Welcome to the CoGrammar Logistic Regression

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query:
  [www.hyperiondev.com/support]

- Report a **safeguarding** incident:
  [www.hyperiondev.com/safeguardreporting]

- We would love your **feedback** on lectures: [Feedback on Lectures]

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Logistic Regression

May 2024

Learning Objectives

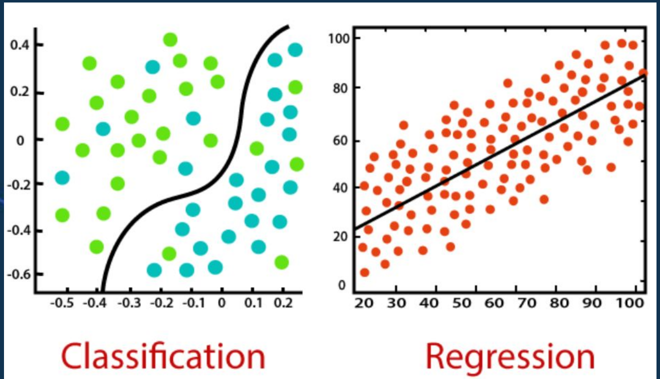
- **Logistic regression** used for solving binary and multi-class **classification problems.**
- Underlying *mathematics* of logistic regression.
- **Preprocess** categorical variables (**label encoding** and **one-hot encoding**), prepare datasets for logistic regression analysis and enhance data preparation skills for **classification tasks**.

Learning Objectives

- Hands-on experience in building, training, and evaluating **classification models**.
- Evaluate the performance: **confusion matrix**, **accuracy, precision, recall,** and **F1 score**.
- Analyse and interpret the confusion matrix (**false positives** and **false negatives**).

CoGrammar

Logistic Regression



CoGrammar

Logistic Regression

- **Linear regression** models make **predictions** for the datasets for which dependent variables have **continuous numerical values.**

- **Logistic Regression**
  - **supervised learning** algorithm
  - **classification** algorithm
  - dependent variables are **distinct, non-continuous, categorical**

- **Classification** - predicting **probability** of **categorical variables** for a given observation and assigning the observation to the category with the highest probability.

CoGrammar

Logistic Regression

- **Binary (Binomial) logistic regression:** Response or dependent variable has **only two possible outcomes** (e.g. 0 or 1, True or False, Malignant or Benign tumour, Spam or Not Spam email).

- **Multinomial logistic regression:** Dependent variable has **three or more possible outcomes**; but values have **no specified order** (e.g., movie studios predicting film genres depending on person's age, gender, family status).

- **Ordinal logistic regression:** Response variable has **three or more possible outcome**, and values **have a defined order** (e.g. grading scales from A to F or rating scales from 1 to 5).

Logistic function

CoGrammar

Logistic function

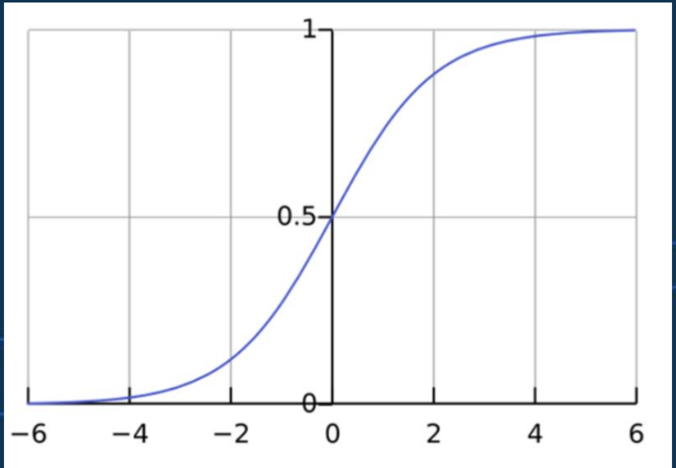
- Logistic regression: statistical model that uses the **logistic (logit) function**, as the equation between x and y (also called **sigmoid function** or **S-shaped curve**).

- Returns only values between 0 and 1 for the dependent variable, irrespective of the values of the independent variable.

- Also model equations between **multiple independent variables** and **one dependent variable.**

Sigmoid function

$$p = \frac{1}{(1 + e^{-y})}$$



CoGrammar

Linear vs. Logistic Regression

Linear regression: Continuous value of output y for given input/s X.

