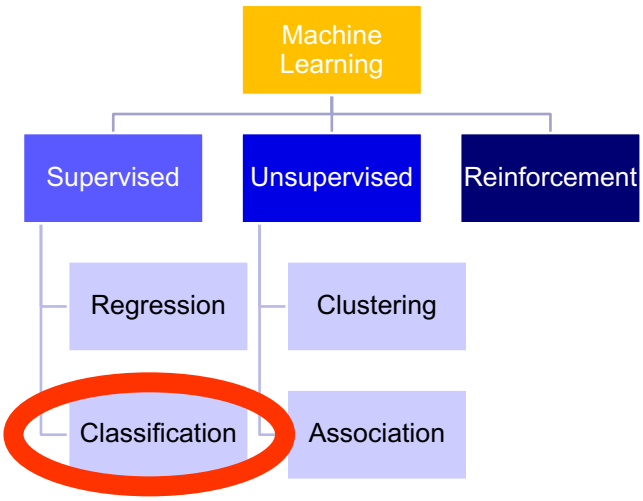
Paper 060010907: Machine Learning

Unit-3 Classification

1

Types of Machine Learning



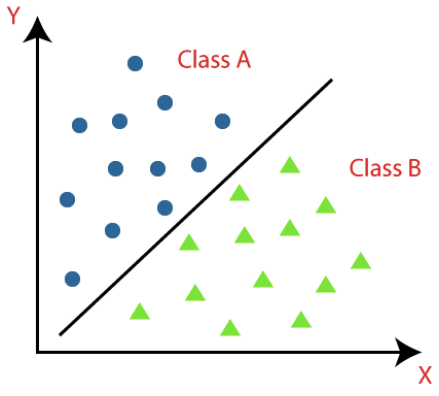
2

Supervised Learning

- **Supervised learning** is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
- It infers a function from labeled training data consisting of a set of training examples.

Types of Supervised Learning

- Classification



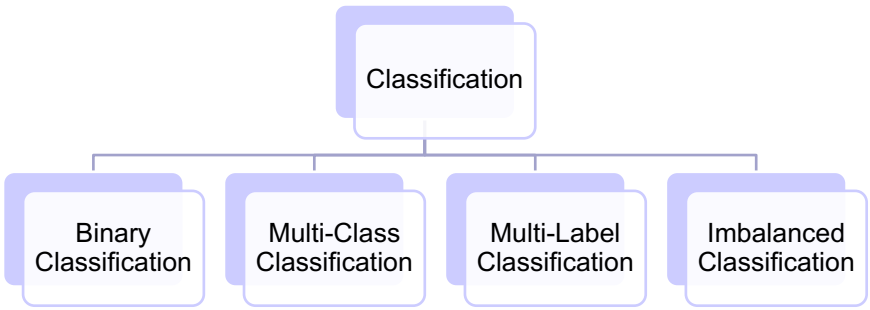
Types of Supervised Learning

- Classification
  - Output variable in classification is categorical or discrete
  - Classification algorithms attempt to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).
  - Helps in solving questions like “Is this email spam?” or ”Is this customer going to purchase the product or not?”

Types of Supervised Learning

- Classification algorithms
  - Logistic Regression
  - K-Nearest Neighbors (K-NN)
  - Support Vector Machine (SVM)
  - Naive Bayes

Types of Classification



7

Classification

- Binary Classification
  - Classifying the elements of a given dataset into two groups
  - i.e.
    - spam or not
    - attack or normal traffic
    - rain or not
  - Typically, binary classification involve one class as the normal and another class as the abnormal.

8

Classification

- Multi-Class Classification
  - Classifying the elements of a given dataset into more than two groups
  - i.e
    - Face detection
    - Mango type classification
    - Optical character recogniztion

9

Classification

- Multi-Class Classification
  - Algorithms that are designed for binary classification can be adapted for use for multi-class problems.
    - One-vs-Rest
    - One-vs-One

10

Classification

- Multi-Class Classification
  - One-vs-Rest
    - Split the multi-class dataset into multiple binary classification problems
    - Each classification sub-problem is one class versus rest all classes

Classification

- Multi-Class Classification
  - One-vs-Rest
    - i.e. For four classes “Triangle”, “Rectangle”, “Pentagon”, “Hexagon”, we require four models
      - Model 1 – “Triangle” vs [“Rectangle”, “Pentagon”, “Hexagon”]
      - Model 2 – “Rectangle” vs [“Triangle”, “Pentagon”, “Hexagon”]
      - Model 3 – “Pentagon” vs [“Triangle”, “Rectangle”, “Hexagon”]
      - Model 4 – “Hexagon” vs [“Triangle”, “Rectangle”, “Pentagon”]
    - Largest score is used to predict the class

Classification

- Multi-Class Classification
  - One-vs-One
    - Split the multi-class dataset into multiple binary classification problems
    - One dataset for each class versus every other classes

13

Classification

- Multi-Class Classification
  - One-vs-One
    - i.e. For four classes “Triangle”, “Rectangle”, “Pentagon”, “Hexagon”, we require six models
      - Model 1 – “Triangle” vs “Rectangle”
      - Model 2 – “Triangle” vs “Pentagon”
      - Model 3 – “Triangle” vs “Hexagon”
      - Model 4 – “Rectangle” vs “Pentagon”
      - Model 5 – “Rectangle” vs “Hexagon”
      - Model 6 – “Pentagon” vs “Hexagon”
    - Largest score is used to predict the class

14

Classification

- Multi-Label Classification
  - have two or more classes and
  - one or more classes may be predicted for each record
  - i.e. identify presence of “human”, “car”, “building” in the photo

15

Classification

- Imbalanced Classification
  - number of examples in each class is unequally distributed
  - Typically like binary classification, but majority of examples in the training dataset belong to the normal class and a minority of examples belong to the abnormal class
  - i.e. fraud detection, outlier detection
  - requires under-sampling the majority class or over-sampling the minority class

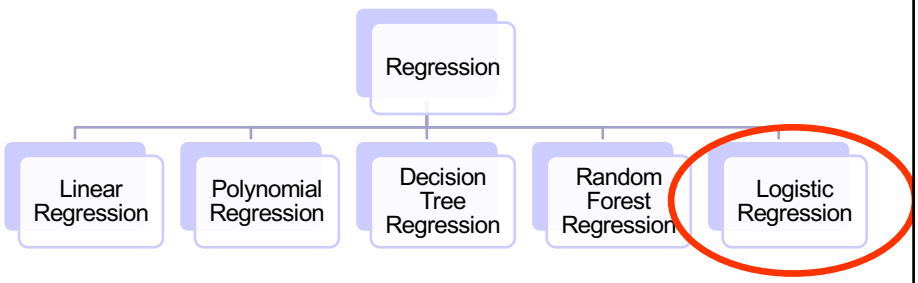
16

Classification

- Logistic Regression
- K-Nearest Neighbors (K-NN)
- Support Vector Machine (SVM)
- Naïve Bays

17

Types of Regression



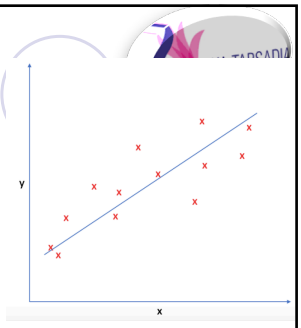
18

Logistic Regression

- Supervised machine learning
- Outcome will be categorical or discrete dependent variable
- Instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

19

Logistic Regression

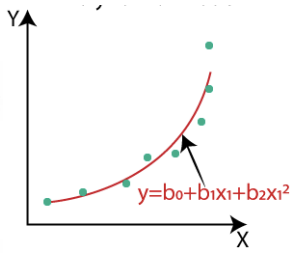


- Simple Linear Regression

$$y = b_0 + b_1 x_1$$

- Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + ..$$



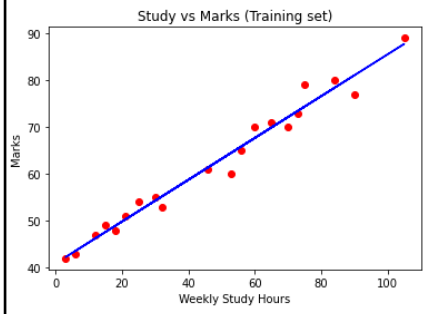
- Polynomial Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

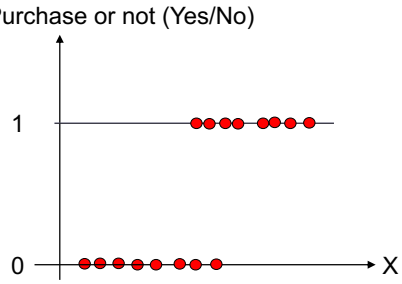
20

Logistic Regression

Linear Regression

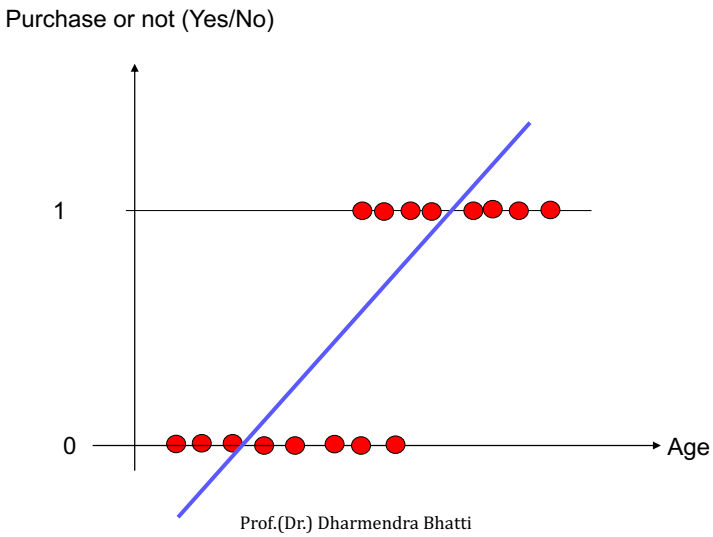


Logistics Regression



21

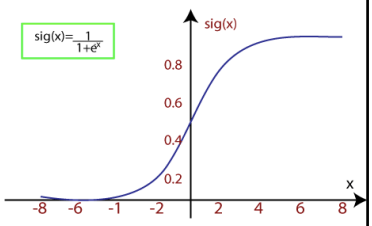
Logistic Regression

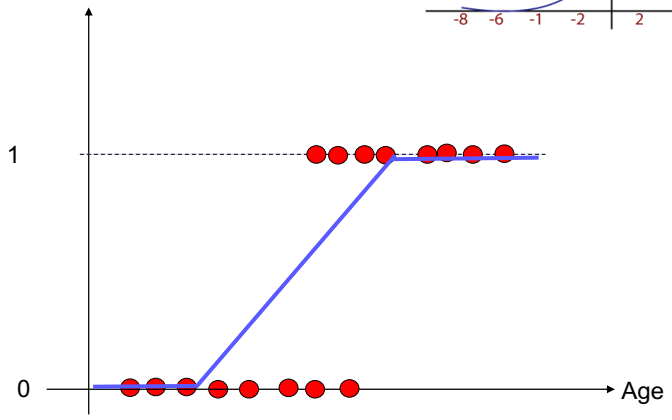


22

Logistic Regression

Purchase or not (Yes/No)

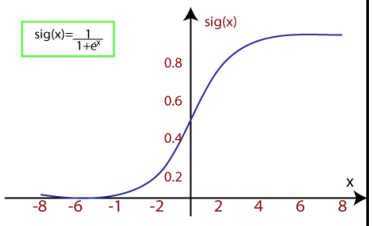


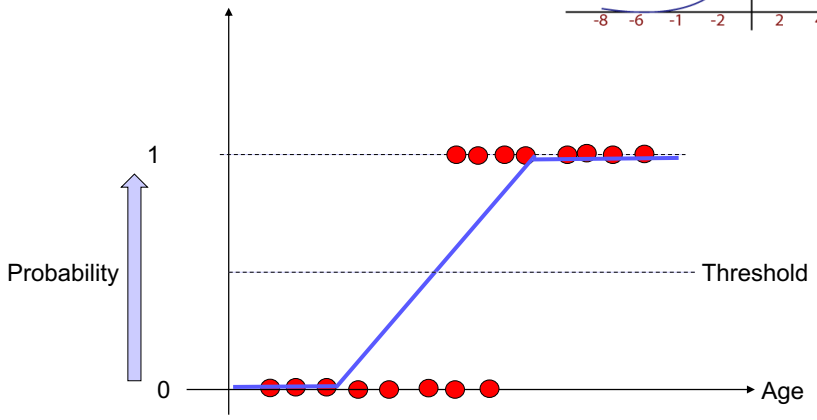


23

Logistic Regression

Purchase or not (Yes/No)





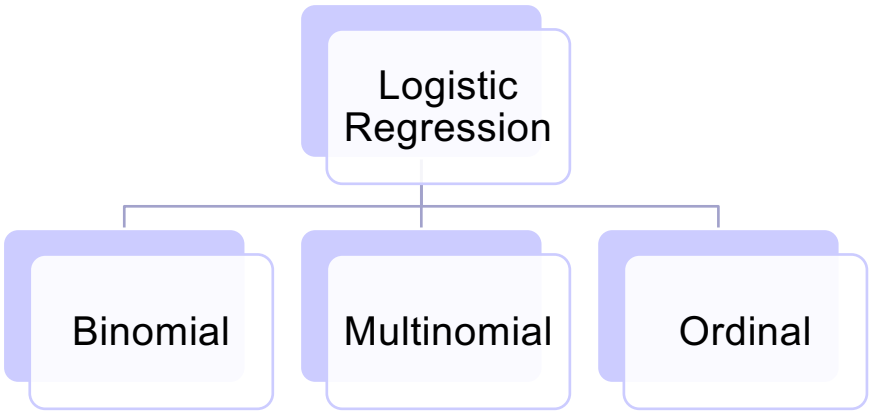
24

Logistic Regression

- Assumptions for Logistic Regression
  - The dependent variable must be categorical
  - No multicollinearity among independent variables

25

Logistic Regression



26

Logistic Regression

- Binomial
  - Dependent variable must have two classes only
  - i.e.
    - yes or no
    - pass or fail
    - spam or normal
    - purchase or no

27

Logistic Regression

- Multinomial
  - Dependent variable must have 3 or more **unordered** classes
  - i.e.
    - red, green, blue
    - Information Technology, Computer Science, Forensic Science

28

Logistic Regression

- Ordinal
  - Dependent variable must have 3 or more **ordered** classes
  - i.e.
    - poor, average, good, very good
    - low, medium, high

29

Logistic Regression – Python

Independent Variables

	A	B			E	
1	User ID	Gender	Age	EstimatedSalary	Purchased	← Dependent Variable
2	15624510	Male	19	19000	0	
3	15810944	Male	35	20000	0	
4	15668575	Female	26	43000	0	
5	15603246	Female	27	57000	0	
6	15804002	Male	19	76000	0	
7	15728773	Male	27	58000	0	
8	15598044	Female	27	84000	0	
9	15694829	Female	32	150000	1	
10	15600575	Male	25	33000	0	
11	15727311	Female	35	65000	0	
12	15570769	Female	26	80000	0	
13	15606274	Female	26	52000	0	
14	15746139	Male	20	86000	0	
15	15704987	Male	32	18000	0	
16	15628972	Male	18	82000	0	
17	15697686	Male	29	80000	0	
18	15733883	Male	47	25000	1	
19	15617482	Male	45	26000	1	
20	15704583	Male	46	28000	1	

30

Logistic Regression – Python

- Importing Libraries

```
 9  # Logistic Regression
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


31

Logistic Regression – Python

- Importing Dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Social_Network_Ads.csv')
18  X = dataset.iloc[:, [2, 3]].values
19  y = dataset.iloc[:, -1].values
```


32

Logistic Regression – Python

- Importing Dataset

dataset - DataFrame

Index	User ID	Gender	Age	stimatedSala	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0

Format Resize Background colo Column min/max Save and Close Close

33

33

Logistic Regression – Python

- Create two vectors X and y

X - NumPy object array

	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000

Format Resize Background color Save and Close Close

y - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

Format Resize Background color Save and Close Close

34

Logistic Regression – Python

- Splitting the dataset into the Training set and Test set

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.25, random_state = 0)
```

- Total 400 records: 300 for training, 100 for testing

Logistic Regression – Python

- Feature Scaling

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Logistic Regression – Python

Feature Scaling

X_train - NumPy object array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

X_train - NumPy object ar

	0	1
0	0.581649	-0.886707
1	-0.606738	1.46174
2	-0.01254…	-0.567782
3	-0.606738	1.89663
4	1.37391	-1.40858
5	1.47294	0.997847
6	0.0864882	-0.799728
7	-0.01254…	-0.248858
8	-0.210609	-0.567782

X_test - NumPy object array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

X_test - NumPy object arr

	0	1
0	-0.804802	0.504964
1	-0.01254…	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.04590…

37

Logistic Regression – Python

- Training the Logistic Regression model on the Training set

```
32  # Training the Logistic Regression model on the Training set
33  from sklearn.linear_model import LogisticRegression
34  classifier = LogisticRegression(random_state = 0)
35  classifier.fit(X_train, y_train)
```


38

Logistic Regression – Python

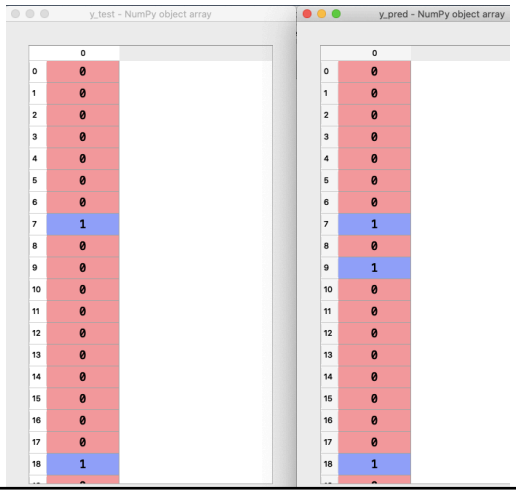
- Predicting the Test set results

