JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

May '22

Wednesday 18

Week 21 Day (138-227)

09.00 — DECISION TREES —

- TREE BASED MODELS CAN BE USED FOR REGRESSION AND

10.00 CLASSIFICATION.

- THEY INVOLVED IN DIVIDING THE PREDICTIONS SPACE INTO A

11.00 NUMBER OF REGIONS. THE SET OF SPLITTING RULES CAN BE SUMMARIZED IN A TREE.

12.00 → HENCE IT IS CALLED DECISION TREE.

★ A SINGLE DECISION TREE IS OFTEN NOT BETTER THAN LINEAR REGRESSION

13.00 LOGISTIC REGRESSION OR LDA.

★ BAGGING, RANDOM FOREST AND BOOSTING DRAMATICALLY IMPROVE

14.00 PERFOMANCE.

15.00 BOSIC TERMINOLOGIES

→ Trees are drawn upside down.

16.00 → The final regions are called leaves. The point where split occurs is called a node.

17.00 → The segement that nodes are (The segement that connects the nodes are called BRANCHES).

18.00

HOW REGRESSION TREAT WORKS →

On next page →

19 '22 May

Week 21 Day (139-226) Thursday

MAY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

09.00 TO CREATE A REGRESSION TREE

- Divide the predictor space into 'J' distinct and non-overlapping regions
- For every observation that falls in a region, predict the mean of the response value in that region.
- Each REGION IS SPLIT TO MINIMISE THE RSS (Residual sum of squares)
- It uses a top-down greedy approach called recursive binary splitting.

TOP DOWN GREEDY APPROACH -

- TOP DOWN - ALL OBSERVATIONS ARE IN A SINGLE REGION BEFORE THE FIRST SPLIT.
- GREEDY - THE BEST SPLIT OCCURS AT A PARTICULAR STEP TO MAKE THE BEST PREDICTION AT THAT STEP, RATHER THAN LOOKING FORWARD AND MAKING A SPLIT THAT WILL GIVE A BETTER RESULT IN A FUTURE STEP.

Regression Tree

$R_1(j,s) = \{X | X_j < s\}$ and $R_2(j,s) = \{X | X_j \geq s\}$

we seek j and s to minimise:

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Pray as if everything depe

09.00 Regression Tree in Python

```
From sklearn.tree import
DecisionTreeRegressor
clf = DecisionTreeRegressor()
clf.fit(X_train, y_train)
```

10.00

11.00

12.00

13.00

CLASSIFICATION TREE

14.00 - Predict the most commonly occuring class in a region.

- RSS cannot be used, each split must minimize the classification

15.00 error rate.

16.00 CLASSIFICATION ERROR RATE.

17.00 $$E = 1 - \max(\hat{p}_{mk})$$

18.00

- Not sensitive enough for Tree growing
- So instead we use Gini Index

21 '22 May

Saturday

Week 21 Day (141-224)

MAY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

09.00 GINI INDEX - MEASURE OF TOTAL VARIANCE ACROSS ALL CLASSESS.

10.00

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$$

11.00 - Small if proportion is close to 0 or 1.

12.00 - Good measure of node purity.

13.00

CROSS-ENTROPY - IT CAN ALSO BE USED FOR TREE GROWING.

14.00

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

15

16

17

Sunday 22

18.00

Classification tree in Python

```
From sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, Y_train)
```

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

May '22

Monday 23

Week 22 Day (143-222)

09.00 ADVANCE TOPICS ON DECISION TREE.

- BAGGING - BOOTSTRAP AGGREGATION

10.00 - BOOTSTRAP CAN COMPUTE STANDARD DEVIATION OF ANY QUANTITY. FOR DECISION TREES, VARIANCE TENDS TO BE HIGH.

11.00 - BAGGING CAN REDUCE THE VARIANCE AND IMPROVE THE PERFOMANCE OF A DECISION TREE.

12.00 - REPEATEDLY DRAW SAMPLES FROM THE DATASET, GENERATING B DIFFERENT BOOTSTRAP TRAINING SETS.

13.00 - TRAIN ON ALL BOOTSTRAPPED TRAINING SETS TO GET A PREDICTION FOR EACH SET AND AVERAGE THE PREDICTION.

14.00

BAGGING -

15.00

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

16.00 - we can construct high number of trees with high variance and low bias.

17.00 - Then, average the prediction to reduce variance and improve perfomance.

18.00

BAGGING IN PYTHON

```
from sklearn.ensemble import BaggingClassifier
bagging_clf = BaggingClassifier()
bagging_clf.fit(X_train, y_train.ravel())
```

24 '22 May

Week 22 Day 144-221

Tuesday

MAY 2022

09.00 RANDOM FOREST -

- RANDOM FOREST IMPROVES UPON BAGGING, BY MAKING
10.00 A SMALL TWEAK THAT DECORRELATES THE TREES.
- MULTIPLE DECISION TREES ARE BUILT. AT EACH SPLIT,
11.00 ONLY A RANDOM SAMPLE OF 'm' PREDICTORS IS CHOSEN
FROM ALL 'p' PREDICTORS.
12.00 - THE SPLIT IS ONLY ALLOWED TO USE ONE OF THE 'm'
PREDICTORS.

13.00

Random forest -

$m = \sqrt{p}$

14.00

15.00

16.00

- IN BAGGING, IF THERE IS A STRONG PREDICTOR, IT WILL LIKELY
17.00 BE THE TOP SPLIT, AND ALL TREES WILL BE SIMILAR.
- VARIANCE WILL NOT BE REDUCED

18.00

- RANDOM FOREST - ONLY A SUBSET OF PREDICTORS IS FORCED FOR EACH SPLIT.
- If $m = p$, then it is like bagging

Aim to perform to the best of your ability.

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

May '22

Wednesday 25

Week 22 Day (145-220)

09.00 RANDOM FOREST IN PYTHON

10.00

```
from sklearn.ensemble import
RandomForestClassifier
random_clf = RandomForestClassifier(100)
random_clf.fit(X_train, y_train.ravel
())
```

11.00

12.00

13.00

14.00

15.00

16.00

BOOSTING

17.00 - SIMILAR TO BAGGING, BUT TREES ARE GROWN SEQUENTIALLY

EACH TREE USES INFORMATION FROM THE PREVIOUSLY

18.00 GROWN TREES.

- LEARNS SLOWLY
- EACH TREES FIT THE RESIDUAL INSTEAD OF TARGET VARIABLE.

EACH TREE IS SMALL AND WILL SLOWLY IMPROVE THE PREDICTIONS:

- 3 TUNING PARAMETER FOR BOOSTING -

→ NO. OF TREES (B) - BOOSTING CAN OVERFIT IF B IS TOO LARGE. USE CROSS VALIDATION.

26 '22 May

Week 22 Day (146-219) | Thursday

MAY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

09.00 - SHRINKAGE PARAMETER (α) - small positive number that controls the learning rate.

10.00 Typically 0.01 or 0.001

- NUMBER OF SPLITS IN EACH TREE (d) - controls the

11.00 complexity of the boosted ensemble. A Single split works well ($d=1$) Also called interaction depth.

12.00

BOOSTING IN PYTHON-

```
from sklearn.ensemble import
Gradient-Boosting Classifier
boost_clf = Gradient-Boosting Classifier (
learning_rate = 0.01, n_estimators = 150,
max_depth = 2)
boost_clf.fit(X_train, y_train.ravel())
```

The tongue is but three inches long, yet it can kill a man six feet high.

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

May '22

Friday 27

Week 22 Day (147-218)

09.00 - DECISION TREES PYTHON HANDS ON

10.00

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import plot_confusion_
matrix
% matplotlib inline
```

11.00

12.00

13.00

14.00

15.00

16.00

numpy - Library for scientific computing in python.

17.00 pandas - Library that provides data structure for efficient data manipulation and analysis.

18.00 matplotlib - Library provides variety of visualization

seaborn - Library provides high level interface for creating attractive graphics.

sklearn.metrics.plot_confusion_matrix - function allow to plot confusion matrix.

% matplotlib inline - The magic command.

28 22 May

Week 22 Day (148-217)

Saturday

MAY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

09.00

IMPORTING THE DATASET.

```
datapath = 'url'
data = pd.read_csv(datapath)
data.head()
```

10.00

11.00

12.00

13.00

14.0

15.0

16.

17.

18.

```
def violin_plots(x, y, data):
    for i, col in enumerate(y):
        plt.figure(i)
        sns.set(rc={"figure.figsize":
        (11.7, 8.27)})
    ax = sns.violinplot(x=x, y=col, data=data)
y = data.columns[:-1]
x = data.columns[-1]

violin_plots(x, y, data)
```

Sunday 29

x is a string that represent column name of independent variable in the dataset.

y is list of strings

data is pandas dataframe.

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

May 22 30
Monday

09.00 function iterates over columns in y and create a seperate figure for each column. it sets figure size using sns.set(rc={"figure.figsize": (11.7, 8.27)}) and creates

10.00 a seaborn violin plot using sns.violinplot(x=x, y=col,

11.00 data=data), where col is the current dependant variable in the iteration. Finally returns the violin plots

12.00

```
for col in data.columns:
print(f"{col}: {data[col].
isnull().sum()}")
```

13.00

14.00

15.00

16.00 code iterates over columns in data dataframe using for a loop for each coloumn, it prints the caumn name and

17.00 number of missing values in the column using the print function and f-string. the number of missing

18.00 values is computed using isnull() method, which returns a boolean mask indicating whether each value in the column is missing or not., and sum() method which sums up the number of missing values in the column.

MAY 2022

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

```
from sklearn.preprocessing import
 LabelEncoder                                    — 1
le = LabelEncoder()
data['Classification'] = le.fit_transform       — 2
(data['classification'])
data.head()                                     — 3
```

Label Encoder class from sciKit learn Library to encode a
16.00 categorical variable in pandas dataframe.

1) converts categorical variables into numeric labels
17.00 2) le is instance of LabelEncoder class
3) Classification is a string that represent column
18.00 name of categorical variable.

→ Method first fits the encoder to the unique values in the column and then transform the column by replacing each unique value with corresponding integer label.

→ the resulting encoded values are then stored back in Classification column of data using statement data (3) statement.

09.00
10.00
11.00
12.0
13.0
14.0
15.0

```
from sklearn.model_selection
import train_test_split
y = data['Classification'].values.reshape
(-1,1)
X = data.drop(['Classification'], axis=1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.1,
random_state= 42)
```

1) Split data into training and testing

16.00 2) Classification is string that represent the column name of the target variable in data dataframe.

17.00 y contains values of target variable, reshaped into a 2D array using reshape method.

18.00 X is pandas dataframe that contains feature variable of the dataset, with target variable column dropped using the drop method and axis parameter set to 1. (column-wise)

test size is float that represent proportion of the dataset to include in testing_set.

random_state is integer that sets the seed for the random number generator used to split the dataset.

Code is useful for preparing dataset for Machine Learning

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00

- Baseline Model -

10.00

```
from sklearn.tree import DecisionTree
Classifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
plot_confusion_matrix(clf, X_test,
y_test, cmap = plt.cm.Blues)
plt.grid(False)
plt.show()
```

11.00

12.00

13.00

14.00

15.00

16.00

DecisionTreeClassifier is class from sklearn.tree module that implements a decision tree classifier.

17.00

clf is an instance of DecisionTreeClassifier class

18.00

X_train and y_train are numpy arrays that contain the feature and target variable of training set.

X_test and y_test — testing set.

cmap - colormap is used to color cells of confusion matrix.

plt.grid(False) turns off the grid lines in the plot.

```
from sklearn.tree import plot_tree
plot_tree(clf, max_depth = 5)

// plot_tree function is scikit-learn
library to visualize decision tree
classifier.
// clf is an instance that has been trained
on dataset.
// max_depth integer that specifies the
maximum depth of tree to be plotted
```

Code uses the plot_tree function to plot decision trees classifier using maximum depth of 5. Result shows each node representing decision based on feature variables and each leaf node representing predicted target variable. Sunday 05
The plot also displays impurity score and number of samples at each node.

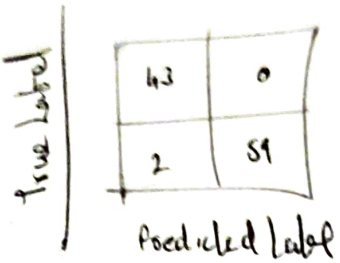
Code is useful for visualizing the decision making process of a decision tree classifier, and can help to identify patterns and insights in the data that classifier is using to make predictions.

06 '22 June

Monday

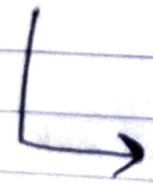
BAGGING -

```
from sklearn.ensemble import BaggingClassifier
bagging_clf = BaggingClassifier()
bagging_clf.fit(X_train, y_train.ravel())
plot_confusion_matrix(bagging_clf,
X_train, y_train, cmap = plt.cm.Blues)
plt.grid(false)
plt.show()
```



Explanation ipynb file

Random Forest Classifie



```
from sklearn.ensemble import RandomForestClassifier
random_clf = RandomForestClassifier(100)
random_clf.fit(X_train, y_train.ravel())
plot_confusion_matrix(random_clf,
X_test, y_test, cmap = plt.cm.Blues)
plt.grid(false)
plt.show()
```

Beware of little

09.00 BOOSTING-

```
from sklearn.ensemble import
Gradient-BoostingClassifier
boost_clf = GradientBoostingClassifier()
boost_clf.fit( X_train, Y_train.ravel())
plot_confusion_matrix (boost_clf, X_test,
y_test, cmap = plt.cm.Blues)
plt.grid (false)
plt.show ()
```

10.00
11.00
12.00
13.00
14.00
15.00
16.00 Explanation in Source file.
17.00
18.00