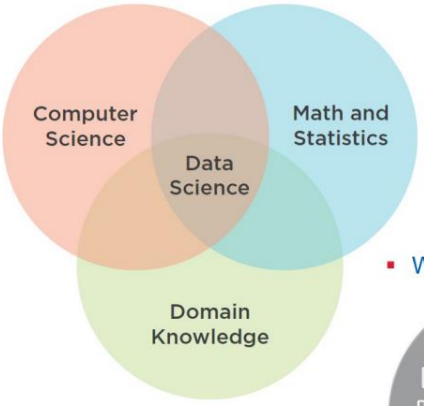
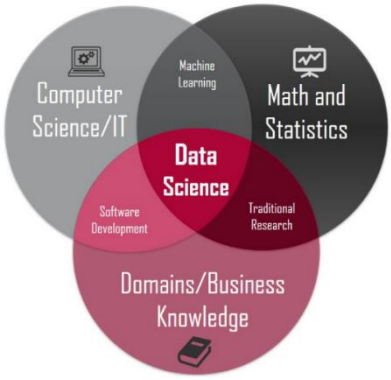


Data Science

+- 2.5 Exabytes (2.5 billion gigabytes) Per Day

- 1 exabyte = 1,000 petabytes (PB)
- 1 exabyte = 1,000,000 terabytes (TB)
- 1 exabyte = 1,000,000,000 gigabytes (GB)
- 1 exabyte = 1,000,000,000,000 megabytes (MB)
- 1 exabyte = 1,000,000,000,000,000 kilobytes (KB)

▪ What is Data Science:



Data Science Definitions:

- Set of fundamental principles that guide the extraction of knowledge from data.
- Field of study that combines domain expertise, programming skills, and knowledge of maths and statistics to extract meaningful insights from data

Data Scientist Skills:

♣ Programming

♣ Working with data

♣ Descriptive statistics

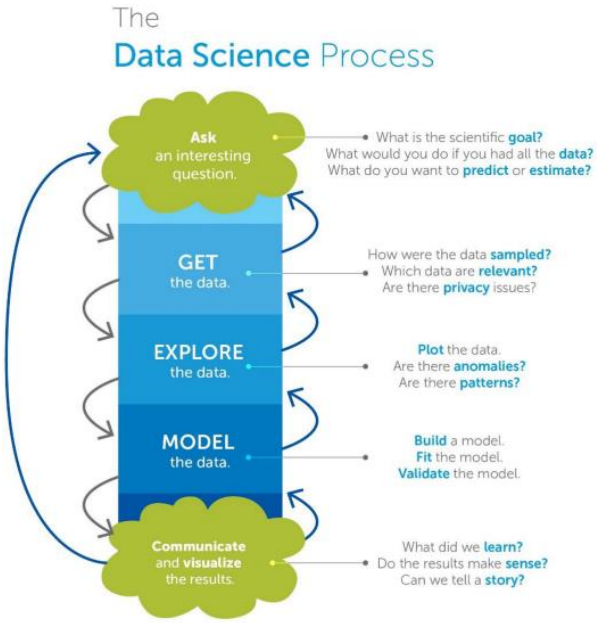
♣ Data visualization

♣ Statistical modeling

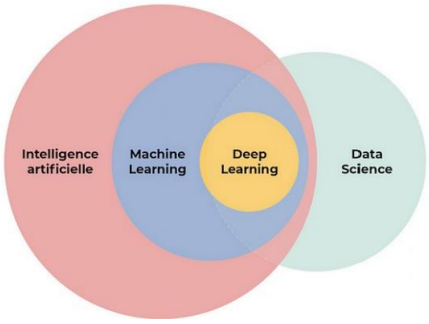
♣ Handling Big Data

♣ Machine learning

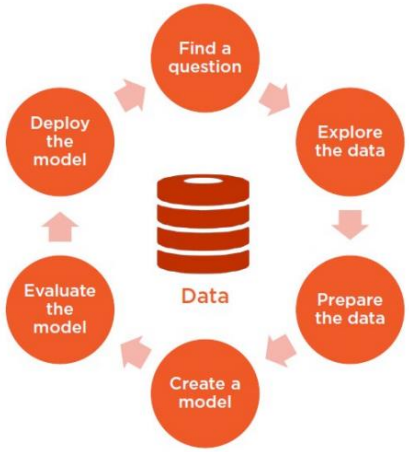
♣ Deployment of solutions



▪ Where Data Science is situated:



▪ The Data Science Process (another example):



What is a Data Scientist:

♣ Is a Scientist ♣

and ♣ Is a Developer

♣ and ♣ Is an Analyst

♣ i.e. They perform data science

Importance of Data Exploration and Visualization in Data Science

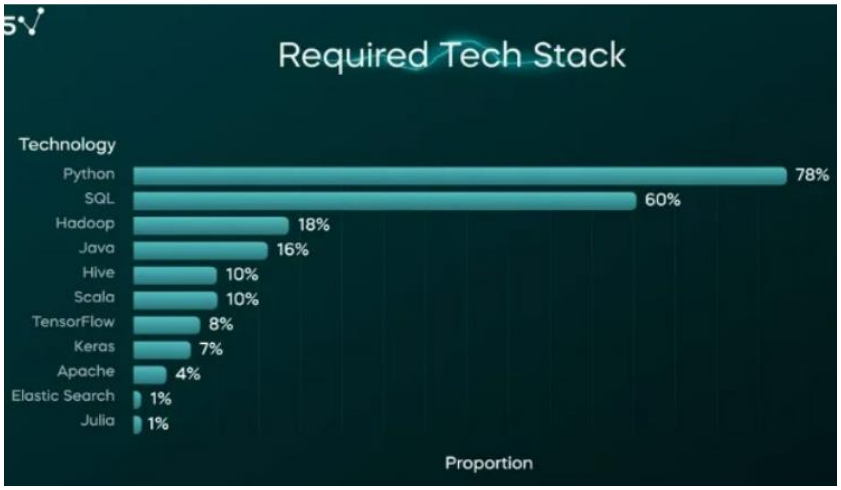
• Data Exploration and Visualization play a fundamental and pivotal role in the field of Data Science, serving as the cornerstone of informed decision-making and insightful analysis. Their importance lies in their ability to unearth hidden patterns, trends, and relationships within complex datasets, translating raw information into actionable insights.

• **Data Quality Assessment**: Visualization aids in identifying data quality issues such as outliers, missing values, or inconsistencies. These visual cues guide data cleaning and preprocessing, enhancing the accuracy and reliability of subsequent analyses.

• **Feature Selection and Engineering**: Effective data exploration helps in selecting relevant features for modelling while also inspiring the creation of new features that might enhance predictive performance. This contributes to more robust and accurate machine learning models.

• **Enhanced Creativity**: Visualization encourages creative thinking and innovative problem-solving. Exploring data visually can lead to the discovery of new angles or perspectives that may not have been evident through traditional numerical analysis alone.

Data Science Tools:



8 Automated EDA Tools (saves manual work time)

- SweetViz
- Pandas-Profiling
- DataPrep
- AutoViz
- D-Tale
- Dabl
- QuickDA
- Lux

Exploratory Data Analysis (EDA)

•Generating descriptive statistics and visualizations.

•Distribution plots, histograms, and box plots.

•Correlation analysis and scatter

Data Cleaning and Preprocessing

• **Handling Missing Values, Duplicates, and Outliers:**

• Missing Values: When data points are missing, it can lead to skewed analysis and biased results. Handling missing values involves imputing or removing them. For example, in a dataset of survey responses, some participants might not have provided their age. You can impute missing ages with the median age of the respondents.

• Duplicates: Duplicate entries can distort analysis and lead to overrepresentation. Detecting and removing duplicates ensures data integrity. For instance, in an e-commerce dataset, there might be multiple identical orders due to system errors. Removing duplicates ensures accurate order counts.

