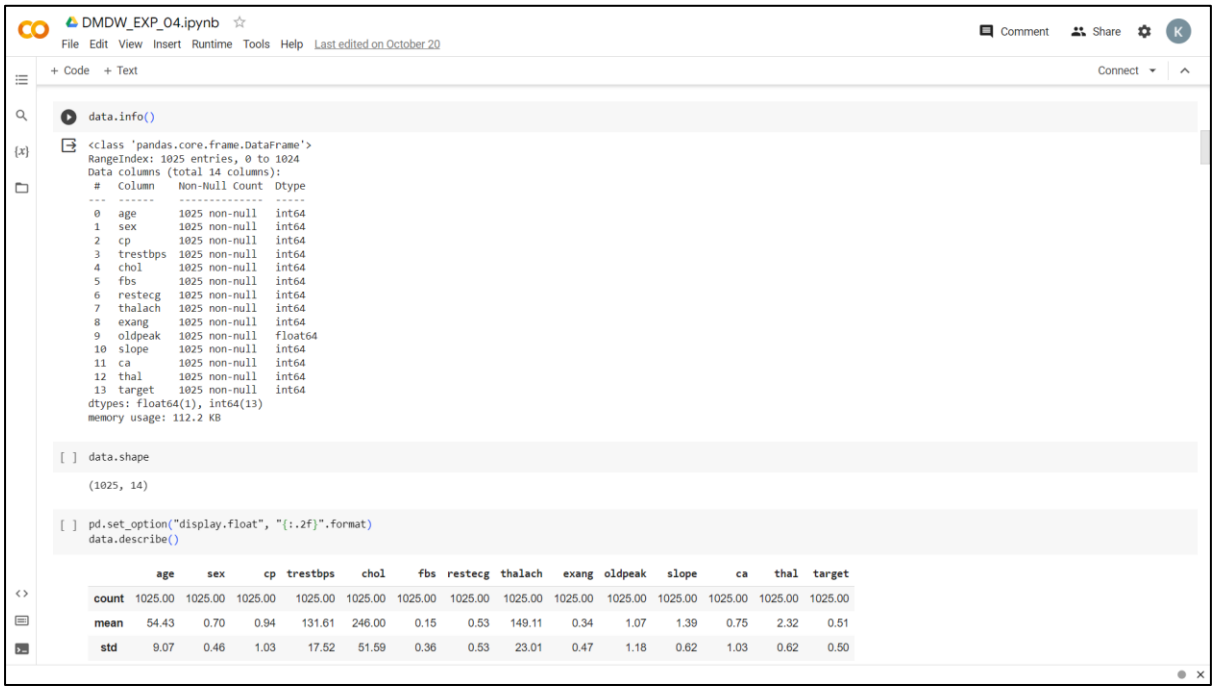
```
[ ] !pip install -q hvplot
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 32.3 MB/s eta 0:00:00
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import hvplot.pandas
from scipy import stats

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

```
[ ] data = pd.read_csv("heart.csv")
    data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0