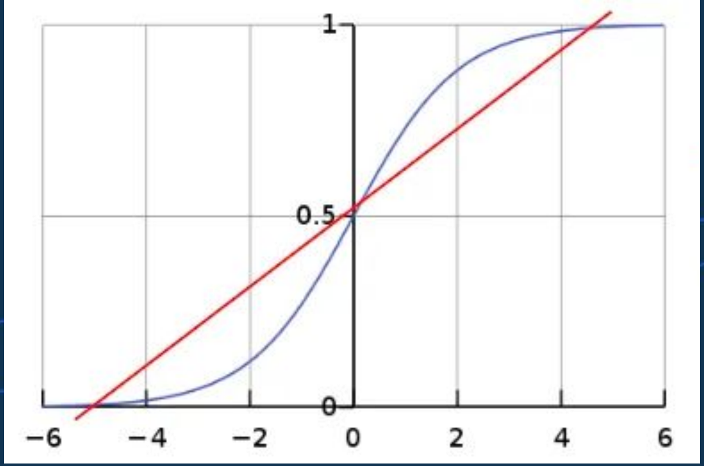
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \text{ (red line)}$$

Logistic regression

Sigmoid function $$p = \frac{1}{(1 + e^{-y})} \text{ (blue line)}$$

Log Odd $$\ln \frac{p}{(1-p)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

- Uses same underlying formula as linear regression but it is regressing for the **probability** of a **categorical** outcome.
- Gives continuous value of P(y=1) for given input/s X, which is later converted to y=0 or y=1 based on threshold value.
- Task is to **predict** different **class labels**.
- Uses **Sigmoid function** to predict output class label for given input.



CoGrammar

Assumptions of Logistic Regression

- The **independent variables** should **not be correlated** with each other i.e. the model should have little or **no multicollinearity**.
- The **dependent variable** must be **categorical** in nature.
- The relationship between the **independent variables** and the **log odds** of the dependent variable should be **linear**.
- There should be **no outliers i**n the dataset.
- The data sample size should be **sufficiently large.**

CoGrammar

Implementing Logistic Regression

Using scikit-learn

CoGrammar

Logistic Regression Example

We will build a **logistic regression model** to predict whether an individual is a **smoker** based on features like **age, sex, BMI, number of children, region**, and **insurance charges**.

- ❖ Loading and preprocessing the data.
- ❖ Exploratory data analysis.
- ❖ Feature encoding, data normalization.
- ❖ Model training and evaluation, making predictions.

```
from sklearn.linear_model import LogisticRegression
```

CoGrammar

Logistic Regression Example

```python
# Import Libraries
import pandas as pd

#Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# Load the dataset
file_path = 'insurance.csv'
data = pd.read_csv(file_path)

data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Logistic Regression Example

Choosing the categorical variables

```
object_columns = data.columns[data.dtypes == 'object']
object_columns
```

```
Index(['sex', 'smoker', 'region'], dtype='object')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

CoGrammar

Data Preprocessing

Categorical Encoding

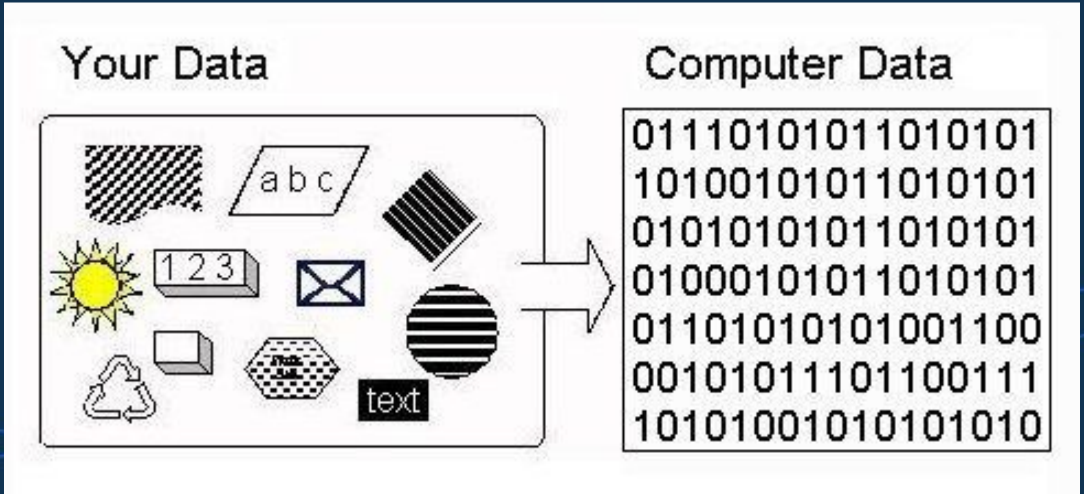
CoGrammar

Categorical Encoding

- Structured datasets include numerical and categorical columns.

