BUSAN302 Week 07-12

Week 07.1

Logistic Regression

Categorical variables, such as gender (male/female), color (pink, blue, purple, orange), contain different classes. While, continuous variables can take on ANY possible values, such as age, height, weight, salary, etc. When the dependent variable is dichotomous (0 or 1), (success or failure), we can use a logistic regression.

Aim: Whether an employee would get a promotion or not based on their performance
A linear graph will not be suitable in this case. As such, we clip the line at zero and one and convert into a sigmoid curve (S-curve). Based on the threshold values, the organisation can decide whether an employee will get a salary increase or not.

For logistic regression, use probabilities. For the straight-line equation, y can take the values of 0 to 1, as opposed from -Inf to Inf for a linear regression. Hence, to compare the two, we take the log of the odds ratio (log-odds), hence if 0 = p / 1 - p, then the log-odds will be $log(\frac{p(x)}{1-p(x)})$.

Types of Logistic Regression

- Binary Logistic Regression: The target only has two possible outcomes, such as Spam or Not Spam, Cancer or no cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of wine.
- Ordinal Logistic Regression: The order of categories is important, for example, if a variable has three or more categories such as restaurant or product ratings from 1 to 5.

Applications of Logistic Regression

- Using the logistic regression algorithm, banks can predict whether a customer would default on loans or not
- Predict weather conditions for a particular location (sunny, windy, rainy, humid,etc.)
- Ecommerce companies can identify buyers if they are likely to purchase a certain product
- Companies can predict whether they will gain or lose money in the next quarter, year, or month based on their current performance
- To classify objects based on their features and attributes

Linear Regression vs Logistic Regression

Linear regression: Solving prediction problems, estimates the dependent variable when there is a change in the independent variable, and is a straight line.
Logistic regression: Predicting the class of a variable, it helps calculate the possibility of a particular event taking place, and is an S-curve.

Logistic Regression in Python

```
# Reading dataset in
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
df = pd.read_csv('/content/diabetes.csv', header=None, names=col_names) # Add column names
```

```
# Selecting and subsetting variables from the dataset
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
x = df[feature_cols]                    # Extract the independent variables
y = df['label']                         # Extract the dependent variable

# Splitting the dataset into training and testing sets
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=16)
```

The dataset is broken into two parts in a ratio of 75:25. It means 75% data will be used for model training and 25% for model testing. random_state=16 is used as a set.seed() function. If you don't set random_state, the split could change each time you run it.

```
# Running the model and making a prediction
from sklearn.linear_model import LogisticRegression          # Import Logistic Regression
logreg = LogisticRegression(random_state=1, max_iter=1000)   # instantiate the model
logreg.fit(X_train, y_train)                                 # fit the model with data
y_pred = logreg.predict(X_test)
```

max_iter=1000 sets the maximum number of iterations the algorithm will try to find the best model. Logistic regression is solved by an optimization algorithm, and if it doesn't find the best solution quickly, it keeps iterating. If you don't set max_iter, sometimes it will stop too soon (default is 100), and you'll get a warning like "solver failed to converge."

```
# Model evaluation: Performance metrics [Confusion Matrix]
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
```

A confusion matrix is a table that is used to evaluate the performance of a classification model. Confusion matrix is the number of correct and incorrect predictions summed up class- wise

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Class Imbalance: In the dataset, out of 768, if 700 rows have the 'label' as 0, and 68 rows have the 'label' as 1. Hence, we have more data for the '0' class and much less for the second class '1'. Ideally, we should have two rows that are relatively close to each other for a more acceptable analysis, data given to train to the model is biased.

This is why we cannot use Accuracy on its own, as a performance metrics, hence why we need a confusion matrix.

```
# Model evaluation: Performance metrics [Accuracy]
```

accuracy = correct_predictions / total_predictions

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Model evaluation: Performance metrics [Precision score]

Out of the total predicted positives, how many are actually positive. Precision= TP/(TP+FP).