```
37  # Predicting the Test set results
38  y_pred = classifier.predict(X_test)
```


39

Logistic Regression – Python

- Predicting the Test set results



40

Logistic Regression – Python

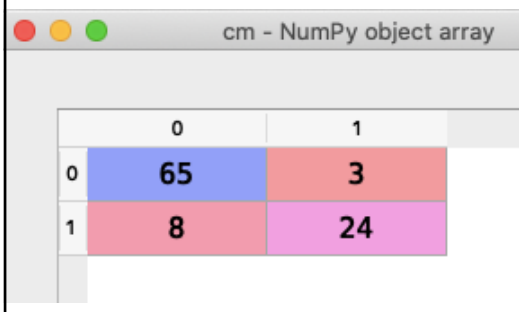
- Making the Confusion Matrix

```
40  # Making the Confusion Matrix
41  from sklearn.metrics import confusion_matrix
42  cm = confusion_matrix(y_test, y_pred)
43  print(cm)
```


41

Logistic Regression – Python

- Making the Confusion Matrix



		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

42

Logistic Regression – Python

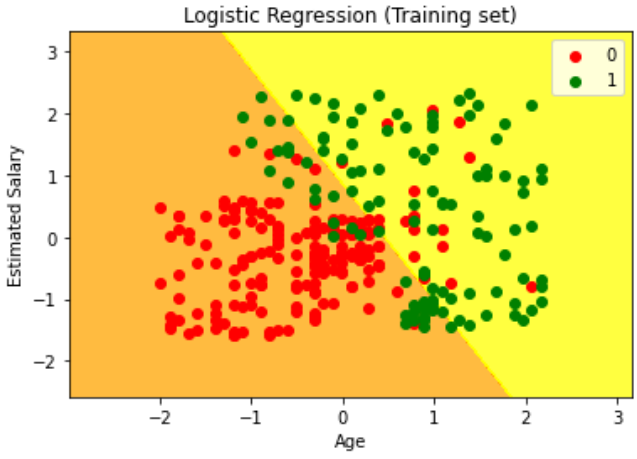
- Plotting training set results

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
            np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
            alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

43

Logistic Regression – Python

- Plotting training set results



44

44

Logistic Regression – Python

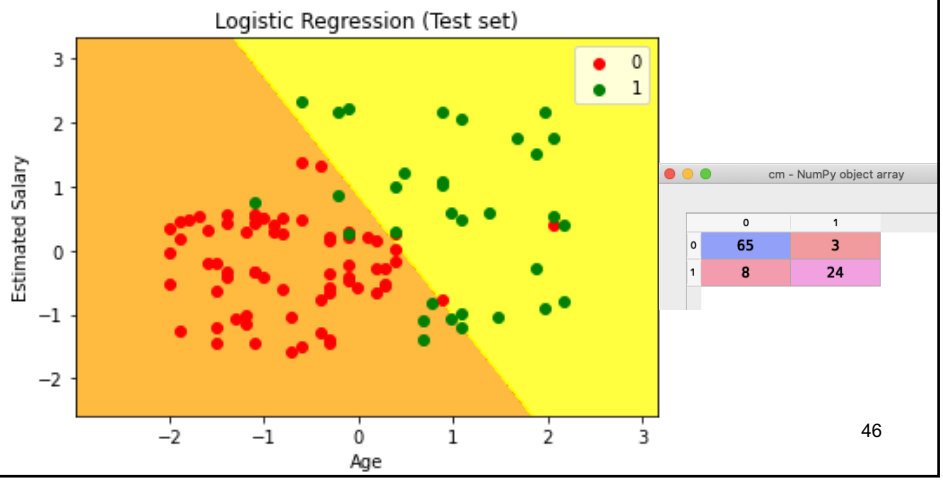
- Plotting testing set results