- **Categorical encoding:** converting categorical columns to numerical columns for a machine learning algorithm to understand.

- Process of converting categories to numbers.



CoGrammar

Categorical Encoding

- **Label Encoding**
  - Assigns a unique integer or alphabetical ordering to represent each label/category,
  - Suitable for categories with an **inherent/intrinsic order** or **rank**
  - E.g. For performances, Poor, Fair, Good, Very Good, Excellent, assign the numbers [1, 2, 3, 4, 5].

- **One-Hot Encoding:**
  - Instead of giving each category a single number, it creates a new binary column (1 or 0) for each unique category.
  - Suitable for **nominal data**, e.g. colors or car brands, where there is no inherent order.
  - Each category is represented as a one-hot vector.

Categorical Encoding

```python
# Encode categorical columns
from sklearn.preprocessing import LabelEncoder

for column in object_columns:
    data[column] = LabelEncoder().fit_transform(data[column])

data.apply(LabelEncoder().fit_transform)
data.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520

CoGrammar

Example: LabelEncoder issues

Players	Runs in ODI
Atherton	102
Broad	92
Flintoff	83
Root	112
Stokes	75
Broad	77
Root	97

LabelEncoder Player data

Players	Runs in ODI
0	102
1	92
2	83
3	112
4	75
1	77
3	97

- ❖ Since the names are alphabetically ranked, the model may capture incorrect correlation between players such as Atherton < Root < Stokes, which might not be true in another data or prediction set.

- ❖ Label Encoded player names into numerical data.
- ❖ Player names do not have an order or rank.

CoGrammar

Use OneHotEncoder instead

Example: Solving with OneHotEncoder

OneHotEncoder Player data

	Players_Atherton	Players_Broad	Players_Flintoff	Players_Root	Players_Stokes	Runs in ODI
0	1.0	0.0	0.0	0.0	0.0	102
1	0.0	1.0	0.0	0.0	0.0	92
2	0.0	0.0	1.0	0.0	0.0	83
3	0.0	0.0	0.0	1.0	0.0	112
4	0.0	0.0	0.0	0.0	1.0	75
5	0.0	1.0	0.0	0.0	0.0	77
6	0.0	0.0	0.0	1.0	0.0	97

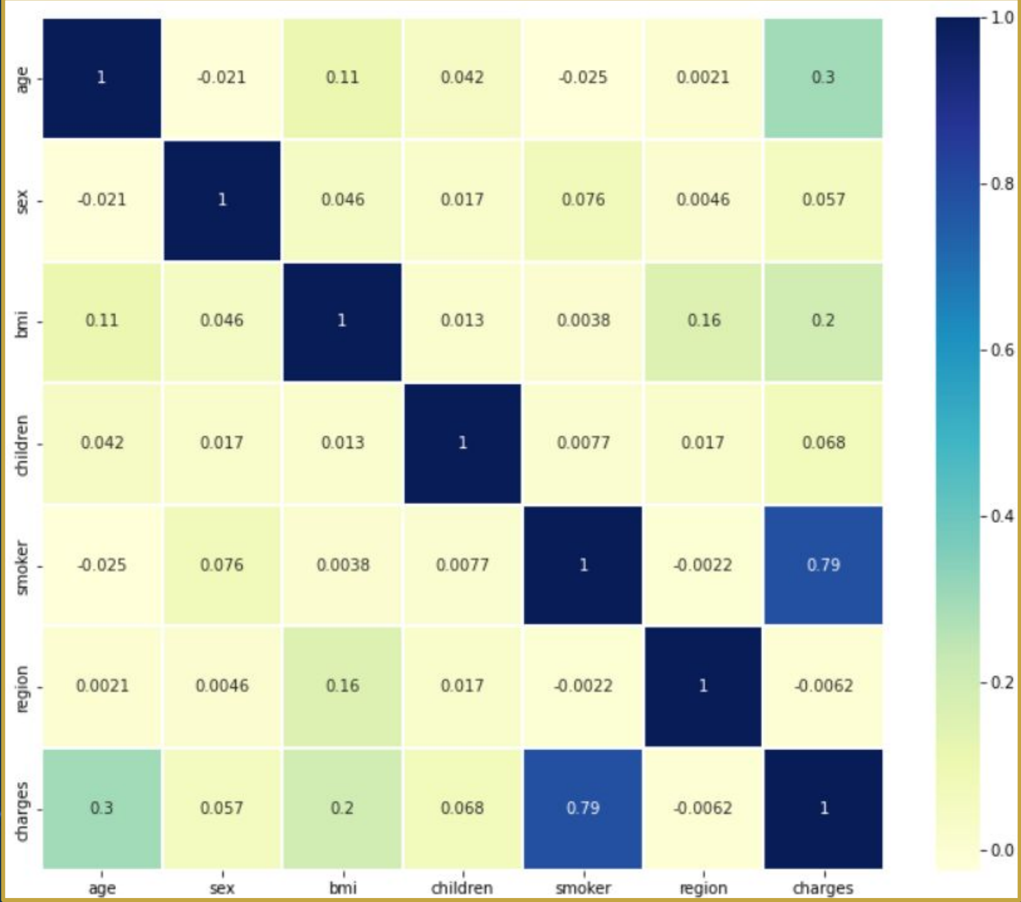
Drawback: One hot encoding requires as many new variables as there are unique values in the original categorical variable. If the categorical variable has 100 unique values, 100 new variables will be created when using one hot encoding, complicated for larger datasets.

Feature Correlations

CoGrammar

Feature Correlations