- Total predicted positives →TP+FP
- Actual positives → TP

```
from sklearn.metrics import precision_score
precision_score(y_test,y_pred)
```

Model evaluation: Performance metrics [Recall score]

Out of the total actual positives, how many are predicted correctly as positive. Recall = TP/(TP+FN)

- Total actual positives → TP+FN
- Predicted correctly as positive →TP

```
from sklearn.metrics import recall_score
recall_score(y_test,y_pred)
```

Week 07.2

F1 Score

F1-Score is the harmonic mean of the Recall and Precision. It is useful when you need to take both Precision and Recall into account.

Mean (x1+x2+x3+..+xn)/n

The harmonic mean will be high if all the values are high and will be low if all the values are low.

False Positives vs False Negatives

Cancer Diagnosis

	Actual	**Predicted**
False Positive	Cancer not detected	Cancer detected
False Negative	Cancer detected	Cancer not detected

False negatives are more severe than false positives because it opens a dangerous scenario where a patient is not treated for an underlying disease. A false positive might include unnecessarily medicating someone and/or causing stress and anxiety to the patient, although the repercussions of this is less severe as that for a false negative.

Spam Emails

For spam emails, false positives are more dangerous because, declaring valid emails as spam would be sent to the spam inbox and likely not reach the reader, and worse, will be deleted.

Improving the accuracy of the model

Change the rando_state value and run the model a few times and use the one with which you get the highest accuracy.

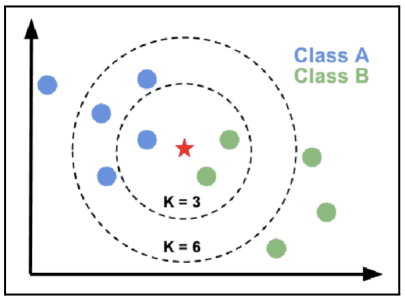
Precision, Recall and F1 Scores in Python

```
from sklean.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix
from sklearn import metric
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

accuracy_score(y_test, y_pred)
precision_score(y_test, y_pred)
recall_score(y_test, y_pred)
f1_score(y_test, y_pred)
```

Week 08.1

KNN - K-nearest neighbour

ELI5: Imagine you just moved to a new school and you're trying to guess what games a kid likes, but you don't know them yet. So, you look at the 3 kids (or any number k) who sit closest to them at lunch, their nearest neighbours. If most of those nearby kids like soccer, then you guess that the new kid probably likes soccer too. That's K-Nearest Neighbours (KNN), it makes a guess based on what the closest "friends" are like.



KNN is a supervised machine learning algorithm as it involves a labeled data set.

If we consider K=6, for the star, we would consider all the points inside both rings, and the star would be classified as Class A. Although, if we consider K=3, then the star would be classified in class B.

Euclidean distance

The difference between the data point and the consideration of other data points, squared, provides the euclidean distance. Other methods include Manhattan, Minkowski, etc

Example: Age vs Loan

1. Our dataset has Customer, Age, Loan and Default (Yes/No), for one datapoint, Andrew, we do not have a default value. Andrews age is 48 and his loan amount is 142000.
2. The euclidean distance between Andrew and John is 102,000.
3. We then take the Euclidean distance for all the data points, and add a new column called "Euclidean distance".
4. We then sort by ascending order, and find the five smallest distances.
5. For K=5, we see that there are two default=N, and three default=Y
6. Therefore, we predict Andrew's default = Yes

KNN Limitations

KNN Disadvantages

- KNN is a lazy learner, it requires all data points for each classification, so it's mostly better for smaller datasets. KNN performs very slowly. (Computational cost)
- It fails when variables have different scales. Feature scaling (standardisation and normalisation) is required before applying the KNN algorithm to any dataset. Otherwise KNN may provide wrong predictions.
  Example: For a computer, the Age and Salary are just pure numbers, hence it won't understand the context that someone who is 18 might have a lower salary (40,000) then someone older, such as 25 with a higher salary (250,000). Hence, it does NOT understand the context.

If you have these types in datasets, you should use scaling or transformation that brings the features into the same scale. For example, making the range 0 to 1 (log transformations for example).