• Outliers: Outliers are data points that significantly deviate from the norm. Outliers can affect statistical measures and model performance. Addressing outliers may involve removing or transforming them. In a dataset of exam scores, a single exceptionally high score might be an outlier. You can replace it with a more reasonable value based on the distribution of scores.

Data Transformation and Normalization Techniques

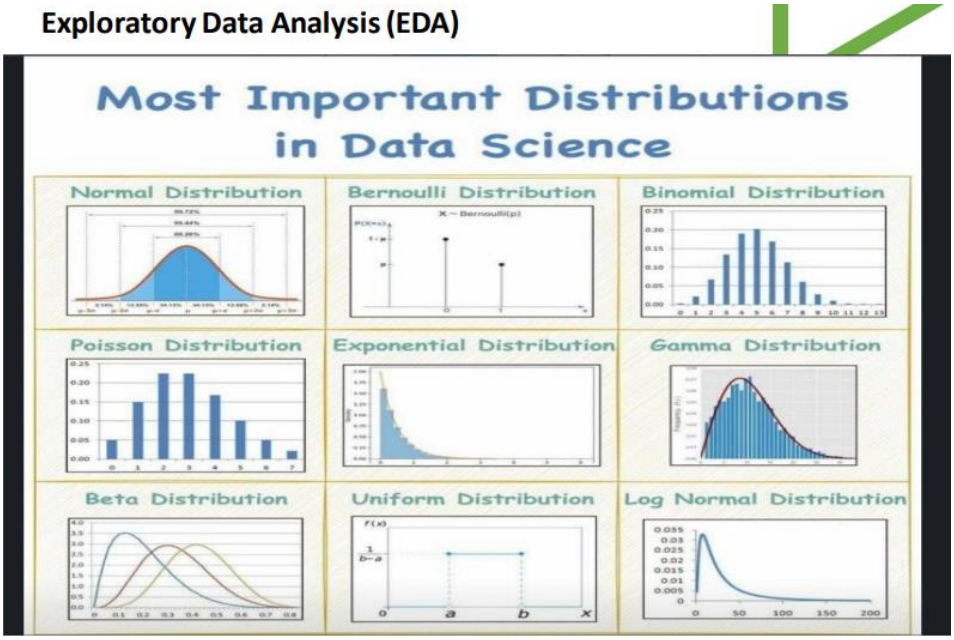
• Data Transformation: Data transformation involves converting data into a more suitable format or scale. For instance, transforming skewed data using logarithms can help achieve a more normal distribution. In a dataset of income levels, applying a logarithmic transformation can reduce the impact of extreme incomes.

• Normalization: Normalization scales data to a common range, making comparisons meaningful. In machine learning algorithms, normalized data prevents certain features from dominating others. For example, consider a dataset with attributes like age (0-100) and income (0-100000). Normalizing these features to a 0-1 range ensures balanced contributions to the analysis.

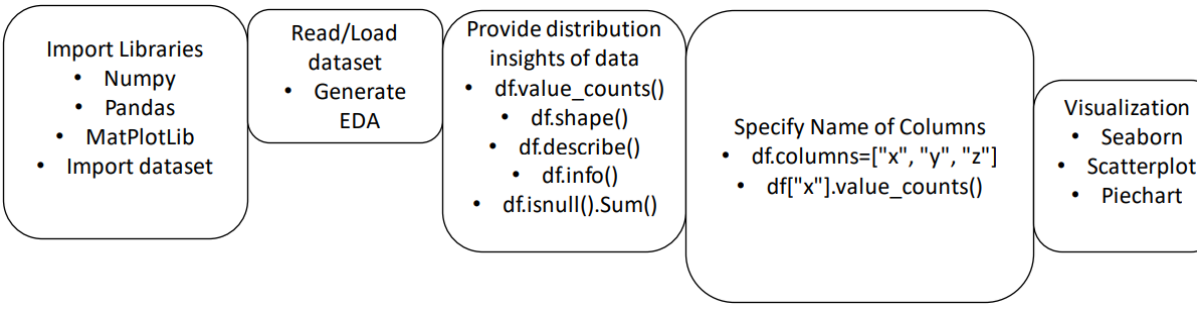
• Standardization: Standardization scales data to have zero mean and unit variance. This is particularly useful for algorithms that assume normally distributed data. For instance, in a dataset containing height and weight, standardization ensures that both attributes contribute equally to clustering algorithms.