```python
# Plot the heatmap to visualize feature correlations
sns.heatmap(data.corr(), annot=True, cmap='YlGnBu')
plt.show()
```



Dataset Splitting

train-test-split

CoGrammar

Data splitting

```python
# Splitting the dataset into features and target
X = data.drop('smoker', axis=1)
y = data['smoker']

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model fitting and prediction

```python
# Fit the logistic regression model
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train_scaled, y_train)
```

```python
# Predict the model
y_pred = log_reg.predict(X_test_scaled)
```

CoGrammar

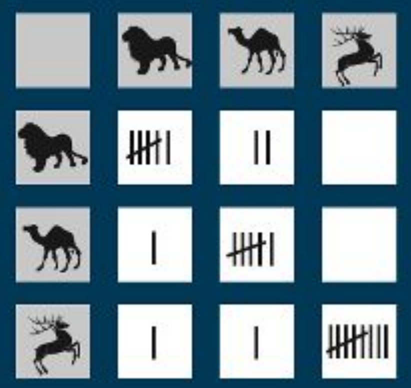
Evaluating Metrics for Classification

CoGrammar

Evaluation Metrics

Confusion matrix

- NxN table for evaluating the performance of a classification model (N = number of target classes, N = 2 for binary), summarises classification model's predictions.
- Compares the **actual target values** (on one axis) with those **predicted by the machine learning model** (on the other axis).
- Gives insights into the behaviour of the classifier
- However, we need **evaluation metrics** to make claims about whether the model did well or not compared to other models.



CoGrammar

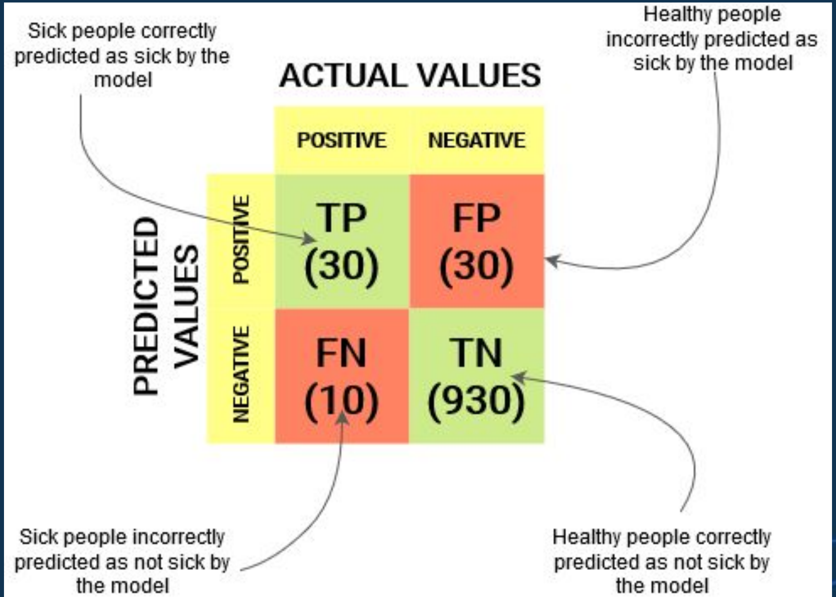
Why a Confusion Matrix?