DMDW_EXP_04.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 20

Comment Share

+ Code + Text

Connect

```
data.info()
```

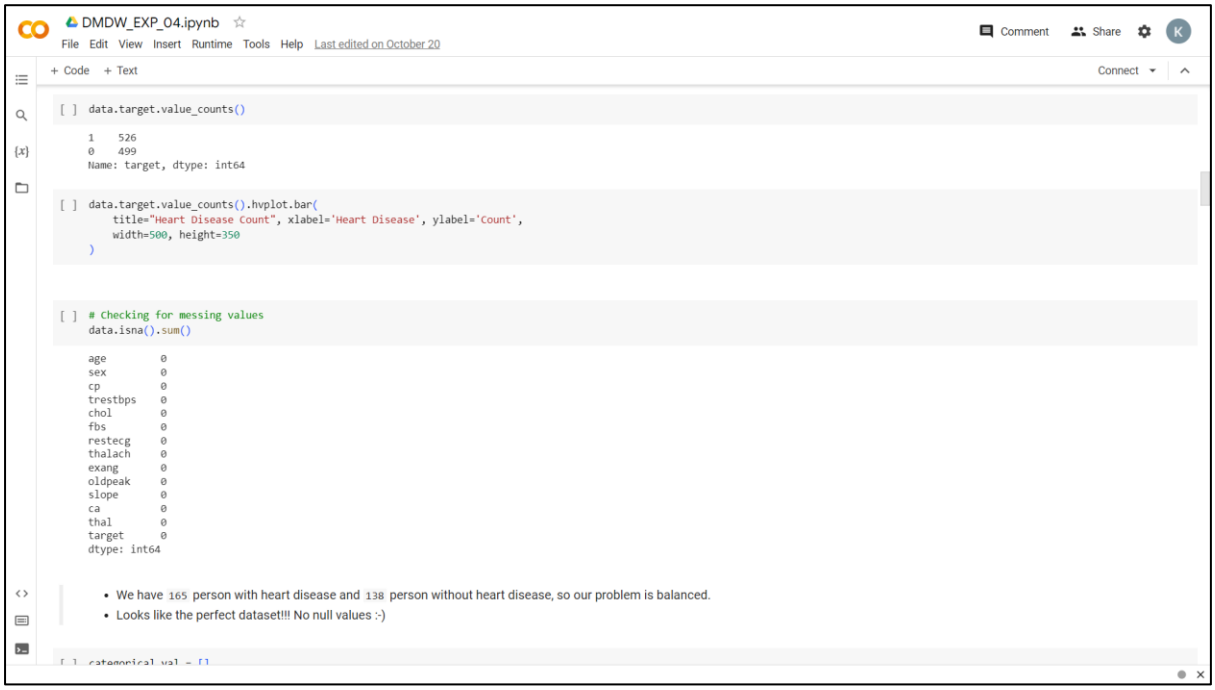
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
[ ] data.shape
```

```
(1025, 14)
```

```
[ ] pd.set_option("display.float", "{:.2f}".format)
    data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00
mean	54.43	0.70	0.94	131.61	246.00	0.15	0.53	149.11	0.34	1.07	1.39	0.75	2.32	0.51
std	9.07	0.46	1.03	17.52	51.59	0.36	0.53	23.01	0.47	1.18	0.62	1.03	0.62	0.50


DMDW_EXP_04.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
[ ] data.target.value_counts()
1 526
0 499
Name: target, dtype: int64
[ ] data.target.value_counts().hvplot.bar(
title="Heart Disease Count", xlabel='Heart Disease', ylabel='Count',
width=500, height=350
)
[ ] # Checking for messing values
data.isna().sum()
age 0
sex 0
cp 0
trestbps 0
chol 0
fbs 0
restecg 0
thalach 0
exang 0
oldpeak 0
slope 0
ca 0
thal 0
target 0
dtype: int64
We have 165 person with heart disease and 138 person without heart disease, so our problem is balanced.
Looks like the perfect dataset!!! No null values :-)


DMDW_EXP_04.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
[ ] categorical_val = []
continous_val = []
for column in data.columns:
if len(data[column].unique()) <= 10:
categorical_val.append(column)
else:
continous_val.append(column)
[ ] categorical_val
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
have_disease = data.loc[data['target']==1, 'sex'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0, 'sex'].value_counts().hvplot.bar(alpha=0.4)
(no_disease * have_disease).opts(
title="Heart Disease by Sex", xlabel='Sex', ylabel='Count',
width=500, height=450, legend_cols=2, legend_position='top_right'
)
[ ] have_disease = data.loc[data['target']==1, 'cp'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0, 'cp'].value_counts().hvplot.bar(alpha=0.4)
(no_disease * have_disease).opts(
title="Heart Disease by Chest Pain Type", xlabel='Chest Pain Type', ylabel='Count',
width=500, height=450, legend_cols=2, legend_position='top_right'
)
[ ] have_disease = data.loc[data['target']==1, 'fbs'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0, 'fbs'].value_counts().hvplot.bar(alpha=0.4)