• Encoding Categorical Variables: Machine learning models often require numerical inputs. Categorical variables (e.g., "red," "blue," "green") need to be encoded into numerical values.

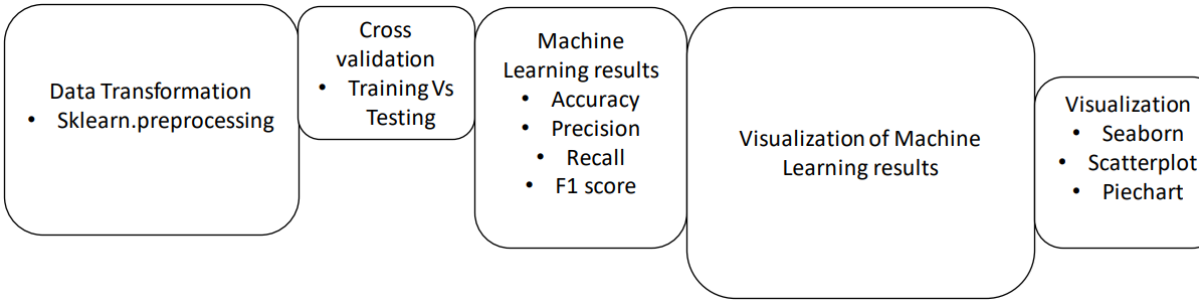
Exploratory Data Analysis (EDA)



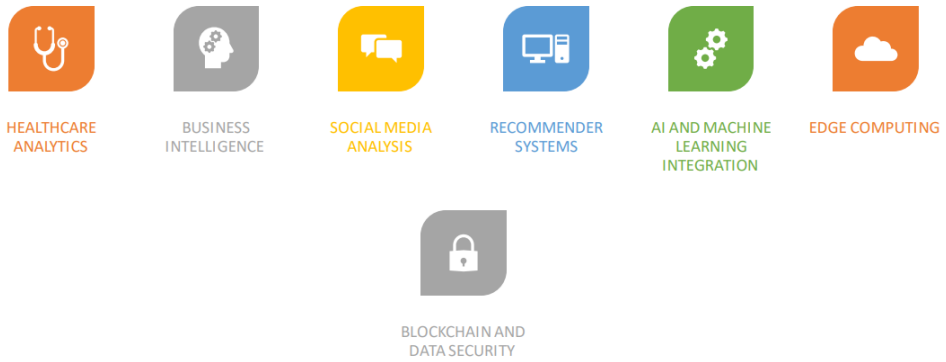
Data Processing Framework



Data Processing Framework: Data Transformation for Machine Learning



Applications of Data Mining and Big Data



Definition of Data Mining

- Data Preparation: Cleaning, transforming, and preprocessing the data to make it suitable for analysis. This includes handling missing values, dealing with outliers, and formatting data.
- Pattern Discovery: Employing algorithms to identify patterns, associations, correlations, or anomalies within the data.
- Predictive Modelling: Building predictive models that can forecast future trends or outcomes based on historical data.
- Clustering and Classification: Grouping similar data points into clusters or categorizing data into predefined classes or categories.
- Data Visualization: Presenting the results of data mining in visual formats such as charts, graphs, or reports for easier interpretation.

The Importance of Big Data in Data Mining

Real-Time Decision-Making: This is critical in applications like autonomous vehicles, where data must be analyzed instantly to ensure safety.

Scientific Discovery: In fields like genomics, particle physics, and climate science, big data plays a crucial role in scientific discovery. Researchers can process and analyze massive datasets to uncover new knowledge and make groundbreaking discoveries.

