



Name - ROSTHNI. R Std _____ Sec _____

Roll No. _____ Subject _____ School/College _____

School/College Tel. No. _____ Parents Tel. No. _____

Write a program to import and export data using pandas library function.

```
import pandas as pd
airbnb_data = pd.read_csv("data/listings.csv")
airbnb_data.head()
```

Output

id	name	host_id	host_name	neighbourhood_group	latitude
2245	Zen-east	2466	Paddy	Null	78702 97.812
5245	Austin	2466	Paddy	Null	78729 -98.98

1.0	2.0	3.0	4.0	5.0	6.0
5.0	8.0	0.0	1.0	2.0	3.0
1.0	2.0	1.0	8.0	1.0	2.0

Step 2: In sub assignment part

(Five students have to work) `listings.csv`

Structure of the file

listing_id, name, neighbourhood, neighbourhood_group

1.0	2.0	3.0	4.0	5.0
5.0	8.0	0.0	1.0	2.0
1.0	2.0	1.0	8.0	1.0

1.0	2.0	3.0	4.0	5.0
5.0	8.0	0.0	1.0	2.0
1.0	2.0	1.0	8.0	1.0

1.0	2.0	3.0	4.0	5.0
5.0	8.0	0.0	1.0	2.0
1.0	2.0	1.0	8.0	1.0

Reading data from URL

```
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"  
iris_data = pd.read_csv(url, names=col_names)
```

```
iris_data = pd.read_csv(url, names=col_names)
```

```
iris_data.head(2)
```

Output: None

sepal-length-in-cm sepal-width-in-cm petal-length-in-cm petal-width-in-cm

5.1	3.5	1.4	0.2
5.9	4.5	1.5	0.2
7.0	4.9	1.6	0.1

Exporting dataframe to a csv file

```
iris_data.to_csv("cleaned_iris_data.csv")
```

Output:

sepal-length-in-cm sepal-width-in-cm petal-length-in-cm petal-width-in-cm

0	5.1	3.5	1.4	0.2
1	5.9	4.5	1.5	0.2
2	7.0	4.9	1.6	0.1

get the data from which will import

import os

import tarfile

import urllib.request, tarfile, forward

↳ use 'tarfile'

Download ROOT: "https://raw.githubusercontent.com/
mlpacker/handson-on-machine-learning/hands-on/"

HOUSING_PATH = os.path.join("data", "o1")

HOUSING_URL = DOWNLOAD_ROOT + "datasets/
housing/housing.tgz";

fetch_housing_data()

def load_housing_data(housing_path=
HOUSING_PATH):

data_path = os.path.join(housing_path,

return pd.read_csv(data_path)

housing.head() → 13 columns, one id

↳ missing values

Discover and visualize the data as

strat_train_set.shape, strat_test_set.shape

strat_test_set.reset_index().to_feather

(fname="data/o1/strat_test_set.f")

housing = strat_train_set.copy(); housing.

↳ drop id, shape

housing.plot(kind='scatter', x='longitude',

y='latitude')

plt.show() → time mark with green dots

↳ missing values and I need to drop

(id, longitude, y)

→ housing.drop(['id', 'longitude', 'y'])

Prepare the data for machine learning algorithm

housing = strat_train_set.drop(["median_house_value"], axis=1)

housing_labels = strat_train_set[["median_house_value"]].copy()

housing.shape, housing_labels.shape

imputer = SimpleImputer(strategy="median")
imputer.fit(housing_num)
imputer.statistics_

Select and train a model and test it

linreg = LinearRegression()
linreg.fit(housing_prepared, y=housing_labels)

some_data = housing.iloc[:5]

housing_predictions = linreg.predict(housing_prepared)

lin_rmse = np.sqrt(linreg.mse)

lin_rmse

some_data['label'] = target[some_data['label']]

some_data['pred'] = target[linreg.predict(housing_predictions)]

some_data['diff'] = target['label'] - target['pred']

some_data['abs_diff'] = abs(some_data['diff'])

some_data['abs_sq'] = abs(some_data['diff'])**2

some_data['sqrt_sq'] = np.sqrt(some_data['abs_sq'])

some_data['rmse'] = some_data['sqrt_sq'].mean()

some_data['rmse'] = np.sqrt(some_data['rmse'])

x-test = strat-test.set.drop(labels =
"median-house value")

y-test = strat-test.set['median-house-
value'].copy()

~~for~~ ~~dataframe~~ ~~is~~ ~~empty~~ ~~so~~ ~~train~~

~~Output~~: ~~median-ho~~ ~~use~~ ~~for~~ ~~train~~

The model is trained using the
dataset.

which includes 1400 training data points

(1400 rows) and 32 test data points

(32 rows) which are used to evaluate the

model's performance.