KNN Advantages

- Simple to implement, only two parameters are required to implement KNN
- Suitable for small datasets

How to choose the K-Factor

- There is no best way to determine the k value. Most times it involves finetuning the K factor to compare whether or not the model performs better, until we find an appropriate one.
- There is a loose rule of thumb that K=5, although it does not really matter too much
- Another loose rule of thumb, take the sqrt(n), where n=number of datapoints, usually an odd value of K is selected to avoid confusion between two classes of data.

KNN in Python – IRIS dataset

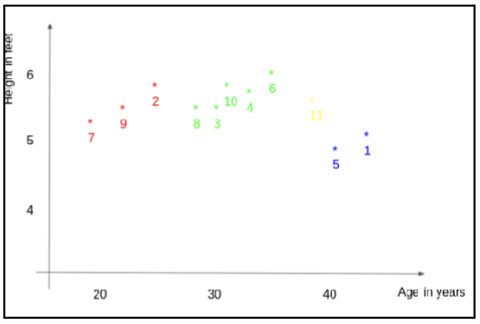
```
# Import required packages
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics

# Create feature and target variables
df = pd.read_csv('/content/iris.csv')
x=df.iloc[:,:4]
y = df.iloc[:, 4].values

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,random_state=42)
```

```
# Generate a k-NN model using neighbors value
knn = KNeighborsClassifier(n_neighbors=7)

# Train or fit the data into the model
knn.fit(X_train, y_train)

# Predict on dataset which model has not seen before
y_pred=knn.predict(X_test)

# Check accuracy and Matrix
accuracy = accuracy_score(y_test,y_pred)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
```

KNN in Regression

Example dataset of height vs age

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?



Week 08.2

Unsupervised Machine Learning

Machine learning using unlabeled data (no correct answers, and no output to guide the analysis, there is nothing right or nothing wrong). There are only inputs. The algorithm tries to group data based on similarities itself. For example, running ML on understanding customers of a store. The algorithm goes through the database and tries to form patterns on age, money spent, etc.

K-Means Clustering

K-means is an UNSUPERVISED clustering algorithm designed to partition unlabelled data into a certain number (K) distinct groups. It finds data points that share important characteristics and classifies them into clusters.

Intra-class similarity: Clusters data points within the same cluster are as similar as possible
Inter-class similarity: Data points from different clusters are as dissimilar as possible.

Example: K-Means ML for grouping shapes

Number of sides
3
4
5
6
4
8
6
4
3
2
3
7

1. Randomly pick three clusters (K=3) and assign each shape to the nearest centroid

Cluster 1: 3	3, 3, 2, 3
Cluster 2: 5	4, 5, 6, 4, 6, 4
Cluster 3: 8	8, 7

Centroids: 3, 5, 8

2. Recalculate centroid: Take the average of each cluster's numbers

Cluster 1: 3	11/4 = 2.75
Cluster 2: 5	39/6 = 4.83
Cluster 3: 8	15/2 = 7.5

New centroids: 2.75, 4.83, 7.5

3. Repeat assigning shapes to the nearest centroid. Eventually, this process settles into stable clusters.

Cluster 1 is probably triangles
Cluster 2 is probably squares, pentagons, or hexagons
Cluster 3 is probably octagons

K-Means disadvantages

- **Sensitivity to initial centroids**: K means is sensitive to the initial selection of centroids and can converge to a suboptimal solution
- **Requires specifying the number of clusters**: The number of clusters K needs to be specified before running the algorithm, which can be challenging in some applications
- **Sensitive to outliers**: K-means is sensitive to outliers, which can have a significant impact on the resulting clusters

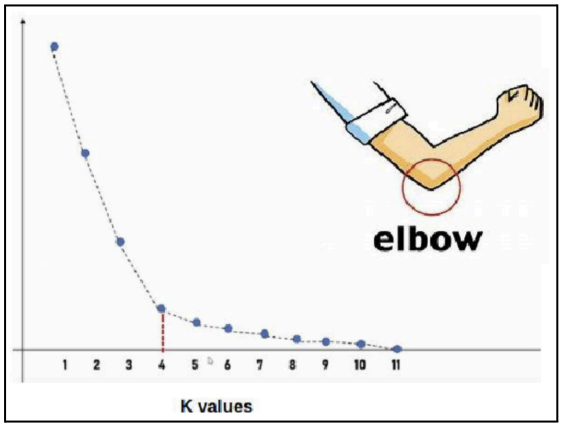
In Python

```
from sklearn.cluster import KMeans
kmeans_model = KMeans(n_clusters=3)
kmeans_predict = kmeans_model.fit_predict(x)
```

Week 09.1

Elbow Method

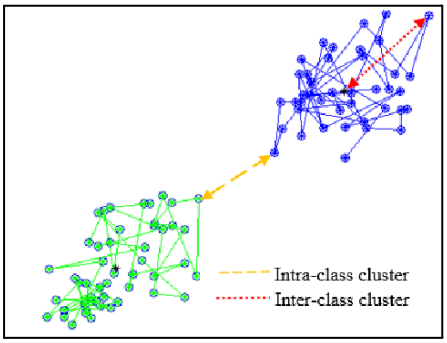
To find the best value of K, select the optimal number of clusters (k). WCSS is the total within-cluster sum of squares, it measures the compactness of the clustering and we want it to be as small as possible.



The elbow method finds the best value of K. If we start with 1, run a k-means algorithm, derive a WCSS value, continue with k=2, k=3, etc until we get a small WCSS value.

In the diagram above, the red line shows the optimal K value because after that point, the graph no longer grows exponentially, hence small changes become negligible.

Intra vs Inter Cluster Distance



Intra-cluster distance: The distance between the centroid of a specific cluster and its outermost point.

Inter-cluster distance: The distance between the outermost points of each cluster from each other.

Silhouette Score

The silhouette score for a data point is calculated by $\frac{(b-a)}{max(a,b)}$

1	Ideally, close data points within a cluster and far away from other clusters (good clustering)
0	Data points are on the border between clusters, indicating some overlap (average clustering)
-1	Datapoints might be assigned to the wrong cluster (poor clustering)

If we took K=1, and got a SS of 0.3, then we took K=2, we got an SS of 0.35, then K=3, SS=0.4, then K=4, and we got a SS of 0.6, hence we can iterate over different values of K to find an optimal SS, that is, the closer to 1 the better.

Feature Engineering

ELI5: Imagine you are baking a cake. You don't usually just dump a raw egg or raw wheat into a bowl, instead, you measure the amount of flour to add, and crack the egg into a bowl first. Feature engineering is about outlier handling, transformations, scaling, etc before we dive into an algorithm. It includes remodeling raw data by selecting, combining, remodelling data to improve the quality of our data. Without data preprocessing, we may not be able to rely on the results from the ML algorithm.

The steps included in Feature Engineering

- **Data understanding**
  There may be features we need and do not need in a study, decide or select the most relevant features for the problem at hand. (ANOVA)
- **Feature extraction**
  If we are given DOB, we can create a new column of the age to make findings more interpretable. Calculating sales price * units sold as a new column "total revenue".
- **Feature selection**
  Excluding or including specific features according to the studies' needs, for example, excluding the user ID column is the ML algorithm does not warrant a unique identifier
- **Feature Creation / Transformation**
  Changing a text column such as Gender into a numeric column by encoding male=1, female=0
- **Feature Scaling and Normalisation**
  Bringing features onto the same scale so each feature can contribute equally. For example, scaling age and salary in a dataset that is classified on a boolean predictor: "purchased" because age is in tens, while salary is in thousands. Another example could be something weighed in grams and a price of 10 dollars representing two different things (the computer doesnt know that).

Normalisation: Values are bound between 0 and 1
Standardisation: Transforming data such that the average of the column is 0 and its variance (SD) is 1

Absolute Maximum Scaling [Normalisation]

ELI5: Imagine you have cars of different lengths, the longest car is 10cm long, and the shortest is -8cm long. We then take the absolute maximum, and divide every row by that length. For example:

Car length	Absolute Maximum Scaling
9	9/10 = 0.9
8	8/10 = 0.8
5	5/10 = 0.5
10	10/10 = 1
-8	-8/1- = -0.8
-6	-6/10 = -0.6

It’s a way to shrink values proportionally while keeping their signs and relationships.