Identification of Rare Events: Big data facilitates the detection of rare events or anomalies. In fields like fraud detection, cybersecurity, and quality control, data mining techniques can identify unusual patterns or behaviors that might signal a problem or threat.

Improved Predictions: Larger datasets enable data mining models to make more accurate predictions. By analyzing vast amounts of historical and current data, these models can identify trends and patterns that lead to better forecasting, whether in financial markets, consumer behavior, or healthcare outcomes.

Data Preprocessing

- Involves identifying and correcting errors or inconsistencies in a dataset to improve its quality.
- Example: In a customer database, there may be entries with missing values, such as email addresses or phone numbers.
- Data cleaning would involve either filling in these missing values with reasonable estimates or removing records with too many missing values.

Data Integration

Data integration combines data from different sources into a unified view, providing a comprehensive and coherent dataset.

Example: An e-commerce company integrates data from various systems, including sales records, inventory databases, and customer information, to create a complete view of its business operations.

Data Transformation

- Data transformation involves converting data into a different format, structure, or scale to make it suitable for analysis or a specific application.
- Example: Converting temperature readings from Fahrenheit to Celsius is a data transformation. Similarly, aggregating daily sales data into monthly totals is a transformation to a different scale.
- Data reduction aims to decrease the volume but produce the same or similar analytical results. It often involves selecting a representative subset of data.
- Example: In machine learning, feature selection is a form of data reduction. Choosing the most relevant features (variables) from a dataset while discarding less important ones can improve model performance and reduce computational costs.

EDA

- EDA is the process of summarizing, visualizing, and understanding the main characteristics of a dataset before conducting formal analyses.
- Example: Before building a predictive model, a data scientist might use EDA techniques to create histograms, scatter plots, and summary statistics to uncover trends, anomalies, or relationships in the data.

Data Analysis

- Univariate analysis focuses on a single variable or feature in a dataset to understand its distribution, central tendencies, and variability.
- Example: If you want to understand the distribution of ages in a population, you can create a histogram or a box plot showing the frequency or density of different age groups.
- Bivariate analysis involves analysing the relationship or association between two variables in a dataset.
- Example: To study how the price of a product is affected by its advertising spending, you can create a scatter plot with advertising spending on the x-axis and product price on the y axis.

- Multivariate analysis deals with the simultaneous analysis of more than two variables in a dataset to uncover complex relationships and patterns.
- Example: Principal Component Analysis (PCA) is a multivariate technique that reduces the dimensionality of a dataset while preserving as much variance as possible. It's often used for feature reduction in machine learning.
- Visualization Techniques: involves representing data graphically to provide insights, identify trends, or convey information effectively
- Example: Creating a heatmap to visualize the correlation matrix of variables in a dataset. High correlations are shown in one color, while low correlations are shown in another.

Data Mining

- Data mining techniques are methods used to extract patterns, knowledge, or valuable information from large datasets.
- Example: Using association rule mining to identify patterns in customer purchase data, such as discovering that customers who buy cereal and milk are also likely to buy bread.

Classification and Prediction

- Classification involves assigning data instances to predefined categories or classes based on their characteristics.
- For example, classifying emails as spam or not spam based on their content.
- Prediction aims to estimate a numerical or categorical outcome for a data instance based on historical data.
- For instance, predicting a student's future GPA based on their past academic performance.
- Clustering groups similar data points together based on their inherent characteristics, without predefined categories
- Example: Clustering customer data to segment them into distinct groups, such as "high spenders," "budget shoppers," and "window shoppers," based on their shopping behaviour.

Association Rule Mining

- Association rule mining identifies patterns or relationships between items in a dataset. It's commonly used in market basket analysis to discover which items are frequently purchased together.
- Example: Supermarkets analyzing customer purchase data to find associations like "Customers who buy bread also tend to buy butter."
- Anomaly Detection: Anomaly detection identifies data instances that deviate significantly from the norm or exhibit unusual behavior. It's used to find rare events or outliers
- Example: Detecting fraudulent credit card transactions by identifying transactions that significantly differ from a user's typical spending behavior

Machine Learning

- Machine learning algorithms are computational techniques that enable computers to learn and make predictions or decisions from data without being explicitly programmed.
- Example: Using a decision tree algorithm to predict whether a loan applicant is likely to default based on features like credit score, income, and loan amount.