Hypothetical Classification Problem

Predict how many people are infected with a contagious virus in times before they show the symptoms and isolate them from the healthy population. The two values for our target variable would be **Sick (Positive)** and **Not Sick (Negative)**.

Why a Confusion Matrix?

- **True positives (TP)**: No. of samples correctly predicted as positive.
- **True negatives (TN)**: No. of samples correctly predicted as negative.
- **False positives (FP)**: No. of samples incorrectly predicted as positive.
- **False negatives (FN)**: No. of samples incorrectly predicted as negative.



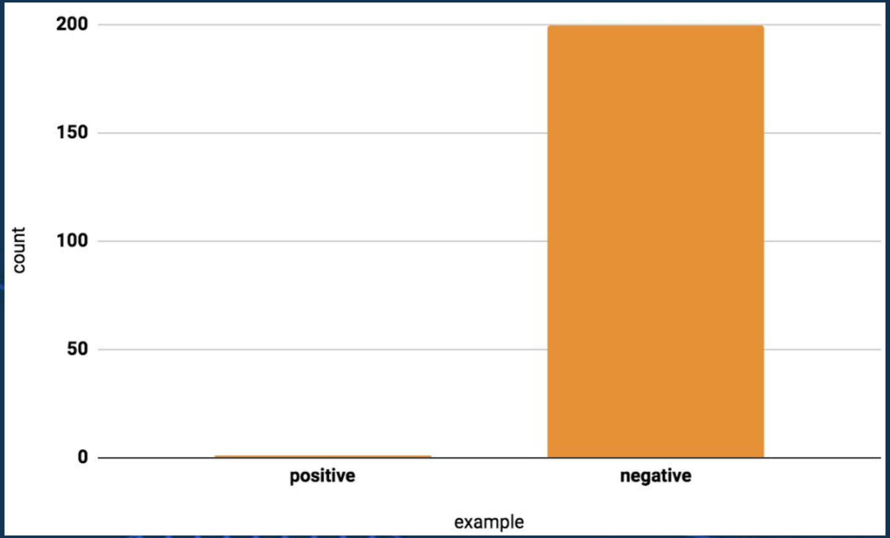
Sick (Positive) and **Not Sick (Negative)**.

CoGrammar

https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

Why a Confusion Matrix?

Useful for Imbalanced dataset: Target class has an uneven distribution of observations, i.e one class label has very high number of observations and the other has very low number of observations.



Example: **Credit card frauds** happen once per 200 transactions, ~ 0.5% of data is positive.

With so few positives relative to negatives, the training model will spend most of its time on negative examples and not learn enough from positive ones.

Other examples:
Disease diagnosis
Customer churn prediction
Natural disasters

CoGrammar

Accuracy

Accuracy of classifier: Total number of correct predictions by the classifier divided by the total number of predictions.

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

For virus example, **Accuracy = 96%**

According to the **Accuracy** value, the model "can predict sick people 96% of the time". However, it is **predicting the people who will not get sick with 96% accuracy while the sick are spreading the virus.**

Better to measure how many **positive cases we can predict correctly** to arrest spread of the contagious virus or **out of the correct predictions**, how many **are positive cases** to check the reliability of the model.

Precision and Recall

- **Precision:** tells us how many of the correctly predicted cases actually turned out to be positive, determine whether the model is reliable or not.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: how many of the actual positive cases we were able to predict correctly with our model.

$$Recall = \frac{TP}{TP + FN}$$

For virus example, Precision = 50%, Recall = 75%

For virus example, 50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by the model.

CoGrammar

Precision and Recall

- **Precision**: useful in cases where **False Positive** is a greater concern.
- *Music* or *video recommendation systems, e-commerce websites.*
- *Wrong results* could lead to customer churn and be *harmful* to the business.

- **Recall:** useful in cases where **False Negative** trumps.
- *Medical cases* where it does not matter whether a false alarm flag is raised, but the *actual positive cases should not go undetected*.

For **contagious virus example**, the **Confusion Matrix** is more insightful measure in such critical scenarios.

Recall, assessing the ability to capture all actual positives, emerges as a **better metric**. **Accuracy** proves **inadequate** as a metric for the model's evaluation.

Avoid mistakenly releasing an infected person into the healthy population, potentially spreading the virus.

CoGrammar

F1-score

- Cases where there is no clear distinction between whether Precision is more important or Recall.