Model

(Random Forest)

accuracy

(Random Forest accuracy)

accuracy has grade associated with each data

Training accuracy has 26, others all 1

(Model has 26)

Accuracy is model's ability

(Model 100%)

Model file

random_forest\models\train\rf.pkl

File: [sat, Ias, 26, 8]

File: [sat, Ias, 26, 8]

Handling simple Linear Regression

(Multiple variable and more)

Import libraries:

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.model_selection
```

Import Data

```
df_sal = pd.read_csv('content/Salary.dat  
(csv)')
```

Analyze data

Describe

```
df_sal.describe()
```

Distribution

```
sns.distplot(df_sal['salary'])
```

Relationship between salary and experience

```
plt.scatter(df_sal['yearsofexperience'],  
           df_sal['salary'])
```

plt.title('salary vs experience')

plt.show()

Split data

split into independent/dependent variables

```
X = df_sal.iloc[:, :-1]
```

```
y = df_sal.iloc[:, -1]
```

Visualize predictions

Prediction on training sets and targets

```
plt.scatter(x_train, y_train, color='teal')  
plt.plot('x_train, y-pred_train')  
plt.xlabel('years of experience')  
plt.ylabel('salary')
```

Prediction on test set

```
plt.scatter(x_test, y-test)
```

Coefficient and Intercept

```
print(f'Coefficient = {regressor.coef_}')
```

O/P
11.573

Ridge
23.512

Output: 11.573 & 23.512

Coefficient of regression = 8 (Lasso)

Decision boundary chart

Decision boundary chart

Decision boundary chart

Decision boundary chart

(Decision boundary chart) according to Lasso

Decision boundary chart according to Ridge

Decision boundary chart

(Decision boundary chart) according to Lasso

Decision boundary chart

(Decision boundary chart) according to Ridge

Multiple Linear Regression

Import libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Import data

```
df_start = pd.read_csv('content/...  
(test + substrat up).csv')
```

Analyze data

```
{if Ann 2020 > 0.0? 1 : 0} for fitting
```

Describe

```
df.describe()
```

Distribution

```
plt.title('Profit Distribution plot')
plt.show()
```

Split into independent/dependent

```
X = df_start.iloc[:, :-1]
```

```
y = df_start.iloc[:, -1]
```

Train model

```
regressor = linearRegression()
regressor.fit(X_train, y_train)
```

Predict results

```
y_pred = regressor.predict(X_test)
```

Compare prediction

```
np.set_printoptions(precision=2)
```

Weekly Assignment 1
 Use an appropriate dataset for building the decision tree and apply this learning knowledge to classify

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
iris = load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
y_train = y_test = train_test_split(y, test_size=0.43)
```

tree_model = DecisionTreeClassifier

```
tree_model = tree_model.fit(X_train, y_train)
plt.figure(figsize=(15, 20))
tree.plot_tree(tree_model, filled=True,
               feature_names=iris.feature_names);
```

Output:

plot width = 0.8

gini = 0.663

sample = 85

values = 24, 30, 3

gini = 0.0

sample = 24

values = 24, 0, 0

petal width ≤ 1.4

gini = 0.5

sample = 61

value = 0.30.31

```
y_pred = true_model.predict(test)  
from sklearn.metrics import accuracy  
accuracy = accuracy_score(y_test, y_pred)  
print("Accuracy", accuracy)
```

Output:

Accuracy = 0.96723

It means our model is 96.72% accurate.

So, we can say our model is good.

Now, let's make a confusion matrix.

(True Positive, True Negative)

(False Positive, False Negative)

(True Positive, False Positive)

It shows the number of correct predictions.

It is also called a 2x2 matrix.

True Positive = 893

True Negative = 106

False Positive = 10

False Negative = 7

Confusion matrix is very useful.

It is also called a 2x2 matrix.

It shows the number of correct predictions.

It is also called a 2x2 matrix.

It shows the number of correct predictions.

It is also called a 2x2 matrix.