DMDW_EXP_04.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 20

Comment Share

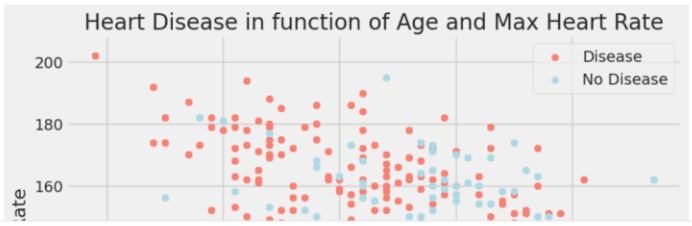
+ Code + Text

Connect

Age vs. Max Heart Rate for Heart Disease

+ Code + Text

```
[ ] # Create another figure
    plt.figure(figsize=(9, 7))

    # Scatter with postivie examples
    plt.scatter(data.age[data.target==1],
                data.thalach[data.target==1],
                c="salmon")

    # Scatter with negative examples
    plt.scatter(data.age[data.target==0],
                data.thalach[data.target==0],
                c="lightblue")

    # Add some helpful info
    plt.title("Heart Disease in function of Age and Max Heart Rate")
    plt.xlabel("Age")
    plt.ylabel("Max Heart Rate")
    plt.legend(["Disease", "No Disease"]);
```