- **F1-score:** harmonic mean of **Precision** and **Recall**, gives a combined idea about these two metrics, appropriate metric for imbalanced dataset.

- **Maximum** when **Precision** is **equal** to **Recall**.

- Use in combination with other evaluation metrics.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

CoGrammar

Metrics using scikit-learn

```python
#Evaluate the model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

print("Accuracy Score:",accuracy_score(y_pred, y_test))
print("Confusion Matrix: \n",confusion_matrix(y_pred, y_test))
print("Classification Report: \n " ,classification_report(y_pred, y_test))
```

Classification report: Precision, Recall, and F1-score for each target class.

Macro average = average of Precision / Recall /F1-score.

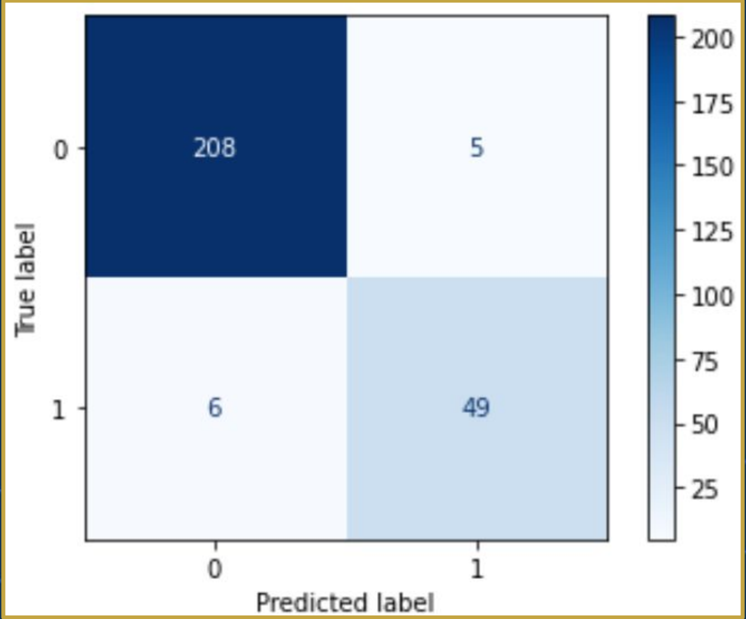
Weighted average of Precision / Recall / F1-score.

```
Accuracy Score: 0.9589552238805971
Confusion Matrix:
 [[208   5]
 [  6  49]]
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.98      0.97       213
           1       0.91      0.89      0.90        55

    accuracy                           0.96       268
   macro avg       0.94      0.93      0.94       268
weighted avg       0.96      0.96      0.96       268
```

CoGrammar

Metrics using scikit-learn

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_pred, y_test,labels=log_reg.classes_)
# sns.heatmap can also be used to get the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=log_reg.classes_)
disp.plot(cmap='Blues')
```



CoGrammar

Metrics: Key Takeaways

- **TP** and **TN** values mean the predicted value matches the actual value.
- Ideally, we want both **precision** & **recall = 1**, but this seldom is the case.

- **Low Precision/High Recall**: Cases where we need to **reduce** the number of **FN** without necessarily reducing the number of FP.
  - **Cancer diagnosis:** We do not want *any affected patient to be classified as not affected (FN)* without giving much heed to if the *patient is being wrongfully diagnosed with cancer (FP)*. Absence of cancer can be detected by further tests, but presence of the disease cannot be detected in an already rejected candidate.

- **High Precision/Low Recall:** Cases where we need to **reduce** the number of **FP** without necessarily reducing the number of FN.
  - **Personalised advertisement**: We want to be absolutely sure that the customer will *react positively to the advertisement* because otherwise, a *negative reaction can cause a loss of potential sales from the customer*.

Example: Multiclass Confusion Matrix

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

CoGrammar

Summary

CoGrammar

Key Takeaways from Logistic Regression

- Fundamental **classification** technique, provides a **probability** score for observations.
- **Efficient** and **straightforward** **linear classifier**, does not require high computation power, easily implemented and interpreted, used widely.

- Not able to handle a **large number of categorical features/variables**.
- Vulnerable to **overfitting**.
- Cannot handle **non-linear features**, requires a transformation
- Do not perform well with **independent variables** that are **not correlated** to the **target variable** and are very similar or correlated to each other.

CoGrammar

Further Resources

- https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/
- https://realpython.com/logistic-regression-python/
- https://www.geeksforgeeks.org/understanding-logistic-regression/
- https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar