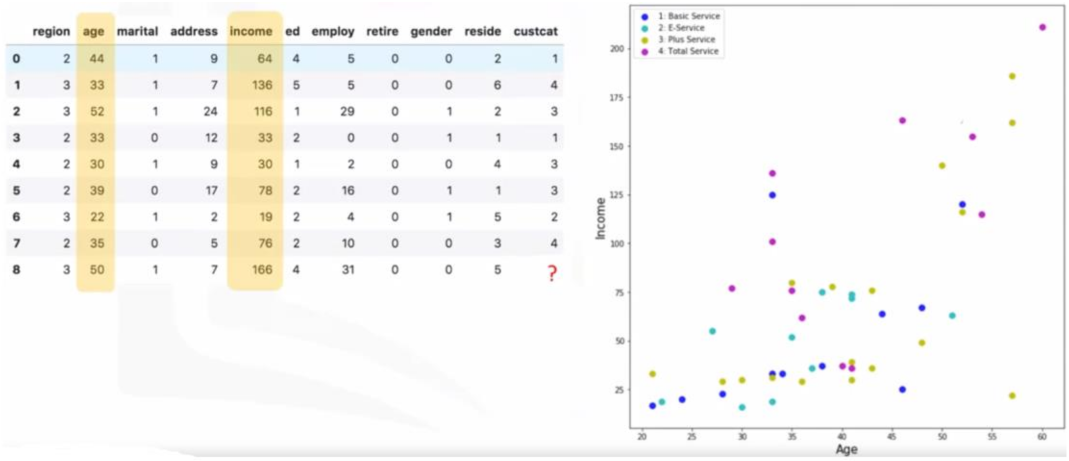
K-NEAREST NEIGHBORS (KNN)

K-Nearest Neighbors (KNN) can be defined as

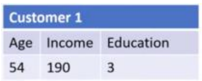
- A classification algorithm that takes a bunch of labeled points and uses them to learn how to label other points
- This algorithm classifies cases based on their similarity to other cases
- In KNN, data points that are near each other are said to be neighbors
- KNN is based on similar cases with same class labels are near each other thus the distance between two cases is a measure of their dissimilarity
- There are different ways to calculate the similarity or conversely, the distance or dissimilarity of 2 data points
    - Example – this can be done using Euclidean distance



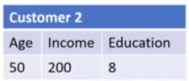
	region	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	44	1	9	64	4	5	0	0	2	1
1	3	33	1	7	136	5	5	0	0	6	4
2	3	52	1	24	116	1	29	0	1	2	3
3	2	33	0	12	33	2	0	0	1	1	1
4	2	30	1	9	30	1	2	0	0	4	3
5	2	39	0	17	78	2	16	0	1	1	3
6	3	22	1	2	19	2	4	0	1	5	2
7	2	35	0	5	76	2	10	0	0	3	4
8	3	50	1	7	166	4	31	0	0	5	?

KNN algorithm works as follows

- Pick a value for K
    - A low value of K causes a highly complex model as well which might result in overfitting of the model means the prediction process is not generalized enough to be used for out-of-sample cases
    - And if we choose a very high value of K then the model becomes overly generalized
    - The general solution to choose the right value of K is to reserve a part of our data for testing the accuracy of the model
        - Choose K equals 1 and then use the training part for modeling and calculate the accuracy of prediction using all samples in test set
        - Repeat this process increasing the K value and see which K is best for the model
- Calculate the distance of unknown case from all cases
    - We can calculate this value (either 1-dimensional or multi-dimensional space) using **Euclidean distance** equation
        - $Dis(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$
        - x_1 and x_2 are the observed data points
        - n is the number of dimensional that we use (total features)
        - i is the feature position we take for calculation



Customer 1		
Age	Income	Education
54	190	3



Customer 2		
Age	Income	Education
50	200	8

$$= \sqrt{(54-50)^2 + (190-200)^2 + (3-8)^2} = 11.87$$

- Select the K-observations in in the training data that are “nearest” to the measurements of unknown data point
- Predict the response of the unknown data point using the most popular response value from the KNN

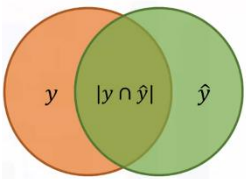
Evaluation Metrics in Classification

Evaluation metrics explain the performance of a model and now we'll describe about model evaluation metrics that are used for classification

- Basically, we compare the actual values in the test set with the values predicted by the model to calculate the accuracy of the model
- There are different model evaluation metrics here
  - **Jaccard index**
  - **F1-Score**
  - **Log Loss**

Jaccard index also known as *the Jaccard similarity coefficient*

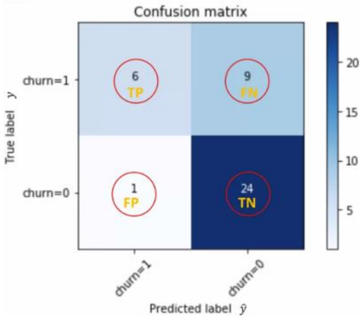
- Here is the math equation for Jaccard index $J(y,\hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} = \frac{|y \cap \hat{y}|}{|y|+|\hat{y}|-|y \cap \hat{y}|}$
- Let's say $\boldsymbol{y}$ shows the true labels of the target dataset and $\boldsymbol{\hat{y}}$ shows the predicted values by the classifier
- Then we can define Jaccard using its equation
  - For example
    - y: $[0,0,0,0,0,1,1,1,1,1]$
    - $\hat{y}$: $[1,1,0,0,0,1,1,1,1,1]$
    - $J(y,\hat{y}) = \frac{8}{10+10-8} = \mathbf{0.66}$



- **The higher of Jaccard value we get then the better our model for prediction**
- The range of Jaccard value is between 0 and 1

F1-Score value can be obtained from confusion matrix

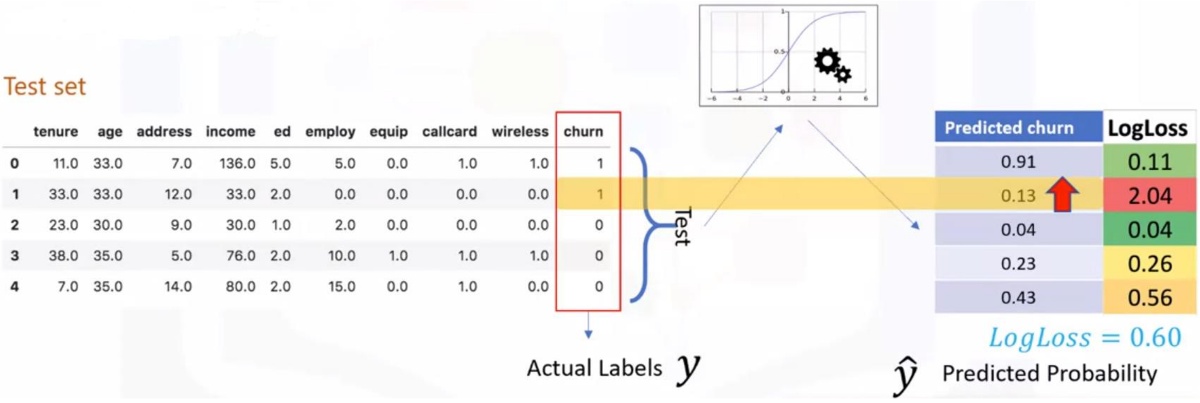
- On the right is the example and explanation of confusion matrix that we can use to obtain F1-Score



- Let's assume that the test set has only 40 rows and this matrix show the corrected and wrong predictions in comparison with the actual labels
- Each confusion matrix row shows the actual/True labels in the test set and the columns show the predicted labels by classifier
- In the 1st row the churn value of 15 of them is 1 and out of these 15 the classifier correctly predicted as 1 is 6 values and 9 of them as 0 which is not very good
- In the 2nd row, the classifier correctly predicted 24 of them as 0 and 1 of them wrongly predicted as 1 so it has done a good job in predicting the observe values with a churn value of 0
- In the specific case of a binary classifier, such as this example, we can interpret these numbers as the count of **True Positives (TP)**, **False Negatives (FN)**, **True Negatives (TN)** and **False Positives (FP)**
- Based on the count of each section, we can calculate the **Precision** and **Recall** of each label
  - Precision is defined by $Precision = TP/(TP + FP)$
    - Precision is a measure of the accuracy, provided that a class label has been predicted
  - Recall is defined by $Recall = TP/(TP + FN)$
    - Recall is the true positive rate value
- Now we can calculate the F1-Score for each label based on the Precision and Recall value of that label
  - $F1 - score = 2 * (Precision * Recall)/(Precision + Recall)$
  - The F1-Score is the harmonic average of the Precision and Recall where an F1-Score reaches its **best value at 1** and its **worst at 0**
  - We can tell the average accuracy for the classifier is the average of F1-Score for each label

Logarithmic Loss (Log Loss) can be described as

- Log Loss measures the performance of a classifier where the predicted output is a probability value between 0 and 1
- This equation $(y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y}))$ is to find Log Loss for each observation row
- And this equation $LogLoss = -\frac{1}{n}\sum(y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y}))$ is to find the average Log Loss value
- Sometimes the output of a classifier is the probability of a class label instead of the label
- Here is the example



Test set

	tenure	age	address	income	ed	employ	equip	callcard	wireless	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	1
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	1
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	0
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	0
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	0

Predicted churn	LogLoss
0.91	0.11
0.13	2.04
0.04	0.04
0.23	0.26
0.43	0.56

$LogLoss = 0.60$

  - In logistic regression, the output can be the probability of the target value (i.e. yes equals to 1) which is a value between 0 and 1
  - Predicting a probability of 0.13 when the actual label is 1 would be bad and would result in a high Log loss
  - We can calculate the Log Loss for each row using its equation which measures how far each prediction is from the actual label
  - Then we calculate the average Log Loss across all rows of the test set
- It is obvious that more ideal classifiers have progressively smaller values of log loss so **the classifier with lower Log Loss has better accuracy**