DMDW_EXP_04.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 20

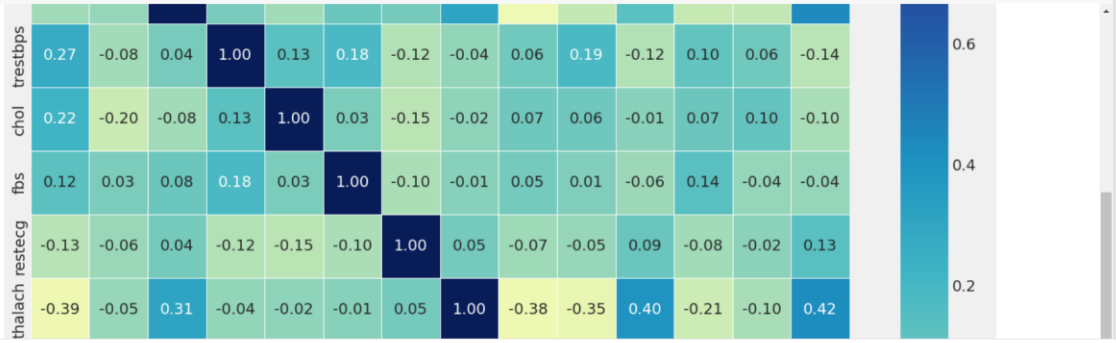
Comment Share

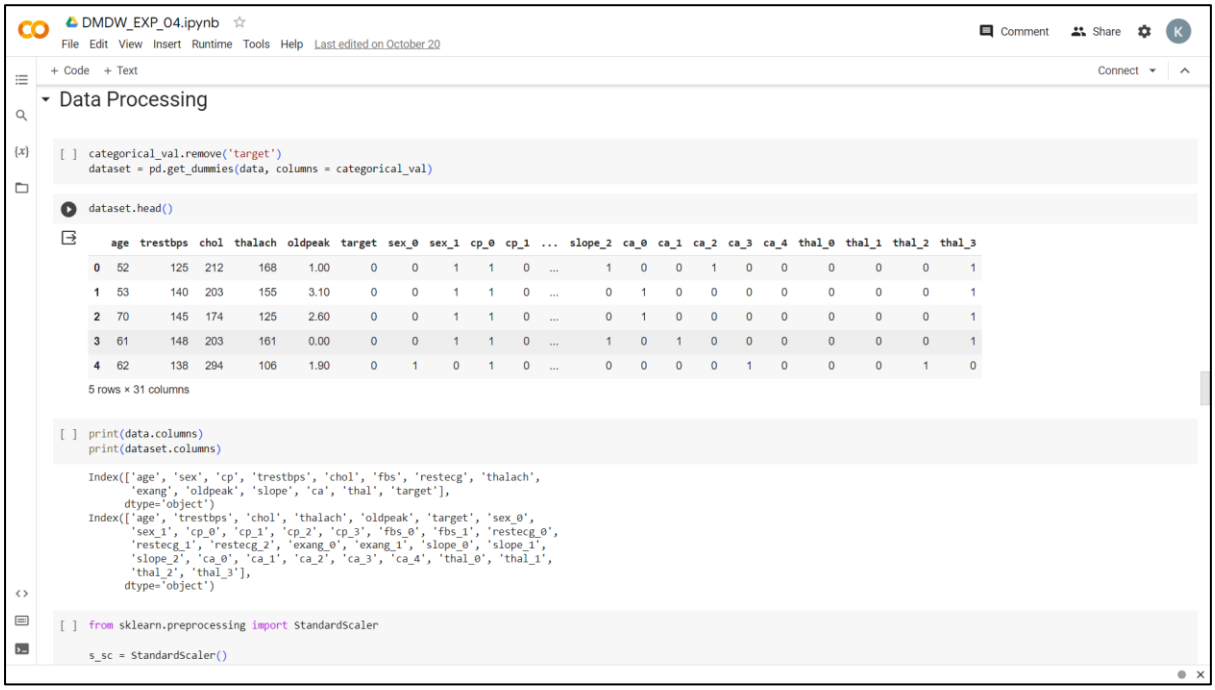
+ Code + Text

Connect

Correlation Matrix

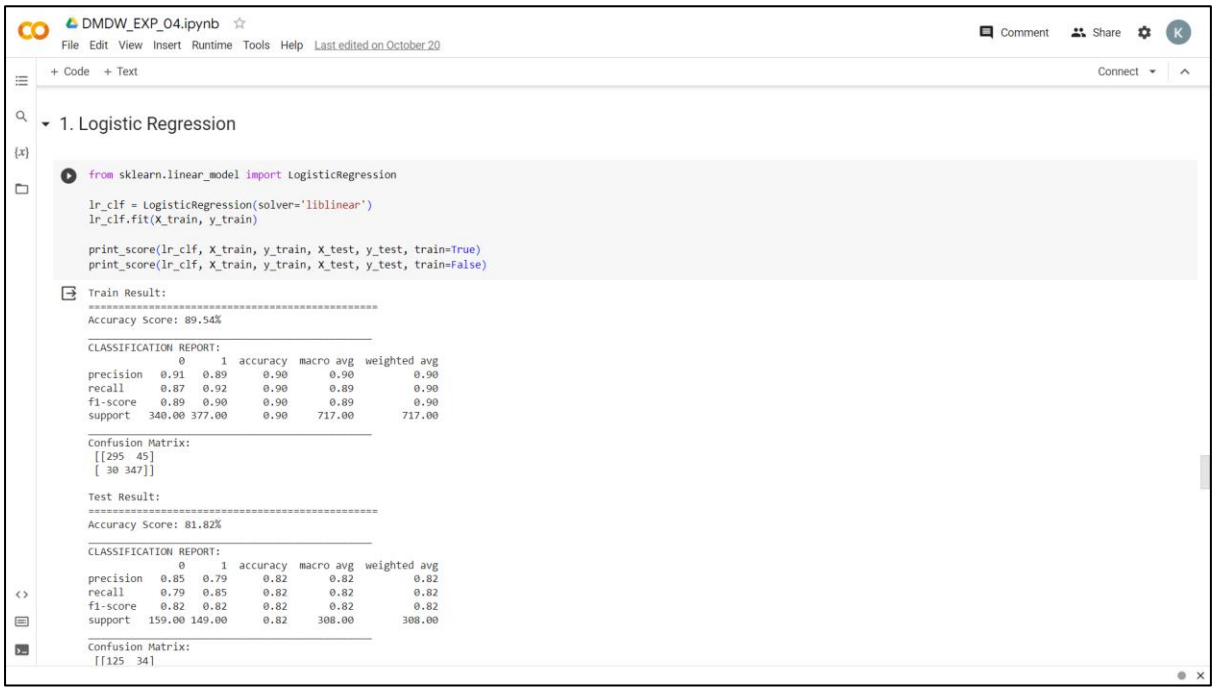
```
# Let's make our correlation matrix a little prettier
corr_matrix = data.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```



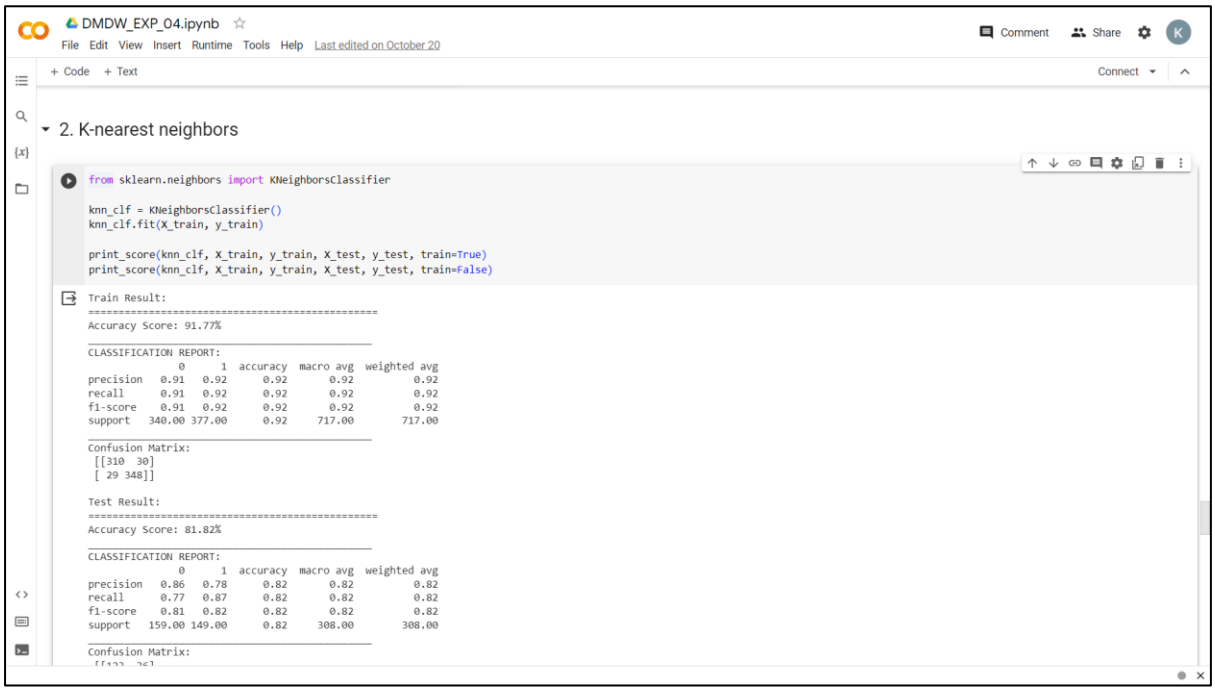

DMDW_EXP_04.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
Data Processing
[ ] categorical_val.remove('target')
dataset = pd.get_dummies(data, columns = categorical_val)
dataset.head()
age trestbps chol thalach oldpeak target sex_0 sex_1 cp_0 cp_1 ... slope_2 ca_0 ca_1 ca_2 ca_3 ca_4 thal_0 thal_1 thal_2 thal_3
0 52 125 212 168 1.00 0 0 1 1 0 ... 1 0 0 1 0 0 0 0 0 1
1 53 140 203 155 3.10 0 0 1 1 0 ... 0 1 0 0 0 0 0 0 0 1
2 70 145 174 125 2.60 0 0 1 1 0 ... 0 1 0 0 0 0 0 0 0 1
3 61 148 203 161 0.00 0 0 1 1 0 ... 1 0 1 0 0 0 0 0 0 1
