Dt:-
26/4/22

**** Random Forest

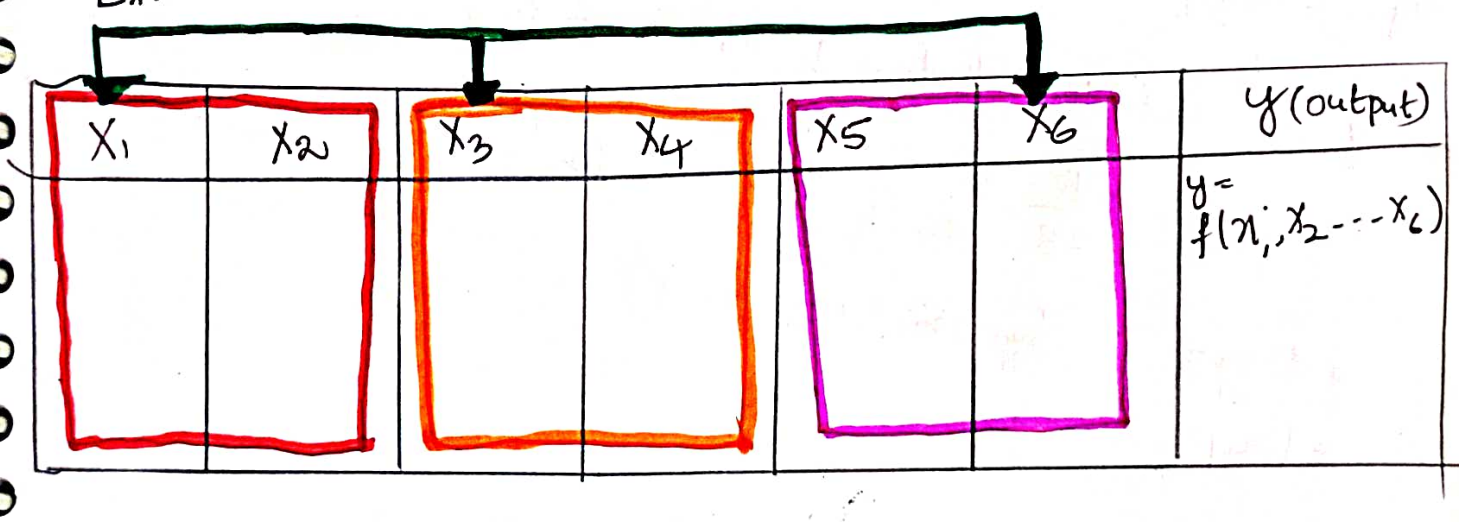
* Random forest classifier is a type of "Ensemble algorithm".

"Ensemble" → Combination of Algorithm.

* Random forest :- Number of Trees (Decision)
  (or)
  Multiple decision Trees.

* it creates a Set of Decision Tree from Randomly Selected subset of Training set.

Ex:-



* Decision Tree 1 :- X_1, X_2
* Decision Tree 2 :- X_3 X_4
* Decision Tree 3 :- X_5 X_6
* Decision Tree 4 :- X_1, X_3 X_6
* Subset of Features :- Make Decision Tree

Apply Randomly :-
"Permutation Combination" concept

Que:- Whether X_1 and X_2 are used in First Decision Tree Again in DecisionTree2 X_1 is used ?

A: Decision Tree1 , Decision Tree 2

X_1 X_2

X_1 X_3

replaced again

* with replacement : Bagging
* without replacement :

Ex:- Student - 28 → Prob: $\frac{1}{28}$

Day:1 asked Question for "Jack" : Student 1

Day:2 asked Question to student 2: removing student 1 = $\frac{1}{27}$

If we include Student 1 again in Day "2"

Prob: $\frac{1}{28}$. " Every Time Probability remains common. "

* Day3 : $\frac{1}{28}$ (including S:1)
* Day4 : $\frac{1}{28}$

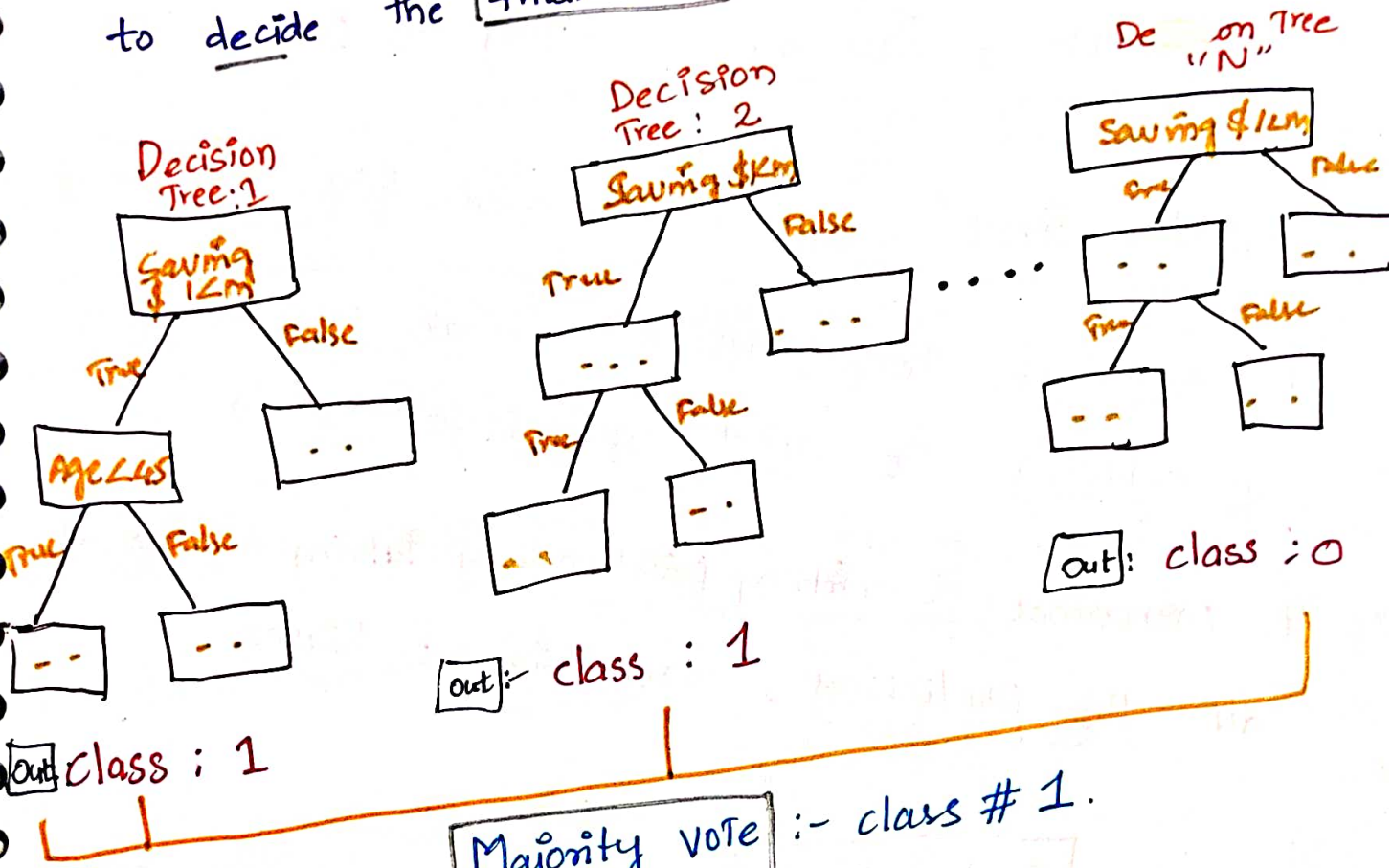
* Decision Tree 1
* Decision Tree 2.

If Decision Tree 1 has Probability

Decision Tree 2 has the Same probability.

* Backend code :- np.random.rand( )

* it then Combines Votes from different decision Trees to decide the final class of the test object.



Majority vote :- class # 1.

Random forest is also called as "Bagging" "Parallel Algorithm" Technique.

* Bagging :- B = Boostrap (data replacement) (with replacement)
agg = aggregration (combine)

* with replacement : Probability of selecting data Remains Constant for all the decision Trees. Hyper Parament (default) *bootstrap = True (wit

* bootstrap = False (without replacement) records will not use For another decision Tree

Que :- How do you reduce Overfitting problem in Decision Tree ?

A :- "Random forest"

But, with in only in Decision Tree ? How ?

Ans :- "Pruning" → cutting the Decisiontree.

"Random Forest" :- Each Trees are grown "fully" - No pruning

* it overcomes the issues with single decision trees by reducing the effect of noise. → overfitting

* it overcomes Overfitting problem by Taking average of all the predictions, cancelling out biases.

* apply on → Random Forest

Both

Regression

import Decision Tree Regressor

Math : instead of Entropy, Gini

* We Take "Variance"

classification

import Decision Tree Classifier

math : Entropy, Gini

Que :- Does Random Forest effect by Outliers ?

A :- "No" Random forest doesn't effect by outliers.

* it gives Equal opportunity for all features by selecting Randomly

and "No" normalization of data. is required.

Ex:- Combining Votes from a pool of Experts, each will bring their own Experience and background to Solve the problem resulting better outcome.

Advantages of "Random forest" (or) Bagging

* it applicable for both ✓ Classification & Regression
* Handle The Missing Values and Maintains Accuracy for Missing data. (drop them)
* it won't overfit the MODEL
* Handle large dataset with higher dimensionality
  Ex: 100 columns → Radomly selected.

Que:- How many Decision Trees ? Taken ?

Ans: By Default, it Takes "100" Decision Tree again, in this which is best decision Tree, we do "Hyper Parameter Tuning" → Grid Search CV

CODE :-

```
import Numpy as np
import Pandas as pd
import matplotlib.pyplot as plt
import Seaborn as sns
```

Ex:- In any Project
1. Accuracy (parameter)
2. less Computational Time

→ Same Data Set (Decision Tree) to check difference b/w (D.T)(R.F)

```
# df= pd.read_csv("penguins_size.csv")
# df.head()
```

Out

	Species	Island	culmen length	culmen depth mm	flipper length mm	body mass g	Sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NAN	NAN	NAN	NAN	NAN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

```
# df.info()
# df.isnull().sum()
```

EDA

```
# df = df.dropna()
# df.shape
```

Feature Engneering

```
# pd.get_dummies(df.drop("species", axis=1), drop_first=True)
```

X & Y

```
# x = pd.get_dummies(df.drop("species", axis=1), drop_first
                                          = True)

# y = df["species"]
```

Train Test Split

```
from sklearn.model_selection import train_test_split

# x_train, xtest, y_train, y_test = train_test_split(x,y, test_siz
                                                    =0.3, random_state
```

MODELLING :- Random Forest Classifier

```
# import classifier
from sklearn.ensemble import RandomForestClassifier.

# model = RandomForestClassifier()

# model.fit(x_train, y_train)
```

Out: Random Forest Classifier.

Prediction

```
# ypred_train = model.predict(x_train)

# ypred_test = model.predict(x_test)
```

Evaluation

```
from sklearn.metrics import accuracy_score

# print ("train_Accuracy" : , accuracy_score (ypred_train, y_train)
# print ("Test_Accuracy": , accuracy_score (ypred_test, y_test)
```

Out : Train_Accuracy : 1.0
Test_Accuracy : 0.99

Cross Validation.

```
from sklearn.model_selection import cross_val_score

# scores = cross_val_score (model, x, y, cv=5)
# print (scores)
# scores.mean ()
```

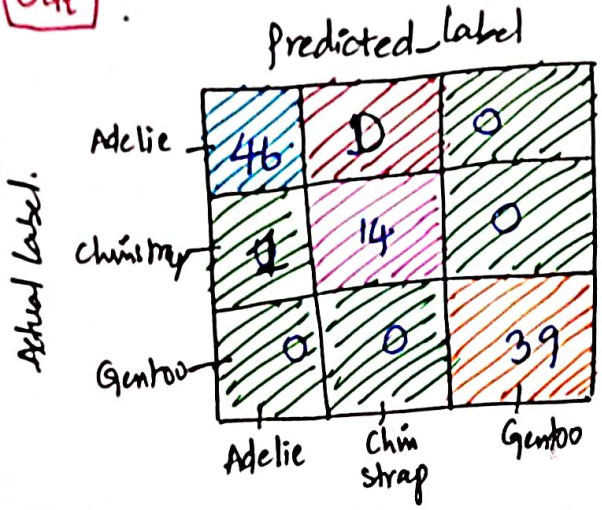
Out : [1. 0.98 , 0.98 , 0.98 , 1.]

0.9909

```
from sklearn.metrics import plot_confusion_matrix

# plot_confusion_matrix (model, X_test , y_test)
# plt.show ()
```

Out :




```
from Sklearn.metrics import classification_report
# print (classification_report (y_test , ypred_test)
```

Out

	Precision	recall	f1 score	support
Adelie	0.98	1.00	0.99	40
Chinstrap	1.00	0.96	0.98	27
Gentoo	1.00	1.00	1.00	33
accuracy			0.99	100
macro Avg	0.99	0.99	0.99	100
weighted Avg	0.99	0.99	0.99	100

Feature importance

```
# model. feature_ importances_
```

Out : array ([0.336 0.161 0.283, 0.112 , 0.076 , 0.022 0.006)]

Applying....

Hyper Parameter Tuning.

```
from sklearn.model_selection import GridSearchCV

#model
# estimator = RandomForestClassifier()
# parameter (which you want to tune and identify the best)
# param_grid = {"n_estimators": list(range(1,101))}

# grid = GridSearchCV(estimator, param_grid, scoring="accuracy", cv=5)

# grid.fit(x_train, y_train)
```

Out: GridSearchCV(cv=5, estimator = RandomForestClassifier(),
param_grid = {"n_estimators": [1,2,3,4,5,6,7,8,9,10,
11,12.....]},
scoring="accuracy")

```
# grid.best_params_
```

out: {"n_estimators": 21}

Final Model with best parameter

#modelling

```
# final_model = RandomForestClassifier(n_estimators=21
                , Random_state=29)

# final_model.fit(x_train, y_train)
```

Out : RandomForestClassifier(n_estimator=21, random_state=29)

#prediction

```
# Predictions = final_model.predict(x_test)
```

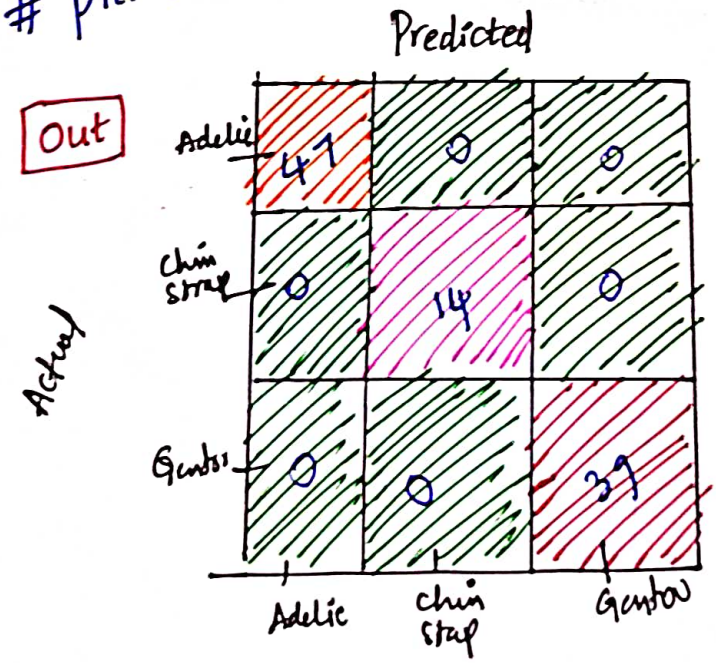
#Evaluation

```
# accuracy_score(predictions, y_test)
```

Out 1.00

```
# plot_confusion_matrix(final_model, x_test, y_test)
# plt.show()
```

Out



26/4/22
4:00pm