```
66  # Visualising the Test set results
67  from matplotlib.colors import ListedColormap
68  X_set, y_set = X_test, y_test
69  X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
70                                 stop = X_set[:, 0].max() + 1, step = 0.01),
71                       np.arange(start = X_set[:, 1].min() - 1,
72                                 stop = X_set[:, 1].max() + 1, step = 0.01))
73  plt.contourf(X1, X2, classifier.predict(
74              np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
75              alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
76  plt.xlim(X1.min(), X1.max())
77  plt.ylim(X2.min(), X2.max())
78  for i, j in enumerate(np.unique(y_set)):
79      plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
80                  c = ListedColormap(('red', 'green'))(i), label = j)
81  plt.title('Logistic Regression (Test set)')
82  plt.xlabel('Age')
83  plt.ylabel('Estimated Salary')
84  plt.legend()
85  plt.show()
```

45

Logistic Regression – Python

- Plotting testing set results



46

46

K-Nearest Neighbors (K-NN)

- Supervised machine learning algorithm
- Used for classification

47

K-Nearest Neighbors (K-NN)

- K-NN is a non-parametric algorithm (it does not make any assumption on underlying data).

48

K-Nearest Neighbors (K-NN)

- K-NN stores information of all training cases and classifies new cases based on similarity.
- For a new data point, K-NN searches through the entire training set for the K most similar training data and select the most common outcome.

49

K-Nearest Neighbors (K-NN)

- It is also called a **lazy learner** algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

50

K-Nearest Neighbors (K-NN)

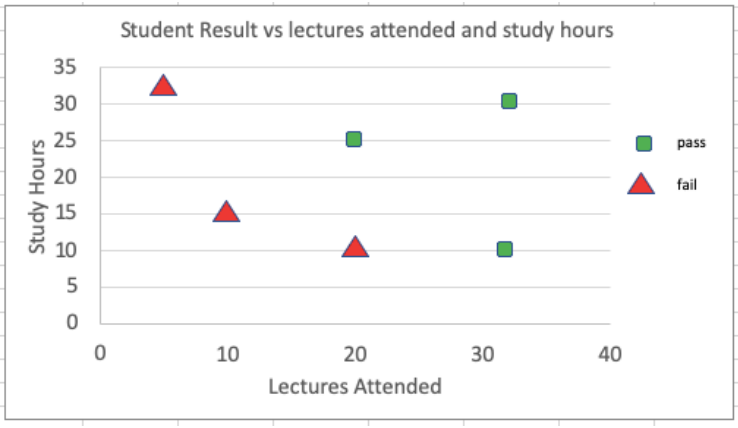
- Example

LecturesAttended	StudyHours	Result
32	30	pass
32	10	pass
10	15	fail
20	25	pass
5	32	fail
20	10	fail

51

K-Nearest Neighbors (K-NN)

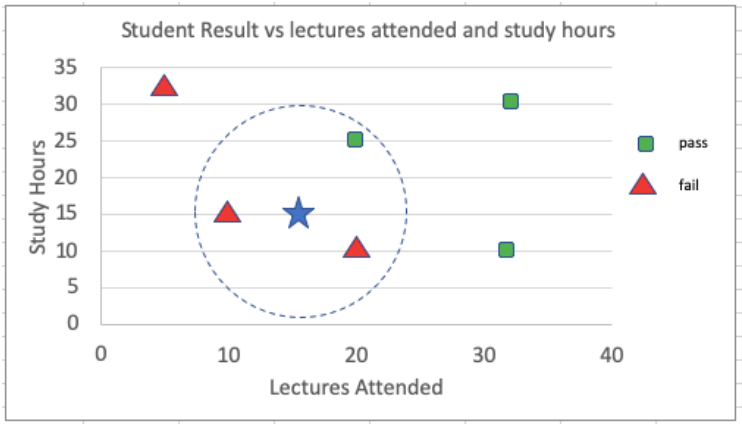
- Example



52

K-Nearest Neighbors (K-NN)

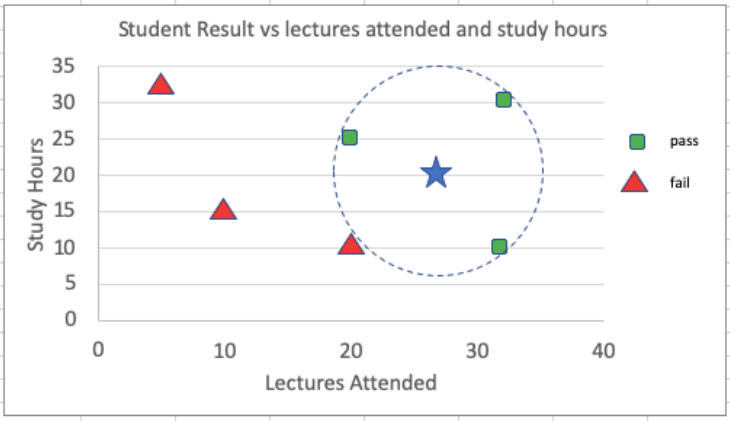
- Example



53

K-Nearest Neighbors (K-NN)

- Example



54

K-Nearest Neighbors (K-NN)

- Step-1:
  - Initialize parameter K = number of nearest neighbors
- Step-2:
  - Calculate the distance between the new/test datapoint and all training records

55

K-Nearest Neighbors (K-NN)

- Step-3:
  - Find the K training points with smallest distances
- Step-4:
  - Among these K neighbors, count the number of the data points in each category

56

K-Nearest Neighbors (K-NN)

- Step-5:
  - Assign the new data points to that category for which the number of the neighbor is maximum.

57

K-Nearest Neighbors (K-NN)

- Calculate result for LecturesAttended=15 and StudyHours=15

LecturesAttended	StudyHours	Result
32	30	pass
32	10	pass
10	15	fail
20	25	pass
5	32	fail
20	10	fail

58

K-Nearest Neighbors (K-NN)

- Step-1:
  - Initialize parameter K = number of nearest neighbors
- Suppose K = 3

59

K-Nearest Neighbors (K-NN)

- Step-2:
  - Calculate the squared distance between the new/test datapoint and all training records

D2 | =POWER(A2-15, 2)+POWER(B2-15,2)

	A	B	C	D
1	LecturesAttended	StudyHours	Result	Squared distance to New Point (15, 15)
2	32	30	pass	514
3	32	10	pass	314
4	10	15	fail	25
5	20	25	pass	125
6	5	32	fail	389
7	20	10	fail	50

60

K-Nearest Neighbors (K-NN)

- Step-3:
  - Find the K training points with smallest distances

LecturesAttended	StudyHours	Result	Squared distance to New Point (15, 15)
32	30	pass	514
32	10	pass	314
10	15	fail	25
20	25	pass	125
5	32	fail	389
20	10	fail	50

61

K-Nearest Neighbors (K-NN)

- Step-4:
  - Among these K neighbors, count the number of the data points in each category
  - Number of datapoints in “fail” category =2
  - Number of datapoints in “pass” category =1

62

K-Nearest Neighbors (K-NN)

- Step-5:
  - Assign the new data point to that category for which the number of the neighbor is maximum.
- No. of datapoints in “fail” category > No. of datapoints in “pass” category so prediction for new datapoint is **“fail”**

63

K-Nearest Neighbors (K-NN)

- How to calculate distance??

64

K-Nearest Neighbors (K-NN)

- How to calculate distance??
  - For continuous variables,

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

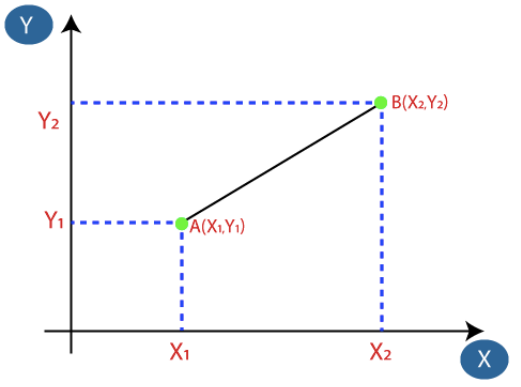
Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

65

65

K-Nearest Neighbors (K-NN)

- How to calculate distance??
  - For continuous variables,



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

66

66

K-Nearest Neighbors (K-NN)

- How to calculate distance??
  - For categorical variables,

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

67

67

K-Nearest Neighbors (K-NN)

- How to calculate distance??
  - When independent variables in training data are measured in different units, standardize variables before calculating distance.

$$Xs = \frac{X - mean}{s.d.}$$

$$Xs = \frac{X - mean}{max - min}$$

$$Xs = \frac{X - min}{\max - min}$$

Standardization

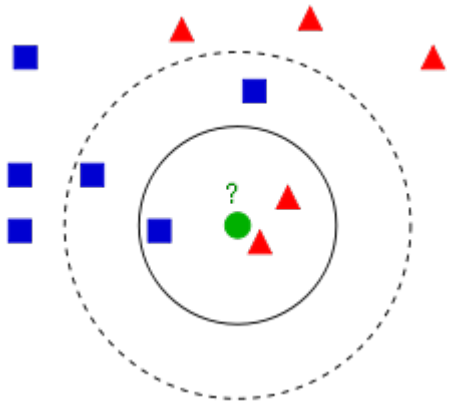
Prof.(Dr.) Dharmendra Bhatti

68

68

K-Nearest Neighbors (K-NN)

- How to select value of parameter K??



69

K-Nearest Neighbors (K-NN)

- How to select value of parameter K??
- Low k-value is sensitive to outliers and a higher K-value is more resilient to outliers as it considers more voters to decide prediction.

70

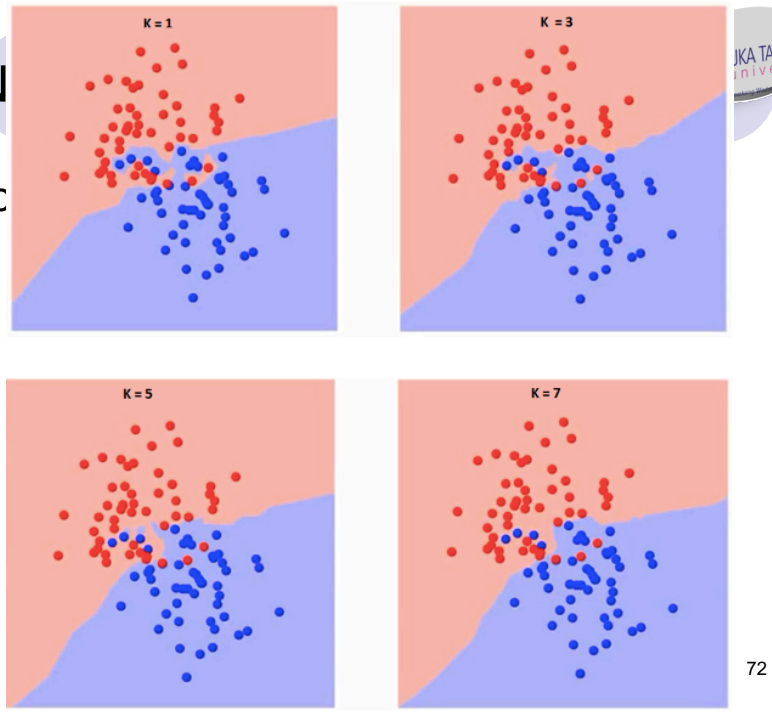
K-Nearest Neighbors (K-NN)

- How to select value of parameter K??
- Solution:
  - Use validation data for deciding value of parameter K.

71

K-N

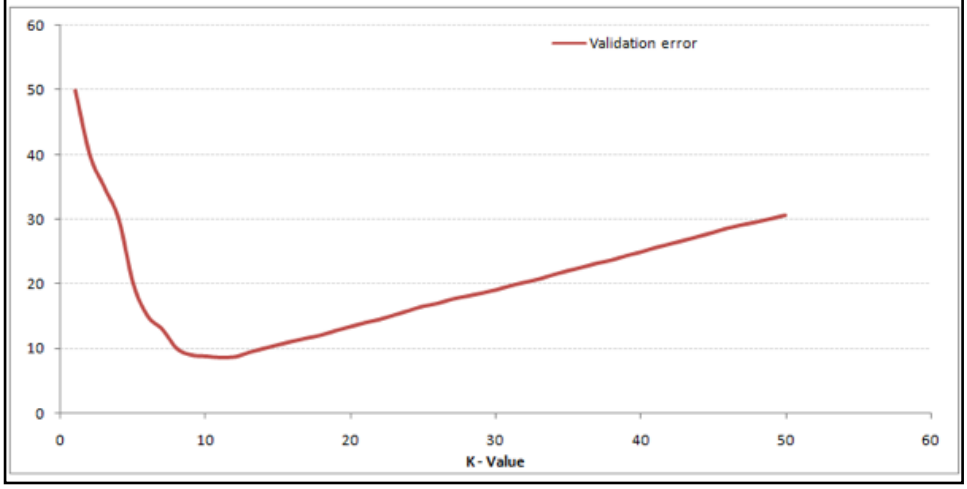
- Hc



72

K-Nearest Neighbors (K-NN)

- How to select value of parameter K??



73

K-Nearest Neighbors (K-NN)

- How to select value of parameter K??
  - Very small K results in high validation error
  - Very high K also results in high validation error
  - Select K where validation error is minimum

74

K-Nearest Neighbors – Python

Independent Variables

	A	B			E
1	User ID	Gender	Age	EstimatedSalary	Purchased
2	15624510	Male	19	19000	0
3	15810944	Male	35	20000	0
4	15668575	Female	26	43000	0
5	15603246	Female	27	57000	0
6	15804002	Male	19	76000	0
7	15728773	Male	27	58000	0
8	15598044	Female	27	84000	0
9	15694829	Female	32	150000	1
10	15600575	Male	25	33000	0
11	15727311	Female	35	65000	0
12	15570769	Female	26	80000	0
13	15606274	Female	26	52000	0
14	15746139	Male	20	86000	0
15	15704987	Male	32	18000	0
16	15628972	Male	18	82000	0
17	15697686	Male	29	80000	0
18	15733883	Male	47	25000	1
19	15617482	Male	45	26000	1
20	15704583	Male	46	28000	1

Dependent Variable

K-Nearest Neighbors – Python

- Importing the Libraries

```
 9  # K-Nearest Neighbors (K-NN)
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```

K-Nearest Neighbors – Python

- Importing the Dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Social_Network_Ads.csv')
18  X = dataset.iloc[:, [2, 3]].values
19  y = dataset.iloc[:, -1].values
```


77

K-Nearest Neighbors – Python

- Splitting the dataset into the Training set and Test set

```
21  # Splitting the dataset into the Training set and Test set
22  from sklearn.model_selection import train_test_split
23  X_train, X_test, y_train, y_test = train_test_split(
24      X, y, test_size = 0.25, random_state = 0)
```


78

K-Nearest Neighbors – Python

- Feature Scaling