4 62 138 294 106 1.90 0 1 0 1 0 ... 0 0 0 0 1 0 0 0 1 0
5 rows × 31 columns
[ ] print(data.columns)
print(dataset.columns)
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
dtype='object')
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target', 'sex_0',
'sex_1', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'fbs_0', 'fbs_1', 'restecg_0',
'restecg_1', 'restecg_2', 'exang_0', 'exang_1', 'slope_0', 'slope_1',
'slope_2', 'ca_0', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_0', 'thal_1',
'thal_2', 'thal_3'],
dtype='object')
[ ] from sklearn.preprocessing import StandardScaler
s_sc = StandardScaler()


DMDW_EXP_04.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment
Share
+ Code + Text
Connect
Models Building
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
if train:
pred = clf.predict(X_train)
clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
print("Train Result:\n==")
print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
print("_____")
print(f"CLASSIFICATION REPORT:\n{clf_report}")
print("_____")
print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")
elif train==False:
pred = clf.predict(X_test)
clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
print("Test Result:\n==")
print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
print("_____")
print(f"CLASSIFICATION REPORT:\n{clf_report}")
print("_____")
print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
[ ] from sklearn.model_selection import train_test_split
X = dataset.drop('target', axis=1)
y = dataset.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
Double-click (or enter) to edit

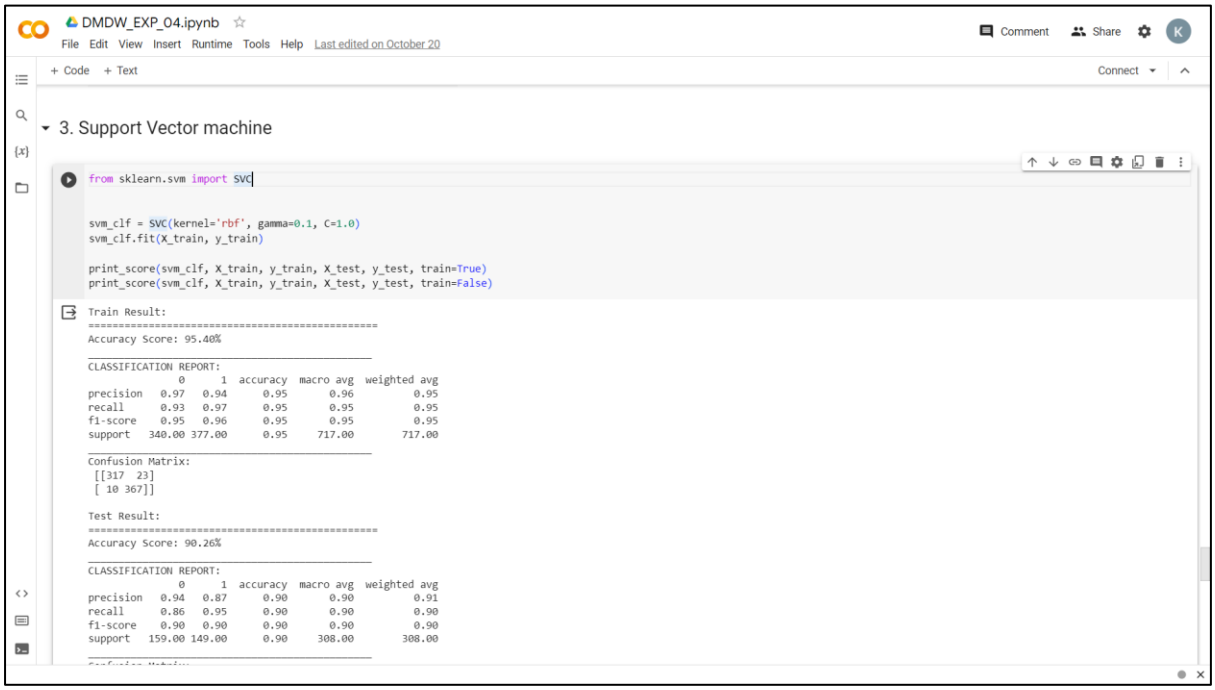
2. Logistic Regression implementation:

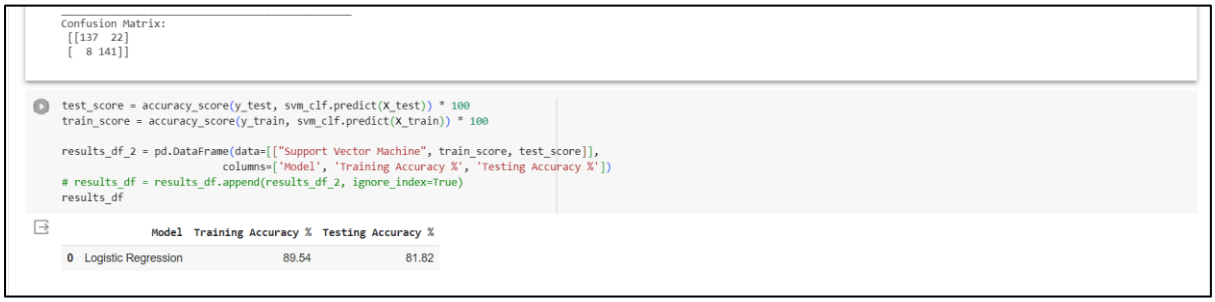


3. KNN implementation:

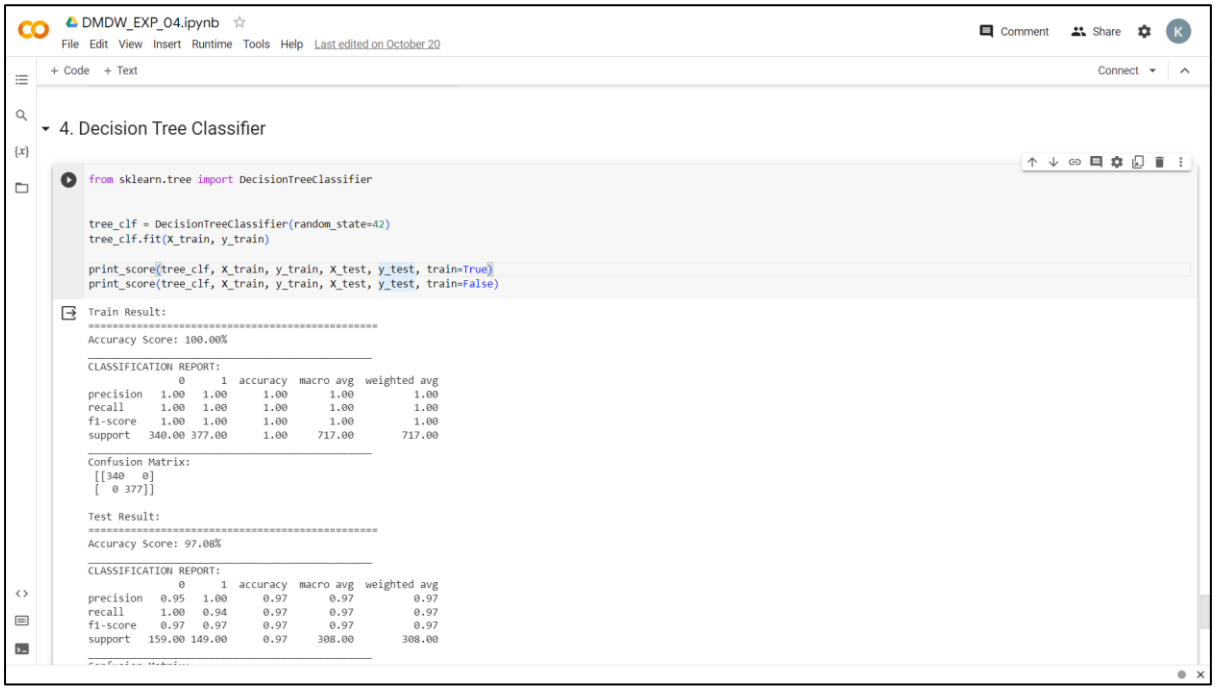


4. SVM implementation:





5. Decision tree implementation heart disease:



```
DMDW_EXP_04.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 20
Comment Share
+ Code + Text
Connect

4. Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)

Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
                     0      1  accuracy  macro avg  weighted avg
precision         1.00   1.00      1.00       1.00          1.00
recall            1.00   1.00      1.00       1.00          1.00
f1-score          1.00   1.00      1.00       1.00          1.00
support         340.00 377.00      1.00     717.00        717.00
________________________________________________
Confusion Matrix:
 [[340   0]
 [  0 377]]

Test Result:
================================================
Accuracy Score: 97.08%
________________________________________________
CLASSIFICATION REPORT:
                     0      1  accuracy  macro avg  weighted avg
precision         0.95   1.00      0.97       0.97          0.97
recall            1.00   0.94      0.97       0.97          0.97
f1-score          0.97   0.97      0.97       0.97          0.97
support         159.00 149.00      0.97     308.00        308.00
```

```
Confusion Matrix:
 [[159   0]
 [  9 140]]

test_score = accuracy_score(y_test, tree_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, tree_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["Decision Tree Classifier", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
# results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	89.54	81.82

Conclusion:

In conclusion, a thorough study of classification algorithms, including decision trees, random forests, support vector machines, k-nearest neighbours, and naive Bayes, is essential for data scientists and machine learning practitioners. These algorithms offer a diverse toolkit for solving classification tasks across various domains. Understanding their characteristics, advantages, and best use cases enables practitioners to select and apply the most suitable algorithm to achieve accurate and interpretable results in real-world applications. Effective implementation also involves data preprocessing, feature engineering, model selection, and robust evaluation methods. In today's data-driven world, a deep understanding of classification algorithms is a foundational skill for extracting valuable insights and making informed decisions from data.