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					

April '22

Tuesday 12

Week 16 Day (102-263)

09.00 # CLASSIFICATION

10.00 - SIMPLE CLASSIFICATION - BINARY CLASSIFICATION

- CLASSIFY IN MORE THAN 2 CLASSESS

11.00

EXAMPLE- SPAM OR NOT SPAM

12.00 FRAUD TRANSACTION OR NOT

EYE COLOR (BLUE, GREEN, BROWN)

13.00

REGRESSION V/S CLASSIFICATION

14.00 - REGRESSION - RESPONSE VARIABLE IS QUANTITATIVE

- CLASSIFICATION - RESPONSE VARIABLE IS CATEGORICAL

15.00 OR QUALITATIVE.

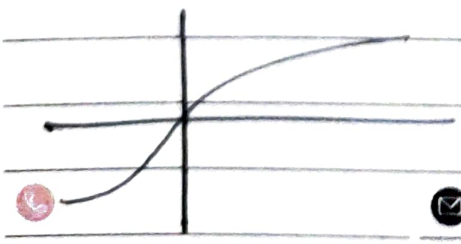
16.00 ## PERFORMING CLASSIFICATION WITH LOGISTIC REGRESSION

- DURING LOGISTIC REGRESSION WE WANT TO DETERMINE

17.00 THE PROBABLITY OF AN OBSERVATION TO BE PART OF A CLASS OR NOT.

18.00 - OUTPUT BETWEEN 0 AND 1 (1 MEANS VERY LIKELY)

THE FUNCTION TO DO THAT - SIGMOID FUNCTION



$$p(x) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

* HERE WE ARE ASSUMING ONLY ONE PREDICTOR 'X'.

APRIL 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

$$\frac{p(x)}{1-p(x)} = e^{\beta_0 + \beta_1 X}$$

10.00

take log both side

11.00

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1$$

12.00

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + \beta_1 X_2 + \text{---} + \beta_p X_p$$

13.00

In Python, Logistic Regression can be performed like this –

14.00

15.00

```
logistic_reg = LogisticRegression()
```

16.00

```
logistic_reg.fit(X_train, y_train.ravel())
```

```
y_prob = logistic_reg.predict_proba
         (X_test)[:,1]
```

17.00

18.00

```
y_pred = np.where(y_prob > 0.5, 1, 0)
```

MAY 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	31					

April '22

Thursday 14

Week 16 Day (104-261)

09.00 LINEAR DISCRIMINANT ANALYSIS (LDA)

10.00 CAVEATS (WARNING) OF LOGISTIC REGRESSION

- WHEN CLASSESS ARE WELL SEPERATED, PARAMETERS OF

11.00 LOGISTIC REGRESSION ARE UNSTABLE.

- UNSTABLE FOR SMALL DATASETS

12.00 - NOT THE BEST TO PREDICT MORE THAN 2 CLASSESS.

- LOGISTIC REGRESSION CAN ONLY BE USED FOR BINARY

13.00 CLASSIFICATION.

↳ BUT WITH LDA WE CAN OVERCOME THESE ISSUES BECAUSE

14.00 • MODELS THE DISTRIBUTION PREDICTORS SEPERATELY FOR

15.00 EACH CLASS.

• USE BAYE'S THEOREM TO ESTIMATE THE PROBABLITY

16.00

BAYES THEOREM FOR CLASSIFICATION

17.00 SUPPOSE WE WANT

$$Pr(Y=k|X=x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

18.00 TO CLASSIFY AN OBSERVATION INTO ONE OF 'K' CLASS

Let $K \geq 2$, then

- π_k is overall probability
- f_k is density function
- if $X=x$, f_k is large

15 '22 April
Week 16 Day (105-260) Friday

APRIL 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 CHALLENGE IS APPROXIMATING THE DENSITY FUNCTION

10.00 → LDA FOR ONE PREDICTOR

11.00 $$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(\frac{-1}{2\sigma^2}(x-\mu_k)^2\right)$$

12.00

it we plot the upper function in the previous formula,

13.00 and then take log, we get —

14.00 $$\underbrace{\delta_k(x)}_{\text{discriminant}} = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