It shows the number of correct predictions.

It is also called a 2x2 matrix.

It shows the number of correct predictions.

It is also called a 2x2 matrix.

It shows the number of correct predictions.

It is also called a 2x2 matrix.

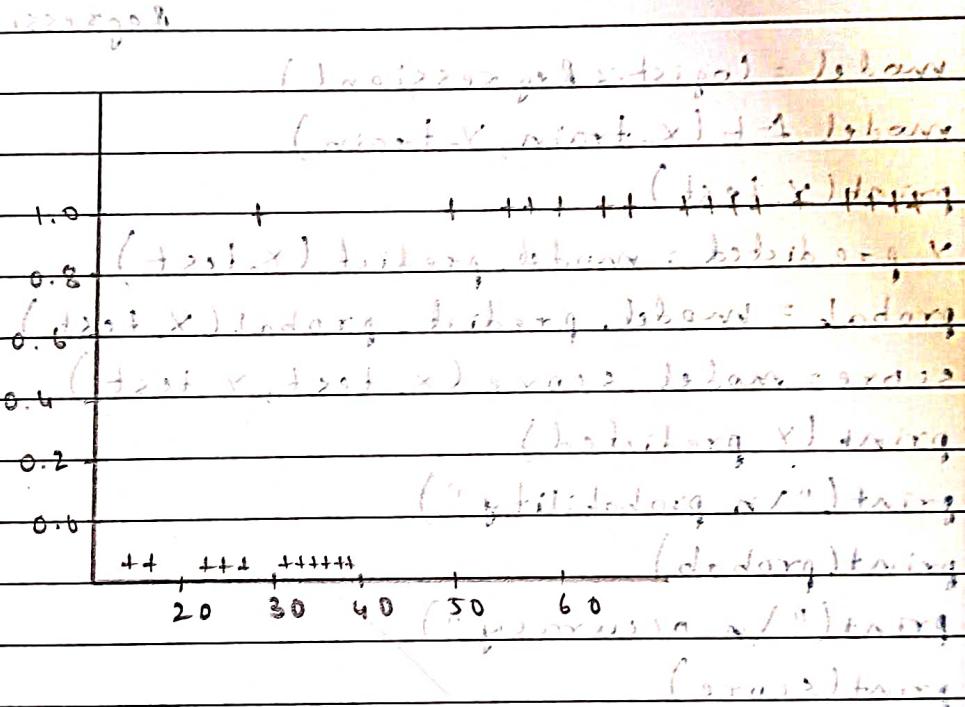
Logistic Regression with Python

```

import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
df = pd.read_csv("insurance.csv")
df.head()
plt.scatter(df['age'], df['bought_insurance'],
            marker='+', colour='red')
from sklearn.model_selection import

```

~~train_test_split~~



```

from sklearn.model_selection import
    train_test_split
x = df[['age']]
y = df['bought_insurance']

```

Perform split

```

x_train, x_test, y_train, y_test = train_test_
split(x, y, train_size=0.8)

```

Display x-test

```
print(x-test)
```

age
3 47
4 61
5 56
21 26
22 50
17 58

from sklearn.linear_model import Logistic
Regression

```
model = LogisticRegression()
```

```
model.fit(x-train, y-train)
```

```
print(x-test)
```

```
y_predicted = model.predict(x-test)
```

```
probab = model.predict_proba(x-test)
```

```
score = model.score(x-test, y-test)
```

```
print(y_predicted)
```

```
print("in probability")
```

```
print(probab)
```

```
print("in accuracy")
```

```
print(score)
```

x-test: 3 47 4 61 5 56 21 26 22 50 17 58

```
x-test:
```

age

9

61

23

45

4

46

x-test: 9 61 23 45 4 46 29 45 15 45 55 25 21 26 22 50 17 58

1

25

predicted values: [111110]

probabilities: $\begin{bmatrix} 0.09781523 & 0.90218477 \\ 0.40313939 & 0.59686061 \\ 0.37545927 & 0.62404073 \\ 0.2700642 & 0.7239358 \\ 0.86924175 & 0.13075825 \end{bmatrix}$

accuracy 1.0

~~Exercise 3: Using the word bank, complete the following sentence.~~

John
John was right to split
up with his girlfriend.

- ((local file) by me)

~~"com.apple.xbs\$-report": xbs\$-report~~

2955982155VH 10.32000 had 11

Endothelial progenitor cells (EPCs) are a subset of adult haemopoietic stem cells.