Unsupervised Learning

- In supervised learning, the algorithm learns from a labeled dataset, where each input data point is associated with a corresponding target or output label.
- The algorithm's goal is to learn a mapping or relationship between the input data and the output labels. It aims to make predictions or classifications based on new, unseen data.
- Example: Consider a spam email classifier. You have a dataset of emails, and each email is labeled as either "spam" or "not spam" (ham). The features of these emails, such as the words used and email metadata, serve as input data.
- In supervised learning, the algorithm learns from this dataset to predict whether incoming emails are spam or not based on their features. It uses the labels (spam or not spam) to train and fine-tune its predictive model.

Supervised Learning

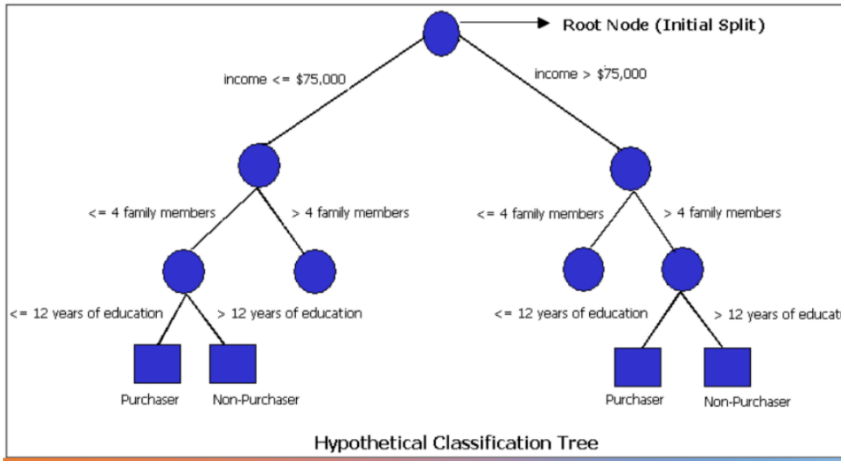
- In unsupervised learning, the algorithm works with unlabelled data, meaning it doesn't have access to explicit target labels or categories. Instead, the algorithm tries to discover patterns, structures, or relationships within the data without any prior guidance
- Unsupervised learning is often used for tasks like clustering, dimensionality reduction, and anomaly detection.
- Example: Imagine you have a dataset of customer purchase history, including what items they bought and when. In unsupervised learning, you might use clustering to group similar customers together based on their purchase behaviour. The algorithm would identify patterns within the data, such as customers who frequently buy electronics, customers who prefer clothing, etc., without any predefined categories. This can help businesses understand customer segments for targeted marketing.

In summary, supervised learning relies on labelled data to make predictions or classifications, while unsupervised learning seeks to discover patterns and structures in unlabeled data. Both approaches have distinct use cases and are essential in various machine learning applications.