In Python

```
import pandas as pd
max_vals = np.max(np.abs(df))
print((df - max_vals) / max_vals)
```

Min-Max Scaling

$scaled\ value\ = \frac{score-min}{max-min}$

Imagine you have a bunch of student's grades, with a minimum of 50 and a maximum of 100. This scales the grades into 0 to 1, 0 being the worst and 1 being the best.

In Python

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data,
columns=df.columns)
scaled_df.head()
```

Mean Normalisation

Same as the min-max method but here instead of the minimum value, we subtract each entry by the mean value of the whole data and then divide the results by the difference between the minimum and the maximum value. $scaled\ value\ = \frac{Xi-Xmean}{Xmax-Xmin}$

Standardisation

First, calculate the mean and SD of the data then we are to subtract the mean value value from each entry and then divide the result by the standard deviation.

In Python

```
from sklearn.preprocessing import
StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data, columns=df.columns)
```

Lecture 09 Example

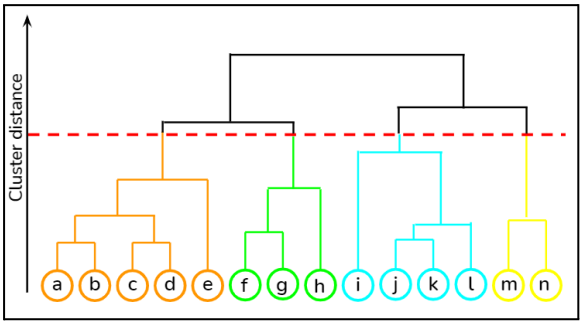
```
# Transform Gender into a binary outcome of 0 or 1
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['Gender'].unique()
```

```
# Extract columns of interest and split the data
x = data.iloc[:, [1,2,3]].values
y = data.iloc[:, 4].values
xtrain,xtest,ytrain,ytest = train_test_split(x,y,train_size=0.25,random_state=0)

# Standardise the x variables to the same scale
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.fit_transform(xtest)
```

Week 09.2

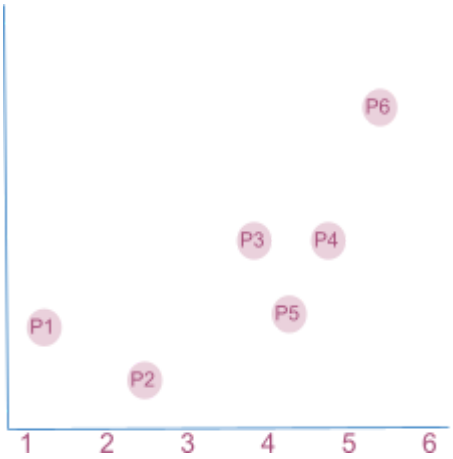
Hierarchical Clustering

Hierarchical clustering helps you find natural groupings in data — even if you don't know how many groups (clusters) there should be. Agglomerative (bottom-up). Divisive (top-down)



Algorithm of Agglomerative Hierarchical Clustering

1. Initialisation
   Treat each data point as a separate cluster
2. Computer Distance Matrix
   Calculate the distance between each pair of clusters (the Euclidean distance). This results in an N x N distance matrix, where the distance between a cluster and itself is zero.



	P1	P2	P3	P4	P5	P6
P1	0	1				
P2	2.5	0				
P3	4	1.5	0			
P4	5	2.5		0		
P5	4.2	2.3			0	
P6	5.5	6	2.6	1.8	3.2	0

In the above matrix, the shortest distance is 1. This is where we should merge P1 and P2 into a single cluster or group. We then re-calculate the distance matrix with -1 data points after 2 points are merged.

The method of recalculating the distance depends on the linkage criterion used.

- Single Linkage
- Complete Linkage
- Average Linkage
- Ward's Method