(Urgent) 09/07/11 10:18 AM

(4, p) > max yba8 "J bdaLw t

() Jan 22, 1966.

(902) 330-0401 or 902-330-0401

Because what I am most qualified to do

[Image] Ah -

~~Substrate lost most of its mass~~

~~Standards 18-19) with 2017 and 2018 in column~~

(writing, writing x) it's mine

(*fast*) *extirpation* - *deprive*

Lycoperdon "variegatum" (L.) Gray

[1] KNN

```
[ 1 ] import pandas as pd  
[ 2 ] import matplotlib.pyplot as plt  
[ 3 ] import seaborn as sns  
[ 4 ] from sklearn.model_selection import  
[ 5 ] train_test_split  
from sklearn.neighbors import  
    KNeighborsClassifier  
from sklearn.metrics import accuracy_score  
df = pd.read_csv("https://raw.githubusercontent.com  
    /palmerpenguins/master/penguins  
print(df.head())
```

```
sns.scatterplot(x="flipper-length-mm",  
    y="body-mass-g", hue="species", data=df)  
plt.title("Penguin species classification")  
plt.xlabel("Flipper length (mm)")  
plt.ylabel("Body mass(g)")  
plt.show()
```

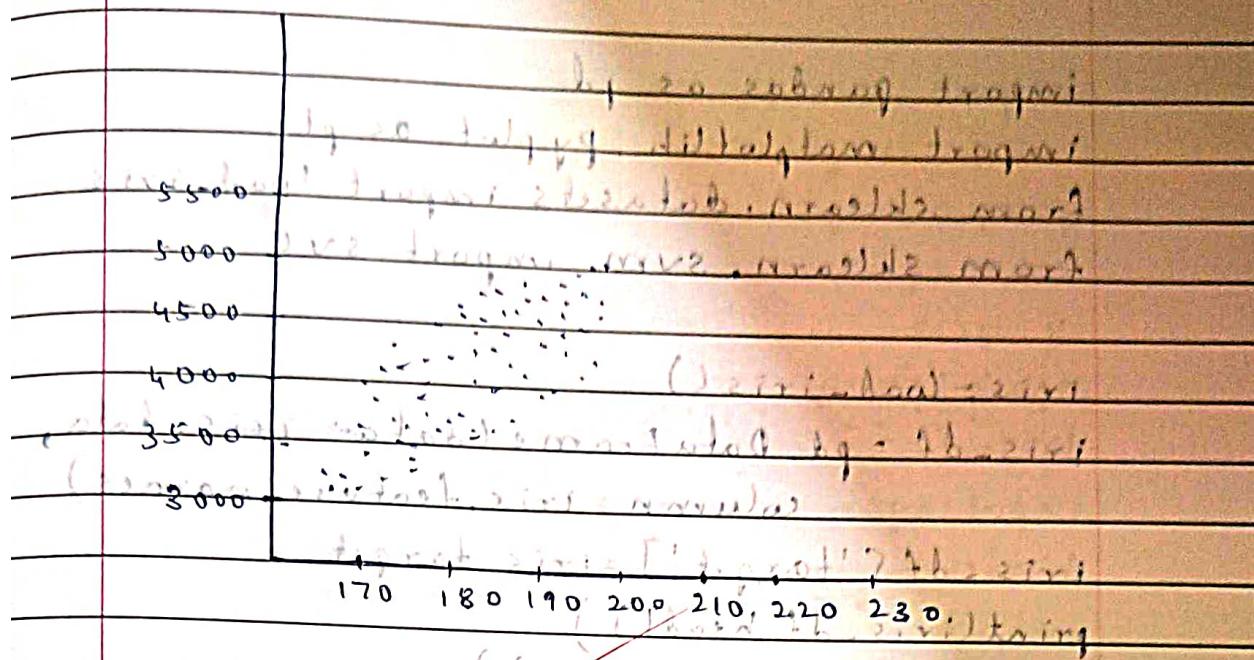
```
df.dropna(inplace=True)
```

```
X = df[['flipper-length-mm', 'body-mass-g']]  
y = df['species']  
X_train, X_test, y_train = test_size=0.2
```

```
KNN = KNeighborsClassifier(n_neighbors=3)  
KNN.fit(X_train, y_train)  
y_pred = KNN.predict(X_test)  
print("Accuracy", accuracy)
```

Output:

卷之三



~~(1) $\text{AgNO}_3 + \text{NaCl} \rightarrow \text{AgCl} \downarrow + \text{NaNO}_3$~~
~~(2) $\text{AgNO}_3 + \text{KCl} \rightarrow \text{AgCl} \downarrow + \text{KNO}_3$~~
~~(3) $\text{AgNO}_3 + \text{CaCl}_2 \rightarrow \text{AgCl} \downarrow + \text{Ca(NO}_3)_2$~~

('Constitution Logos') Index filo
('Constitution Logos') Index filo
('Constitution Logos') Index filo

Day 1 - start off by writing your first & most important
and meaningful sentence.

S.O. - a clear thought

S.O. - a clear thought

(Concurrent demand) over + or - 10%, may
Cause significant short + long term problems, may
Cause significant short + long term problems.

proposal move to permanent stage
Cassano

SVM

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.svm import SVC
```

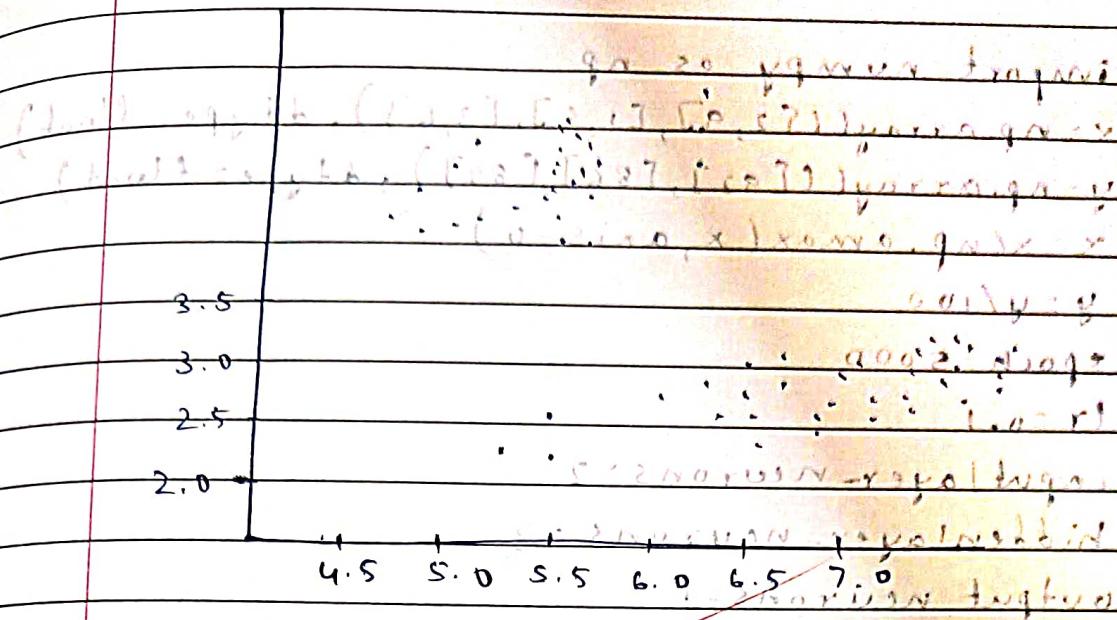
```
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data,
                        columns=iris.feature_names)
iris_df['target'] = iris.target
print(iris_df.head())
plt.figure(figsize=(10, 6))
plt.scatter(iris_df['sepal length (cm)'],
            iris_df['sepal width (cm)'],
            c=iris_df['target'],
            cmap='viridis')
```

```
plt.xlabel('sepal length(cm)')
plt.ylabel('sepal width(cm)')
plt.title('Iris dataset')
```

```
x_train, x_test, y_train, y_test = train_test_
split(iris.data, iris.target,
      test_size=0.3,
      random_state=42)
```