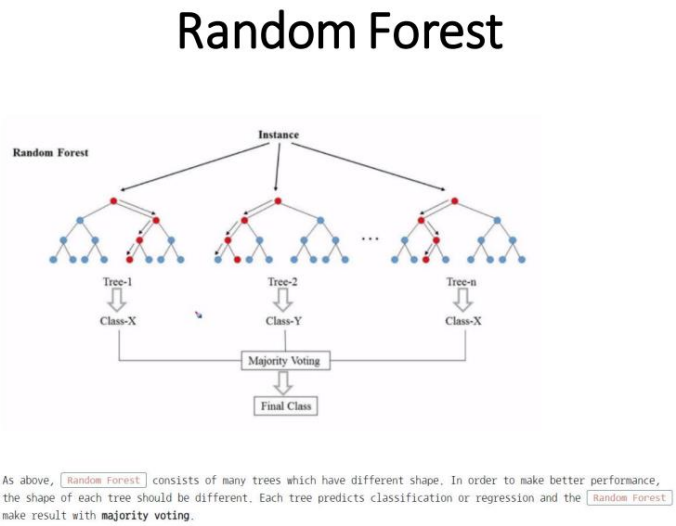
Machine learning Algorithms

- Decision trees are a supervised learning algorithm used for classification and regression tasks. They partition the input data into subsets based on features and recursively make decisions at each node to arrive at a final prediction. Random Forests, on the other hand, are an ensemble method that consists of multiple decision trees. They work by

aggregating the predictions of multiple decision trees to improve accuracy and reduce overfitting.



Hypothetical Classification Tree



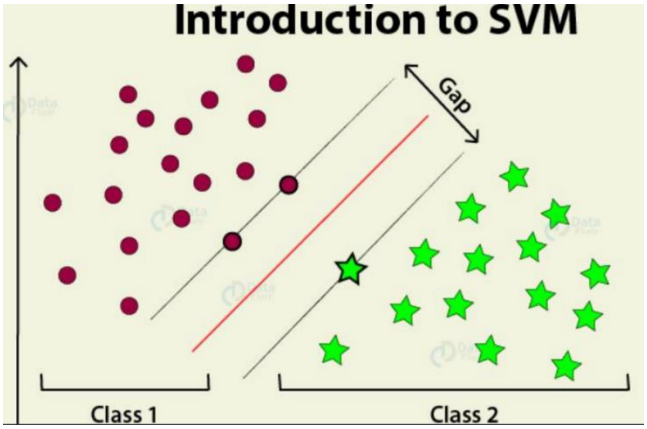
As above, Random Forest consists of many trees which have different shape. In order to make better performance, the shape of each tree should be different. Each tree predicts classification or regression and the Random Forest make result with **majority voting**.

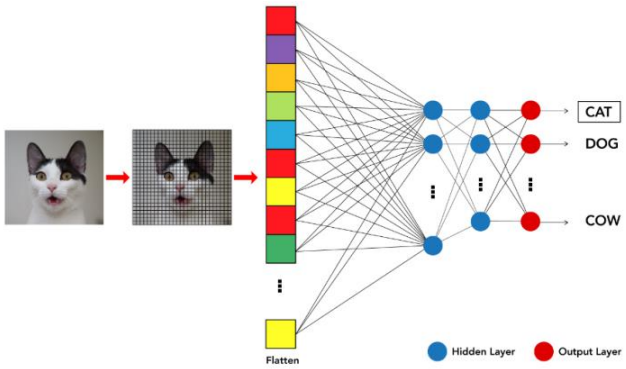
- Decision trees are simple to understand and interpret but can be prone to overfitting when they become too deep.
- Random Forests, on the other hand, mitigate overfitting by averaging the predictions of many decision trees, making them more robust and accurate. While decision trees can be used for both classification and regression, Random Forests are primarily used for classification tasks.
- Supervised or Unsupervised: Both Decision Trees and Random Forests are supervised learning algorithms because they require labeled data during training. In the case of classification, they need labeled examples to learn the decision boundaries.

SVM

- SVM is a supervised learning algorithm used for classification and regression tasks. It finds a hyperplane or decision boundary that best separates different classes in the input data while maximizing the margin between them.
- SVM can also handle non-linear classification by using kernel functions.
- Difference: SVM aims to find the optimal decision boundary that maximizes the margin between data points of different classes. It is effective in high-dimensional spaces and can handle nonlinear data using kernel tricks.
- SVM tries to find the "best" boundary by maximizing the margin, **while other algorithms like decision trees might create simpler boundaries based on recursive splits**
- Supervised or Unsupervised: SVM is a supervised learning algorithm, as it relies on labeled data during training to learn the decision boundary.