15.00

WHEN WE APPLY LDA, WE NEED TO BE AWARE OF

16.00 ASSUMPTIONS —

- EACH CLASS IS DRAWN FROM A GAUSIAN DISTRIBUTION

17.00 - EACH CLASS HAS ITS OWN MEAN

- ASSUME A COMMON VARIANCE

18.00 - ASSUME A COMMON COVARIANCE MATRIX

LDA IN PYTHON →

(ON NEXT PAGE)*

09.00 LDA IN PYTHON

10.00
```
lda = LinearDiscriminantAnalysis()
```
11.00
```
lda.fit(X_train, y_train.ravel())
```
12.00
```
y_prob_lda = lda.predict_proba
             (X_test)[:,1]
```
13.00
```
y_pred_lda = np.where(y_prob_lda
             > 0.5, 1, 0)
```
14.00
15.00
16.00

QUADRATIC DISCRIMINANT ANALYSIS (QDA)

17.00 ASSUMPTIONS OF QDA

Sunday 17

- EACH CLASS IS DRAWN FROM A MULTIVARIATE GAUSIAN

18.00 DISTRIBUTION

- EACH CLASS HAS ITS OWN VECTOR (MEAN VECTOR)
- EACH CLASS HAS ITS OWN COVARIANCE MATRIX

18 '22 April

Week 17 Day (108-257) Monday

APRIL 2022

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

09.00 DISCRIMINANT WITH QDA

10.00 $\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) + \log(\pi_k)$

11.00 ADVANTAGE OF QDA

- BETTER FOR LARGE DATASET

12.00 - LOWER BIAS, HIGHER VARIANCE

13.00 QDA IN PYTHON

14.00

```
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train.ravel())

y_prob_qda = qda.predict_proba(X_test)
             [:,1]
y_pred_qda = np.where(y_prob_qda >
             0.5, 1, 0)
```

15.00

16.00

17.00

18.00

09.00 CLASSIFICATION

HOW TO ASSESS THE PERFOMANCE OF A MODEL

10.00

- SENSTIVITY - TRUE POSITIVE RATE. PROPORTION OF ACTUAL

11.00 POSITIVE IDENTIFIED.

EX- PROPORTION OF FRAUDENT TRANSACTION THATARE

12.00 ACTUALLY FRAUDULENT.

13.00 - SPECIFICITY - TRUE NEGATIVE RATE. PROPORTION OF ACTUAL NEGATIVES IDENTIFIED.

14.00 EX- PROPORTION OF NON- FRAUDULENT TRANSACTIONS THAT ARE ACTUALLY NON- FRAUDULENT.

15.00

WE CAN ALSO USE ROC CURVE (RECIEVER OPERATING CHARACTERSTIC)

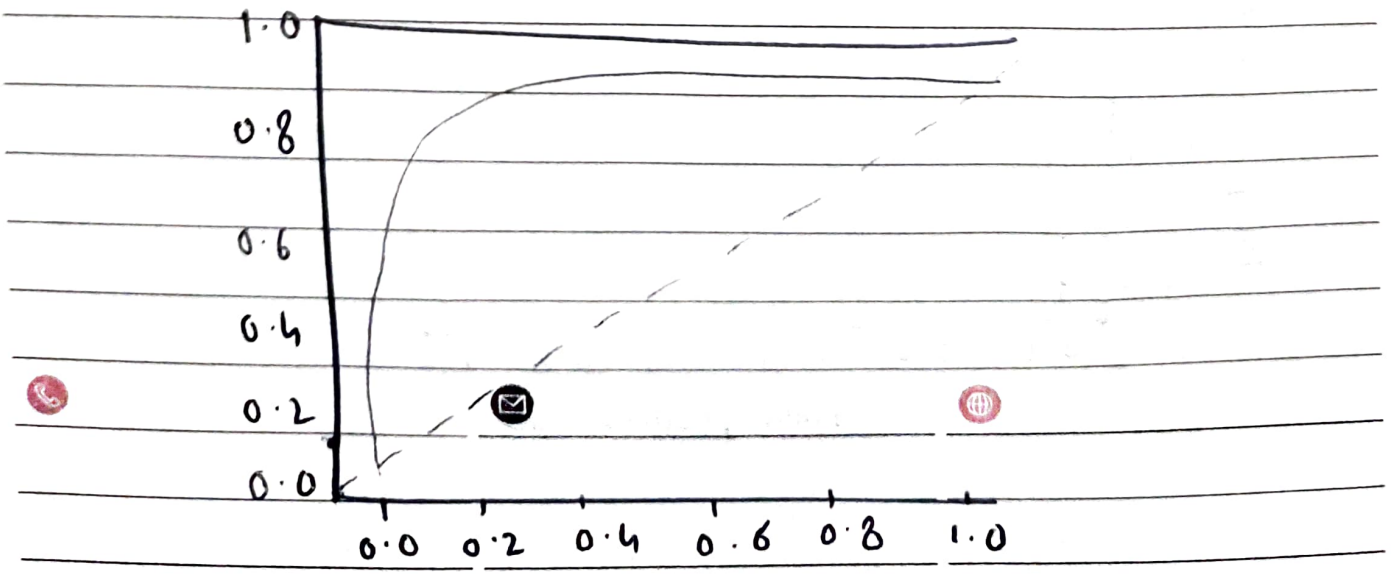
16.00 - TAKE AREA UNDER THE CURVE (AUC)

- THE CLOSER TO 1, THE BETTER IT IS

17.00

ROC CURVE

18.00

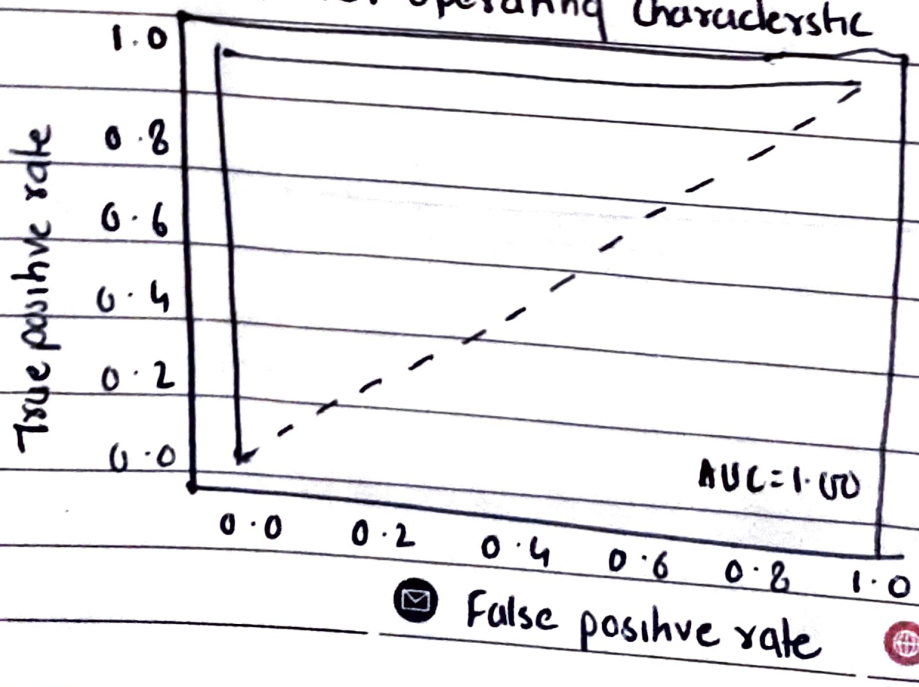


09.00

ROC IN PYTHON

```
def plot_roc (roc_auc):
plt.figure (figsize = (7,7))
plt.title ('Reciever Operating Characteristic')
plt.plot (false_positive_rate,
          true_positive_rate,
          color='red', label='AUC = %.0.2f'
           % roc_auc)
plt.legend (loc='lower right')
plt.plot ([0,1], [0,1], linestyle='--')
plt.axis ('tight')
plt.ylabel ('True positive rate')
plt.xlabel ('False positive rate')
```

ROC IN PYTHON (OUTPUT)



09.00 - HANDS ON IN PYTHON -

10.00 CLASSIFICATION MODEL # CODEDANIEL

11.00
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import
LabelEncoder
from sklearn.model_selection import
train_test_split, cross_val_score
from sklearn.metrics import roc_curve,
auc, confusion_matrix
%matplotlib inline
```

12.00

13.00

14.00

15.00

16.00

17.00

pandas - Library for data manipulation and analysis.
18.00 function for importing & exporting data from various file formats.

numpy - functions for working with arrays and matrices.

matplotlib - plotting library for creating interactive visualization

seaborn - Library built on top of matplotlib for visualization

sklearn - Library for Machine Learning

22 '22 April

Week 17 Day (112-253) Friday

09.00 LabelEncoder - class from sklearn preprocessing that transforms categorical data into numerical data by assigning unique integer to each categ

10.00 train_test_split - function from sklearn.model that splits class into training and testing sets

11.00 cross_val_score - perform cross validation, a technique of model by splitting dataset into multiple training & testing sets.

12.00 roc_curve and auc - function that calculates the operating characteristic (ROC) curve and the area under curve (AUC), which is used to evaluate performance of binary classifiers.

13.00 confusion_matrix - computes confusion matrix.

14.00

15.00

16.00

```
datapath = 'url/path'
data = pd.read_csv(datapath)
```

17.00

18.00

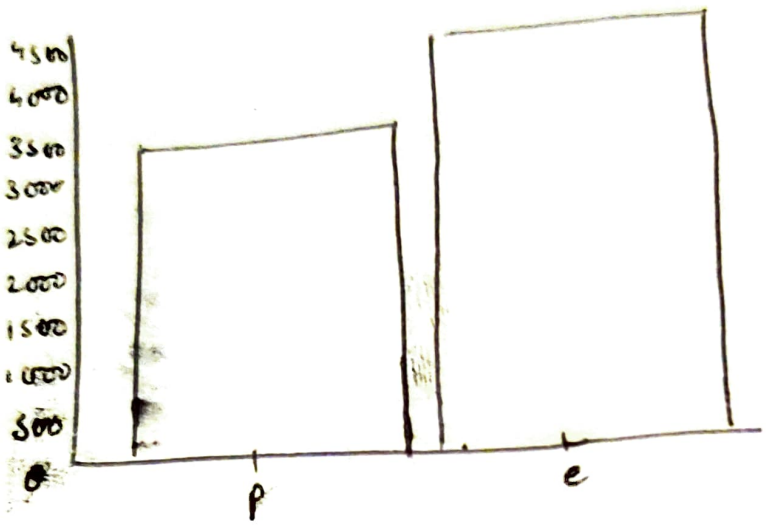
↳ additionally for seeing

```
data.head()
```

23 24 25 26 27 28 29 30

09.00

```
X = data['class']
ax = sns.countplot(x=X, data=data)
```



10.00

11.00

12.00

13.00

14.00

15.00

16.00 1) extracts the coloumn of data with the label 'class' from pandas dataframe data and assign it to variable X.

17.00 2) it creates a countplot using the Seaborn Library Sunday 24
which is barplot that shows the count of each unique value

18.00 X parameter specifies the categorical variable to be plotted which in this case X variable was previously extracted from the data dataframe

data parameter specifies thato dataframe that the categorical variable is coming from, which is also data in this case.

resulting plot is stored in variable ax, which can be used to further customize the plot.

```
def plot_data(hue, data):
    for i, col in enumerate(data.columns):
        plt.figure(i)
    ax = sns.countplot(x=data[col],
                       hue=hue,
                       data=data)
```

plot_data takes 2 parameters -

hue- string specifying the name of column to be used for coloring the countplot. This the column that contains the categorical variable that will be used to group data and color the bars in the plot.

data- pandas dataframe contains data to be plotted

data dataframe- create seperate countplot with specified hue parameter.

enumerate function - get index of each coloumn into data dataframe which is used to create seperate figure for each plot. (i.e plt.figure(i))

sns.countplot is used to create countplot, with X parameter set to the name of coloumn (col) and the hue parameter set to the specified hue parameter.

09.00 Resulting plot for each column is stored in variable ax.
but it is not returned by the function. If function is called,
10.00 it will simply create series of countplot in the dataframe with
specified hue parameter

11.00

12.00

13.00

14.00

15.00

16.00

17.00