```
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(x_train, y_train)
y_pred = svm_classifier.predict(x_test)
print("Accuracy of SVM classifier",
      accuracy)
```

Output: takes user input and prints it in a single line.



~~negative feedback sometimes produces greater~~

~~Chlorophyll containing bacteria~~

reputation, capitalization, ~~and~~ ^{A2} ~~and~~ ^{B3} ~~and~~ ^{C4} organizational grounds.

11. 1907-8. 3rd

~~multiple additional test motions, such as run, quiet time~~

Common features observed

Ligustrum lucidum (L.) Agardh eximium var. *agrestifolium*

Monogram

Cytokinin

(x_0, x_1, \dots, x_n) is a solution.

Chloromyces sanctipetri Tab.

$$(x+1)^3 \times \text{...}$$

(d) $\frac{1}{2} \times 10^{-10} \text{ m}^2$

(Later & later get equal)

Language and gender

(and) becomes a noun or verb

(Chew, 1995) and that $\alpha = 1$ is not

Implementation of ANN using BP

```
import numpy as np
x=np.array([[2,9],[1,5],[3,6]], dtype=float)
y=np.array([[92],[86],[87]], dtype=float)
x=x/np.amax(x, axis=0)
y=y/100
epoch=5000
lr=0.1
inputlayer_neurons=2
hiddenlayer_neurons=3
output_neurons=1
wh=np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh=np.random.uniform(size=(1, hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons, output_neurons))
bout=np.random.uniform(size=(1, output_neurons))

def sigmoid(x):
    return 1/(1+np.exp(-x))

def derivatives_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinp1=np.dot(x, wh)
    hinp=hinp1+bh
    hlayer_act=sigmoid(hinp)
    outinp1=np.dot(hlayer_act, wout)
    boutinp=outinp1+bout
```

netting = netinput + bias * weight

output = sigmoid(netting)

~~for i in range(1, len(wout) + 1):~~

~~E0 = y - output[i - 1]~~

~~outgrad = derivatives.sigmoid(output[i])~~

~~d_output = E0 * outgrad~~

~~EH = d_output * dot(wout[i], T)~~

~~hiddengrad = derivatives.sigmoid(hlayer.act)~~

~~d_hiddenlayer = EH * hiddengrad~~

wout += hlayer.act.T * dot(d_output) * lr

wh += x.T * dot(d_hiddenlayer) * lr

print("Input: " + str(x))

print("Actual Output: " + str(y))

print("Predicted Output: " + str(output))

Output: [0.66667 1.]

Input: ([0.66667, 1.], 1.0)

[0.66667 1.]

[0.33333 0.55556] (floats). I.e long float

[1. 0.66667]]

Actual output: array([0.92144171]) during

[0.92144171]

[0.86144171]

[0.89144171]

Predicted output: array([0.92144171]) during

[0.78415718] (floats). I.e long float

[0.76533937] (floats)

[0.78892728]

Random Forest Algorithm

```
from sklearn.datasets import load_iris  
from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy_score, classification_report  
  
iris = load_iris()  
X = iris.data[:, :4] # we only take the first four features  
y = iris.target  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)  
  
rf_classifier = RandomForestClassifier()  
rf_classifier.fit(X_train, y_train)  
  
y_pred = rf_classifier.predict(X_test)  
  
accuracy = accuracy_score(y_test, y_pred)  
print("Accuracy:", accuracy)  
  
classification_report = classification_report(y_test, y_pred)  
print("Classification Report:\n", classification_report)
```

Output:

traindata

Classification Report:

precision recall f1 score support

f1 score 1.00 precision 1.00 recall 1.00 support 30

0.89 0.81 1.00 21

2 1.00 0.91 0.93 0.79 0.81 24

traindata

accuracy f1 score precision recall support

macro avg 0.94 0.93 0.93 75

weighted avg 0.95 0.93 0.93 75

(1+1) / 2 = 2

2 / 2 = 1

Report 2/2 - y

Total work - took a night to test & run all x

other methods, now acc. best, y, x) + edges

Total work - took a night to test & run all x

other methods, now acc. best, y, x) + edges

(calculated without

(count y, min x) + th. 2/2 - traindata

(test x) + th. 2/2 - traindata + loss 0

(loss y, test x) + loss y, precision = precision

(precision, "precision") + loss

- length

F 0.0000000000000000

0.0000000000000000

Adaboost

```

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy

```

```
iris = load_iris()
```

```
x = iris.data
```

```
y = iris.target
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=42)
```

```
adaboost_clf = AdaBoostClassifier(
    n_estimators=30, learning_rate=1,
    random_state=42)
adaboost_clf.fit(x_train, y_train)
```

```
y_pred = adaboost_clf.predict(x_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

Accuracy: 0.9666666666666667

✓
23/5/24