```
26  # Feature Scaling
27  from sklearn.preprocessing import StandardScaler
28  sc = StandardScaler()
29  X_train = sc.fit_transform(X_train)
30  X_test = sc.transform(X_test)
```


79

K-Nearest Neighbors – Python

Feature Scaling

X_train - NumPy object array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

X_train - NumPy object ar

	0	1
0	0.581649	-0.886707
1	-0.606738	1.46174
2	-0.01254…	-0.567782
3	-0.606738	1.89663
4	1.37391	-1.40858
5	1.47294	0.997847
6	0.0864882	-0.799728
7	-0.01254…	-0.248858
8	-0.210609	-0.567782

X_test - NumPy object array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

X_test - NumPy object arr

	0	1
0	-0.804802	0.504964
1	-0.01254…	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.04590…

80

K-Nearest Neighbors – Python

- Training the K-NN model on the Training set

```
32  # Training the K-NN model on the Training set
33  from sklearn.neighbors import KNeighborsClassifier
34  classifier = KNeighborsClassifier(
35      n_neighbors = 5, metric = 'minkowski', p = 2)
36  classifier.fit(X_train, y_train)
```

KNeighborsClassifier

Definition : `KNeighborsClassifier(*, self, n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, radius=None)`

Classifier implementing the k-nearest neighbors vote.

81

81

K-Nearest Neighbors – Python

- Predicting the Test set results

```
38  # Predicting the Test set results
39  y_pred = classifier.predict(X_test)
```


82

K-Nearest Neighbors – Python

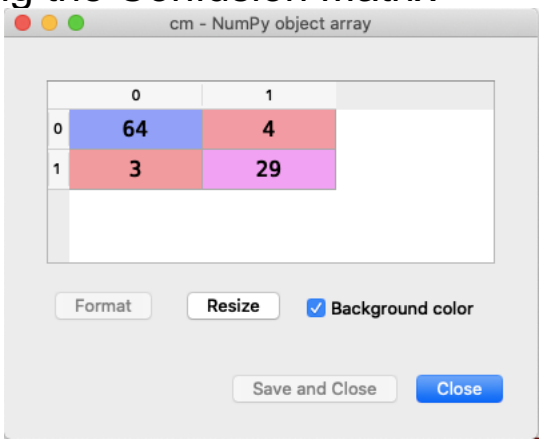
- Making the Confusion Matrix

```
41  # Making the Confusion Matrix
42  from sklearn.metrics import confusion_matrix
43  cm = confusion_matrix(y_test, y_pred)
44  print(cm)
```


83

K-Nearest Neighbors – Python

- Making the Confusion Matrix



84

K-Nearest Neighbors – Python

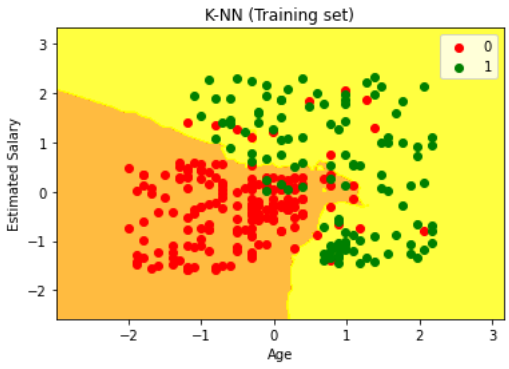
- Visualising the Training set results

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
    np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

85

K-Nearest Neighbors – Python

- Visualising the Training set results



86

K-Nearest Neighbors – Python

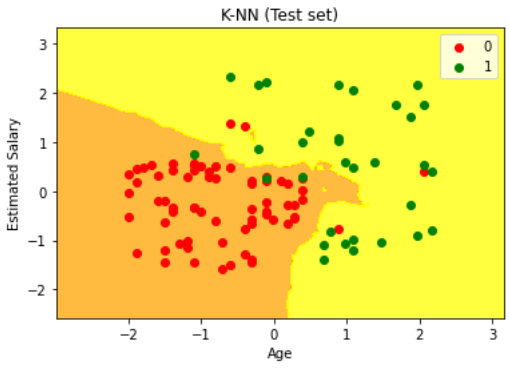
- Visualising the Testing set results

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
    np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
    alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

87

K-Nearest Neighbors – Python

- Visualising the Testing set results



88

Support Vector Machine (SVM)

- Supervised Learning algorithm
- Used for Classification

89

Support Vector Machine (SVM)

- It is a linear binary classifier.
- The SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963.

90

Support Vector Machine (SVM)

- How to classify??
  - A data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p - 1)-dimensional hyperplane.

91

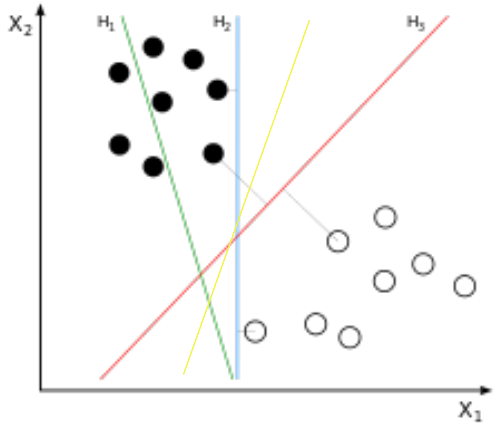
Support Vector Machine (SVM)

- Many hyperplanes are possible which might classify the data.
- Select the best hyperplane which represents the largest margin between the two classes.

92

Support Vector Machine (SVM)

- Select H1 or H2 or H3 hyperplane??



93

Support Vector Machine (SVM)

- Answer:
  - Choose the hyperplane so that the distance from it to the nearest data point on each side is maximized.

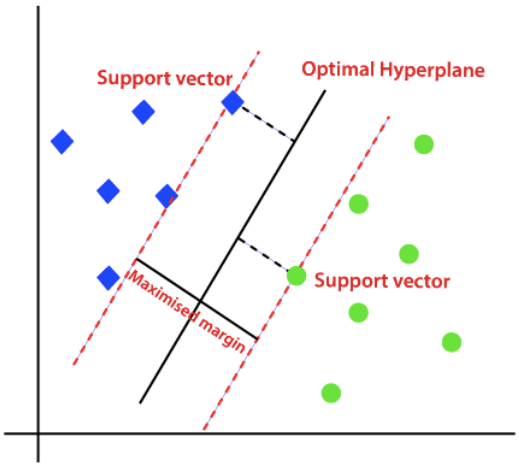
94

Support Vector Machine (SVM)

- SVM chooses the extreme points/vectors that help in creating the hyperplane.
- These extreme cases are called support vectors.

95

Support Vector Machine (SVM)



96

Support Vector Machine (SVM)

- The dimensions of the hyperplane depend on the features present in the dataset
  - If there are 2 features, then hyperplane will be a straight line
  - If there are 3 features, then hyperplane will be a 2-dimension plane
  - If there more than 3 features, then it will be a hyperplane

97

Support Vector Machine (SVM)

- Hyperplane
  - Decision plane which divides set of data points in to two different classes
- Margin
  - gap between two lines on the closet data points of different classes
- Support Vectors
  - Data points/vectors closest to the optimal hyperplane

98

Support Vector Machine – Python

Independent Variables

	A	B			E
1	User ID	Gender	Age	EstimatedSalary	Purchased
2	15624510	Male	19	19000	0
3	15810944	Male	35	20000	0
4	15668575	Female	26	43000	0
5	15603246	Female	27	57000	0
6	15804002	Male	19	76000	0
7	15728773	Male	27	58000	0
8	15598044	Female	27	84000	0
9	15694829	Female	32	150000	1
10	15600575	Male	25	33000	0
11	15727311	Female	35	65000	0
12	15570769	Female	26	80000	0
13	15606274	Female	26	52000	0
14	15746139	Male	20	86000	0
15	15704987	Male	32	18000	0
16	15628972	Male	18	82000	0
17	15697686	Male	29	80000	0
18	15733883	Male	47	25000	1
19	15617482	Male	45	26000	1
20	15704583	Male	46	28000	1

Dependent Variable

99

99

Support Vector Machine – Python

- Importing the libraries

```
# Support Vector Machine (SVM)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```


100

Support Vector Machine – Python

- Importing the dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Social_Network_Ads.csv')
18  X = dataset.iloc[:, [2, 3]].values
19  y = dataset.iloc[:, -1].values
```


101

Support Vector Machine – Python

- Importing the dataset

dataset - DataFrame

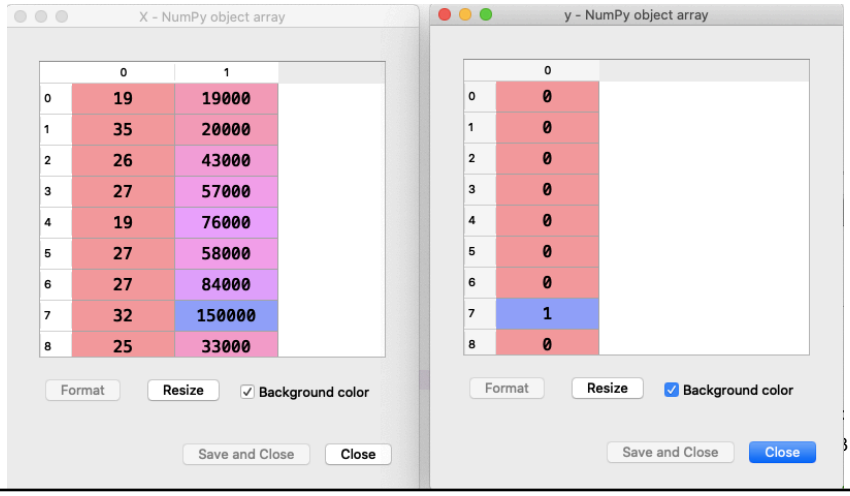
Index	User ID	Gender	Age	stimatedSala	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0

Format Resize Background colo Column min/max Save and Close Close

102

Support Vector Machine – Python

- Create two vectors X and y



103

Support Vector Machine – Python

- Splitting the dataset into the Training set and Test set

```
21  # Splitting the dataset into the Training set and Test set
22  from sklearn.model_selection import train_test_split
23  X_train, X_test, y_train, y_test = train_test_split(
24      X, y, test_size = 0.25, random_state = 0)
```


104

Support Vector Machine – Python

- Feature Scaling

```
26  # Feature Scaling
27  from sklearn.preprocessing import StandardScaler
28  sc = StandardScaler()
29  X_train = sc.fit_transform(X_train)
30  X_test = sc.transform(X_test)
```


105

Support Vector Machine – Python

Feature Scaling

X_train - NumPy object array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

X_train - NumPy object ar

	0	1
0	0.581649	-0.886707
1	-0.606738	1.46174
2	-0.01254…	-0.567782
3	-0.606738	1.89663
4	1.37391	-1.40858
5	1.47294	0.997847
6	0.0864882	-0.799728
7	-0.01254…	-0.248858
8	-0.210609	-0.567782

X_test - NumPy object array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

X_test - NumPy object arr

	0	1
0	-0.804802	0.504964
1	-0.01254…	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.04590…

106

Support Vector Machine – Python

- Training the SVM model on the Training set

```
32  # Training the SVM model on the Training set
33  from sklearn.svm import SVC
34  classifier = SVC(kernel = 'linear', random_state = 0)
35  classifier.fit(X_train, y_train)
```

SVC

Definition : `SVC(*, self, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=1e-3, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)`

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `sklearn.svm.LinearSVC` or `sklearn.linear_model.SGDClassifier` instead, possibly after a `sklearn.kernel_approximation.Nystroem` transformer.

The multiclass support is handled according to a one-vs-one scheme.

107

107

Support Vector Machine – Python

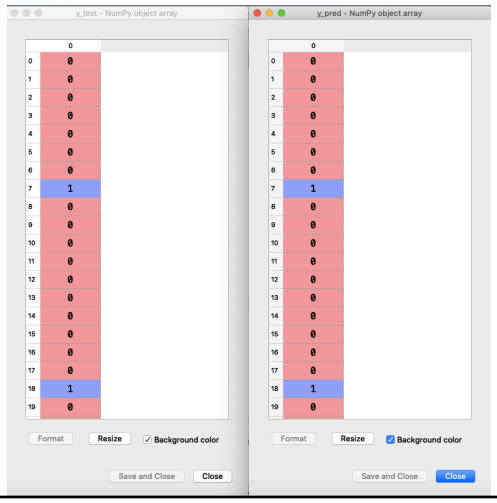
- Predicting the Test set results

```
37  # Predicting the Test set results
38  y_pred = classifier.predict(X_test)
```


108

Support Vector Machine – Python

- Predicting the Test set results



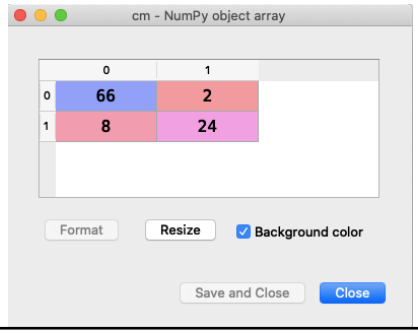
109

109

Support Vector Machine – Python

- Calculate the Confusion Metrix

```
40  # Making the Confusion Matrix
41  from sklearn.metrics import confusion_matrix
42  cm = confusion_matrix(y_test, y_pred)
43  print(cm)
```



110

110

Support Vector Machine – Python

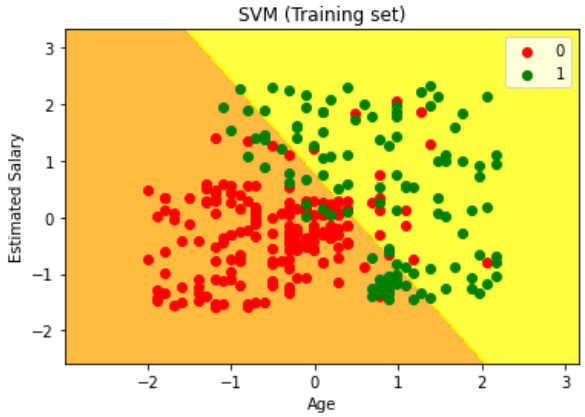
- Visualising the Training set results

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
    np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

111

Support Vector Machine – Python

- Visualising the Training set results



112

Support Vector Machine – Python

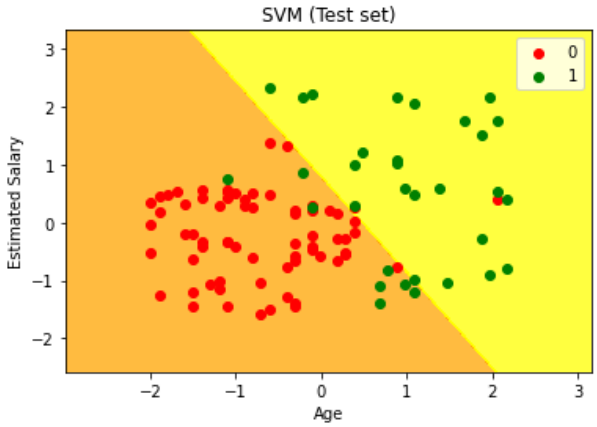
- Visualising the Test set results

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
    np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

113

Support Vector Machine – Python

- Visualising the Test set results



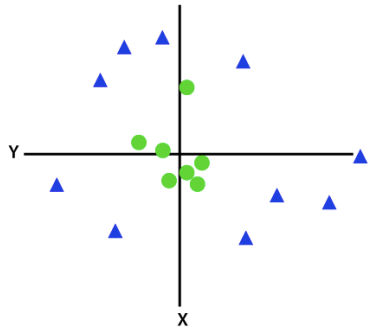
114

Support Vector Machine (SVM)

- In 1992, Bernhard Boser, Isabelle Guyon and Vladimir Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.
- The current incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.

Support Vector Machine (SVM)

- For non-linear data, we cannot have a single straight line.



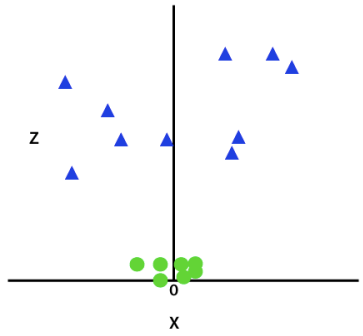
Support Vector Machine (SVM)

- Solution: add one more dimension.
  - For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z.
  - $z=x^2+y^2$

117

Support Vector Machine (SVM)

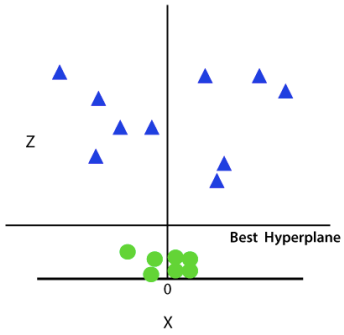
- Solution: add one more dimension.



118

Support Vector Machine (SVM)

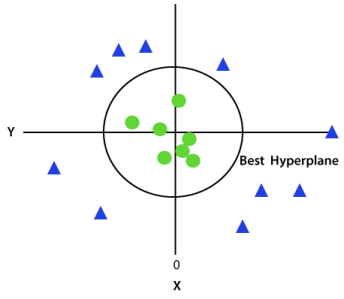
- Now, divide the data into two classes



119

Support Vector Machine (SVM)

- In 2-D space it will look like



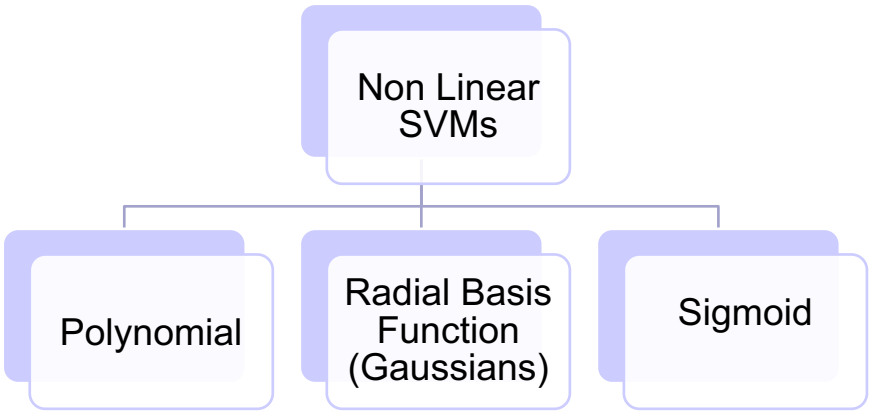
120

Support Vector Machine (SVM)

- A **Kernel function** transforms the training data so that a non-linear decision surface is transformed to a linear equation in a higher number of dimensions.

121

Support Vector Machine (SVM)



122

Support Vector Machine (SVM)

- Linear Kernel
  - It can be used as a dot product between any two observations.
  - The formula of linear kernel is −
    K(x,xi)=sum(x∗xi)

123

Support Vector Machine (SVM)

- Polynomial Kernel
  - It is more generalized form of linear kernel and distinguish curved or nonlinear input space.
  - Following is the formula for polynomial kernel −
    k(X,Xi)=1+sum(X∗Xi)^d
  - Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

124

Support Vector Machine (SVM)

- Radial Basis Function (RBF) Kernel
  - RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space.
  - Following formula explains it mathematically −

    K(x,xi)=exp(−gamma∗sum(x−xi^2))
  - Here, *gamma* ranges from 0 to 1.
  - We need to manually specify it in the learning algorithm.
  - A good default value of *gamma* is 0.1.

125

Support Vector Machine (SVM)

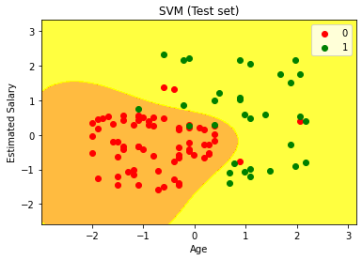
- Sigmoid Kernel
  - Hyperbolic tangent kernel
  - Also known as Neural Network activation function
  - Following formula explains it mathematically −

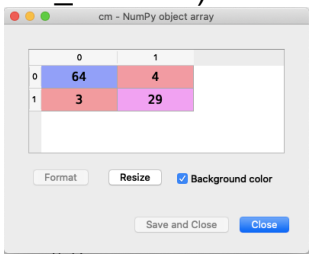
    K(x,xi)=tanh(αx*xi-δ))
  - A common value for alpha is 1/N, where N is the data dimension.
  - SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

126

Support Vector Machine (SVM)

- classifier = SVC(kernel = 'linear', random_state = 0)
- classifier = SVC(kernel = 'poly', degree=3, random_state = 0)
- classifier = SVC(kernel = 'rbf', random_state = 0)





Naïve Bays

- Supervised machine learning algorithm
- Used for classification

Naïve Bays

- Bayes' theorem
  - In probability theory and statistics, Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

129

Naïve Bays

- Bayes' theorem
  - For example, if the risk of developing health problems is known to increase with age, Bayes's theorem allows the risk to an individual of a known age to be assessed more accurately than simply assuming that the individual is typical of the population as a whole.

130

Naïve Bays

- Bayes' theorem – mathematical equation
  - $$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}$$
  - Where A and B are events and $P(B) \neq 0$
    - P(A | B) is conditional probability: the likelihood of event A occurring given that B is true.
    - P(B | A) is also a conditional probability: the likelihood of event B occurring given that A is true.
    - P(A) and P(B) are the probabilities of events A and B respectively; they are known as marginal probability.

Naïve Bays

- In statistics, Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features.

Naïve Bays

- When to use???
  - **Real-time Prediction:** As Naive Bayes is super fast, it can be used for making predictions in real time.
  - **Multi-class Prediction:** This algorithm can predict the posterior probability of multiple classes of the target variable.
  - **Scalable:** Used for Very large datasets

Naïve Bays – Example1

- https://machinelearningmastery.com/naive-bayes-tutorial-for-machine-learning/

Naïve Bays – Example1

- Weather, car condition, action example

	Weather	Car	Class
2	sunny	working	go-out
3	rainy	broken	go-out
4	sunny	working	go-out
5	sunny	working	go-out
6	sunny	working	go-out
7	rainy	broken	stay-home
8	rainy	broken	stay-home
9	sunny	working	stay-home
10	sunny	broken	stay-home
11	rainy	broken	stay-home

135

135

Naïve Bays – Example1

- Encode categorical data

	Weather	Car	Class
2	1	1	1
3	0	0	1
4	1	1	1
5	1	1	1
6	1	1	1
7	0	0	0
8	0	0	0
9	1	1	0
10	1	0	0
11	0	0	0

136

136

Naïve Bays – Example1

- **Calculate the Class Probabilities**
  - P(class=1) = count(class=1) / (count(class=0) + count(class=1))
  - P(class=1) = 5 / (5 + 5)
  - P(class=0) = count(class=0) / (count(class=0) + count(class=1))
  - P(class=0) = 5 / (5 + 5)
- 0.5 probability for any given data instance belonging to class 0 or class 1.

Naïve Bays – Example1

- **Calculate the Conditional Probabilities**
  - The conditional probabilities are the probability of each input value given each class value.

Naïve Bays – Example1

- **Calculate the Conditional Probabilities**
  - **Weather Input Variable**
    - P(weather=sunny|class=go-out) = count(weather=sunny and class=go-out) / count(class=go-out)
    - P(weather=rainy|class=go-out) = count(weather=rainy and class=go-out) / count(class=go-out)
    - P(weather=sunny|class=stay-home) = count(weather=sunny and class=stay-home) / count(class=stay-home)
    - P(weather=rainy|class=stay-home) = count(weather=rainy and class=stay-home) / count(class=stay-home)

Naïve Bays – Example1

- **Calculate the Conditional Probabilities**
  - **Weather Input Variable**
    - P(weather=sunny|class=go-out) = 0.8
    - P(weather=rainy|class=go-out) = 0.2
    - P(weather=sunny|class=stay-home) = 0.4
    - P(weather=rainy|class=stay-home) = 0.6

Naïve Bays – Example1

- **Calculate the Conditional Probabilities**
  - **Car Input Variable**
    - P(car=working|class=go-out) = count(car=working and class=go-out) / count(class=go-out)
    - P(car=broken|class=go-out) = count(car=brokenrainy and class=go-out) / count(class=go-out)
    - P(car=working|class=stay-home) = count(car=working and class=stay-home) / count(class=stay-home)
    - P(car=broken|class=stay-home) = count(car=brokenrainy and class=stay-home) / count(class=stay-home)

Naïve Bays – Example1

- **Calculate the Conditional Probabilities**
  - **Car Input Variable**
    - P(car=working|class=go-out) = 0.8
    - P(car=broken|class=go-out) = 0.2
    - P(car=working|class=stay-home) = 0.2
    - P(car=broken|class=stay-home) = 0.8

Naïve Bays – Example1

- **Make Predictions with Naive Bayes**
  - Test record: weather=sunny, car=working
  - Calculate for go-out class:
    - go-out = P(weather=sunny|class=go-out) * P(car=working|class=go-out) * P(class=go-out)
    - go-out = 0.8 * 0.8 * 0.5
    - go-out = 0.32

143

Naïve Bays – Example1

- **Make Predictions with Naive Bayes**
  - Test record: weather=sunny, car=working
  - Calculate for stay-home class:
    - stay-home = P(weather=sunny|class=stay-home) * P(car=working|class=stay-home) * P(class=stay-home)
    - stay-home = 0.4 * 0.2 * 0.5
    - stay-home = 0.04

144

Naïve Bays – Example1

- **Make Predictions with Naive Bayes**
  - Test record: weather=sunny, car=working
  - We can see that 0.32 is greater than 0.04, therefore we predict “go-out” for this instance

145

Naïve Bays – Example2

- https://www.simplilearn.com/tutorials/machine-learning-tutorial/naive-bayes-classifier
- Game play prediction using weather data

146

Naïve Bays – Example2

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

147

147

Naïve Bays – Example2

- First, we will create a **frequency** table using each attribute of the dataset.

Frequency Table		Play	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	3	2

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	Strong	6	2
	Weak	3	3

148

148

Naïve Bays – Example2

- For each frequency table, we will generate a **likelihood** table.

Likelihood Table		Play		
		Yes	No	
Outlook	Sunny	3/10	2/4	5/14
	Overcast	4/10	0/4	4/14
	Rainy	3/10	2/4	5/14
		10/14	4/14	

$P(x|c) = P(Sunny|Yes) = 3/10 = 0.3$

$P(x) = P(Sunny) = 5/14 = 0.36$

$P(c) = P(Yes) = 10/14 = 0.71$

Naïve Bays – Example2

- Likelihood of '**Yes**' given '**Sunny**' is:
  - **P(c|x)**
  - **= P(Yes|Sunny)**
  - **= P(Sunny|Yes)* P(Yes) / P(Sunny)**
  - **= (0.3 x 0.71) /0.36**
  - **= 0.591**

Naïve Bays – Example2

- Similarly Likelihood of '**No**' given '**Sunny**' is:
  - **P(c|x)**
  - **= P(No|Sunny)**
  - **= P(Sunny|No)* P(No) / P(Sunny)**
  - **= (0.4 x 0.36) /0.36**
  - **= 0.40**

151

Naïve Bays – Example2

- Now, in the same way, we need to create the Likelihood Table for other attributes as well.

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	Weak	6/9	2/5	8/14
	Strong	3/9	3/5	6/14
		9/14	5/14	

$P(Yes|High)$ = 0.33 x 0.6 / 0.5 = 0.42

$P(No|High)$ = 0.8 x 0.36 / 0.5 = 0.58

$P(Yes|Weak)$ = 0.67 x 0.64 / 0.57 = 0.75

$P(No|Weak)$ = 0.4 x 0.36 / 0.57 = 0.25

152

Naïve Bays – Example2

- Test for NEW data point
  - Suppose we have a **Day** with the following values :
  - **Outlook = Rain**
  - **Humidity = High**
  - **Wind = Weak**
  - **Play =?**
- we have to predict whether “we can play on that day or not”.

Naïve Bays – Example2

- **Likelihood of ‘Yes’ on that Day**
  - = P(Outlook = Rain|Yes)*P(Humidity= High|Yes)* P(Wind= Weak|Yes)*P(Yes)
  - **= 2/9 * 3/9 * 6/9 * 9/14**
  - **= 0.0199**
- **Likelihood of ‘No’ on that Day**
  - = P(Outlook = Rain|No)*P(Humidity= High|No)* P(Wind= Weak|No)*P(No)
  - **= 2/5 * 4/5 * 2/5 * 5/14**
  - **= 0.0166**

Naïve Bays – Example2

- Now we normalize the values, then
  - **P(Yes) = 0.0199 / (0.0199+ 0.0166) = 0.55**
  - **P(No) = 0.0166 / (0.0199+ 0.0166) = 0.45**

- Our model predicts that there is a **55%** chance there will be a Game on selected day.

155

Naïve Bays – Python

Independent Variables

	A	B			E
1	User ID	Gender	Age	EstimatedSalary	Purchased
2	15624510	Male	19	19000	0
3	15810944	Male	35	20000	0
4	15668575	Female	26	43000	0
5	15603246	Female	27	57000	0
6	15804002	Male	19	76000	0
7	15728773	Male	27	58000	0
8	15598044	Female	27	84000	0
9	15694829	Female	32	150000	1
10	15600575	Male	25	33000	0
11	15727311	Female	35	65000	0
12	15570769	Female	26	80000	0
13	15606274	Female	26	52000	0
14	15746139	Male	20	86000	0
15	15704987	Male	32	18000	0
16	15628972	Male	18	82000	0
17	15697686	Male	29	80000	0
18	15733883	Male	47	25000	1
19	15617482	Male	45	26000	1
20	15704583	Male	46	28000	1

Dependent Variable

156

Naïve Bays – Python

- Importing the libraries

```
 9  # Naive Bayes
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


157

Naïve Bays – Python

- Importing the dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Social_Network_Ads.csv')
18  X = dataset.iloc[:, [2, 3]].values
19  y = dataset.iloc[:, -1].values
```


158

Naïve Bays – Python

- Importing the dataset

dataset - DataFrame

Index	User ID	Gender	Age	stimatedSala	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0

Format Resize Background colo Column min/max Save and Close Close

159

159

Naïve Bays – Python

- Create two vectors X and y

X - NumPy object array

	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000

Format Resize Background color Save and Close Close

y - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

Format Resize Background color Save and Close Close

160

Naïve Bays – Python

- Splitting the dataset into the Training set and Test set

```
21  # Splitting the dataset into the Training set and Test set
22  from sklearn.model_selection import train_test_split
23  X_train, X_test, y_train, y_test = train_test_split(
24      X, y, test_size = 0.25, random_state = 0)
```


Naïve Bays – Python

- Feature Scaling

```
26  # Feature Scaling
27  from sklearn.preprocessing import StandardScaler
28  sc = StandardScaler()
29  X_train = sc.fit_transform(X_train)
30  X_test = sc.transform(X_test)
```

Naïve Bays – Python

Feature Scaling

X_train - NumPy object array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

X_train - NumPy object ar

	0	1
0	0.581649	-0.886707
1	-0.606738	1.46174
2	-0.01254…	-0.567782
3	-0.606738	1.89663
4	1.37391	-1.40858
5	1.47294	0.997847
6	0.0864882	-0.799728
7	-0.01254…	-0.248858
8	-0.210609	-0.567782

X_test - NumPy object array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

X_test - NumPy object arr

	0	1
0	-0.804802	0.504964
1	-0.01254…	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.04590…

163

Naïve Bays – Python

- Training the Naive Bayes model on the Training set

```
32  # Training the Naive Bayes model on the Training set
33  from sklearn.naive_bayes import GaussianNB
34  classifier = GaussianNB()
35  classifier.fit(X_train, y_train)
```

GaussianNB

Definition : `GaussianNB(*, self, priors=None, var_smoothing=1e-9)`

Gaussian Naive Bayes (GaussianNB)

164

Naïve Bays – Python

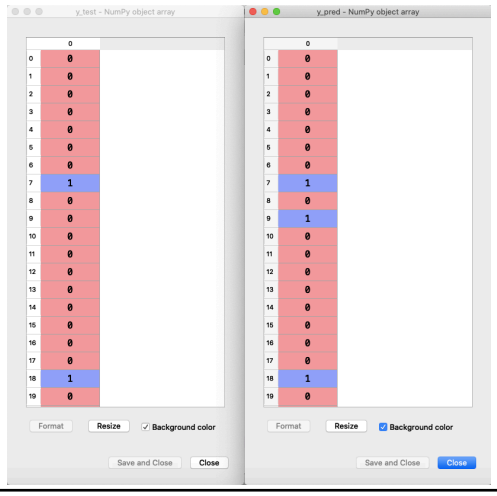
- Predicting the Test set results

```
37  # Predicting the Test set results
38  y_pred = classifier.predict(X_test)
```


165

Naïve Bays – Python

- Predicting the Test set results

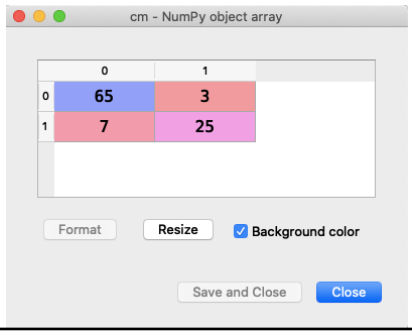


166

Naïve Bays – Python

- Calculate the Confusion Metrix

```
40  # Making the Confusion Matrix
41  from sklearn.metrics import confusion_matrix
42  cm = confusion_matrix(y_test, y_pred)
43  print(cm)
```



167

167

Naïve Bays – Python

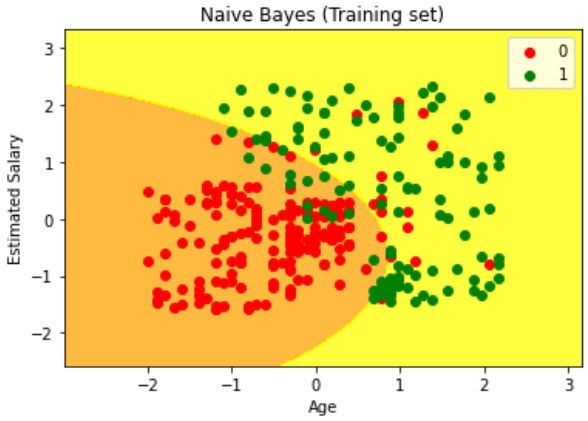
- Visualising the Training set results

```
45  # Visualising the Training set results
46  from matplotlib.colors import ListedColormap
47  X_set, y_set = X_train, y_train
48  X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
49                                 stop = X_set[:, 0].max() + 1, step = 0.01),
50                       np.arange(start = X_set[:, 1].min() - 1,
51                                 stop = X_set[:, 1].max() + 1, step = 0.01))
52  plt.contourf(X1, X2, classifier.predict(
53      np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
54               alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
55  plt.xlim(X1.min(), X1.max())
56  plt.ylim(X2.min(), X2.max())
57  for i, j in enumerate(np.unique(y_set)):
58      plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
59                  c = ListedColormap(('red', 'green'))(i), label = j)
60  plt.title('Naive Bayes (Training set)')
61  plt.xlabel('Age')
62  plt.ylabel('Estimated Salary')
63  plt.legend()
64  plt.show()
```

168

Naïve Bays – Python

- Visualising the Training set results



169

Naïve Bays – Python

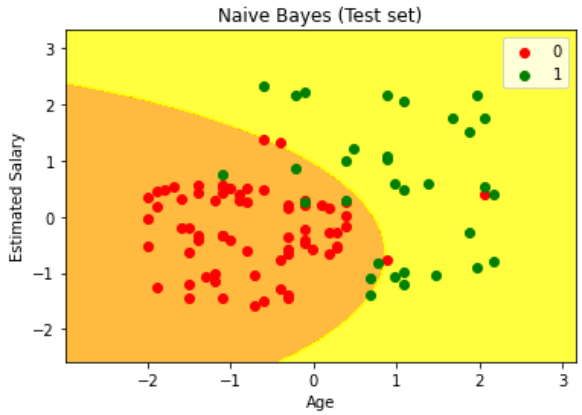
- Visualising the Test set results

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(
    np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('orange', 'yellow')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

170

Naïve Bays – Python

- Visualising the Test set results



Questions ???