```
hue = data ['class']
data_to_plot = data.drop ('class', axis=1)
plot_data (hue, data_to_plot)
```

— 1
— 2
— 3

18.00 1) The code creates new variable named hue that extracts the coloumn of data with Label 'class' from pandas dataframe data and assigns it to hue. This column is used for coloring the countplots.

2) New dataframe named data_to_plot drops column with Label 'class' from the data dataframe. and assigns it to data_to_plot. This new dataframe contain all of columns from orignal data dataframe except 'class' column.

3) This will create series of countplots, one for each column in data_to_plot with 'class' column used for coloring the bars in plot.

09.00 PREPROCESSING -

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

10.00

11.00

12.00

13.00 Code loops through each column in the pandas dataframe data and for each column it prints out the number

14.00 of missing value using the isnull() method to find missing values and sum() method to count them.

15.00

```
le = LabelEncoder()                    — 1
data['class'] = le.fit_transform       — 2
                (data['class'])
data.head()
```

16.00

17.00

18.00

1) create an instance of LabelEncoder class & assign it to variable le
2) the use fit_transform method of LabelEncoder object to encode the value in the 'class' column of pandas dataframe data. fit_transform method fits the encoder to data and transform the data in single step.
It replaces values in 'class' with encoded integer value.

09.00 MODELLING —

10.00
```
y = data['class'].values.reshape(-1,1)                    — 1
X = encoded_data.drop(['class'], axis=1)                  — 2
X_train, X_test, y_train, y_test =                        — 3
  train_test_split(X, y, test_size = 0.2,                 — 4
                   random_state = 42)
```

11.00

12.00

13.00

14.00

15.00

16.00

1) first creates a variable by selecting the "class" column from pandas dataframe using indexing, and then reshaping it using the reshape method of numpy array. The reshape method is used to convert 1D array to 2D array with single column. Because scikit learn expects the target variable to be in 2D array format.

17.00

2) creates a variable X by dropping class column from pandas dataframe encoded-data using drop method with axis parameter set to 1. new dataframe without 'class' column.

18.00

3) 20 % Testing 80 % Training

Random State is set to 42 to ensure reproducibility

09.00 LOGISTIC REGRESSION –

```
From sklearn.linear_model import
LogisticRegression                                  — 1
logistic_reg = LogisticRegression()                 — 2
logistic_reg.fit(X_train, y_train.ravel())
y_prob = logistic_reg.predict_proba                 — 3
        (X_test)[:,1]
y_pred = np.where(y_prob > 0.5, 1, 0)               — 4

log_confusion_matrix = confusion_matrix
                       (y_test, y_pred)             — 5
log_confusion_matrix
```

1) It provides variety of linear models for use in regression and classification

2) Create instance of LogisticRegression class and assign it to variable logistic_reg , ravel method flatten to 1D array

3) code uses predict_proba to predict probability of positive class , predict_proba returns 2D numpy array where each row corresponds to class. Class1 → 0 , Class2 → 1

4) np.where function convert the predicted probabilities in y_prob to binary predictions (y_pred) by applying threshold

5) it calculates confusion matrix for LR (Logistic Regression)

09.00
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00

```
false_positive_rate, true_
positive_rate, threshold =
roc_curve(y_test, y_prob)
roc_auc = auc(false_positive
_rate, true_positive_rate)
roc_auc
```

CODE CALCULATE RECIEVING OPERATING CHARACTERSTIC (ROC) curve for Logistic Regression Model.
ROC curve is a graphical representation of tradeoff blw TPR & FPR. for different threshold of binary classifier.

```
def plot_roc(roc_auc):
    plt.figure(figsize=(7,7))
    plt.title('Reciever Operating Characteristic')
    plt.plot(false_positive_rate, true_positive
    _rate, color='red', label='AUC = %0.2f'
    % roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1], linestyle='--')
    plt.axis('tight')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
plot_roc(roc_auc)
```

Funchon used to define properties of GRAPH

Same will be applied to LDA & QDA with slight changes.