Neural Networks and Layers

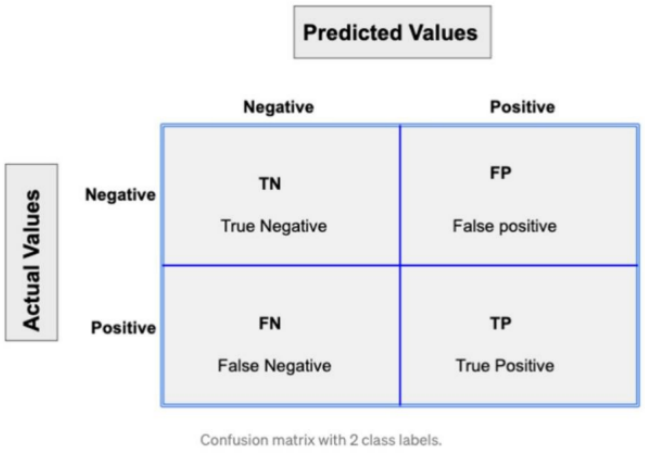




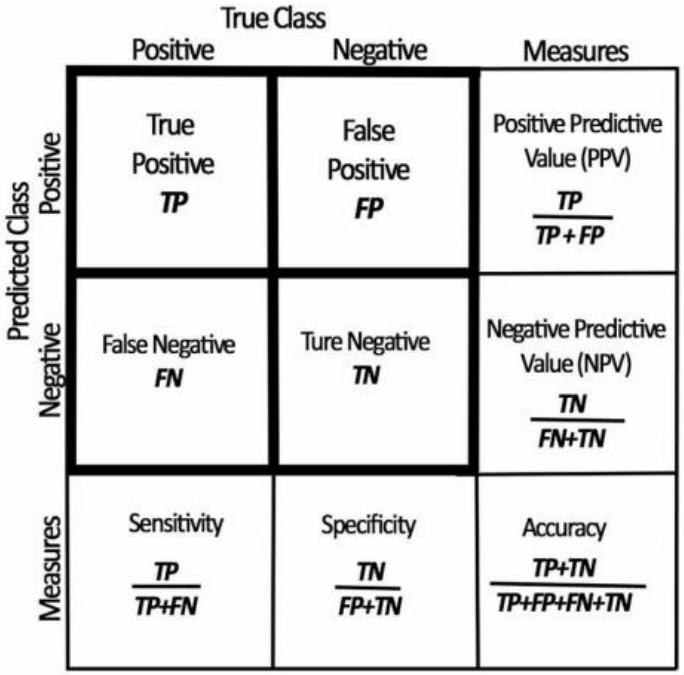
Neural Networks

- Neural networks are a family of machine learning algorithms inspired by the structure and function of the human brain.
- Deep Learning is a subfield of machine learning that focuses on neural networks with many layers (deep neural networks). Neural networks consist of interconnected nodes or neurons that process and transform data through multiple layers to make predictions.

Machine Learning Evaluation



Confusion matrix with 2 class labels.



Challenges of Mining Big Data

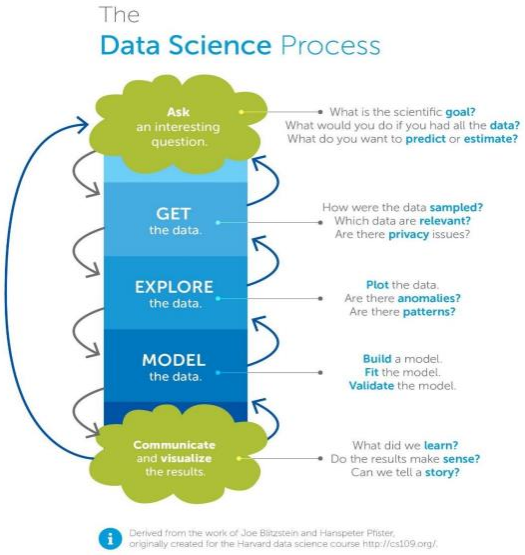
- Volume: Big data involves massive volumes of information that traditional data mining tools and techniques struggle to handle efficiently.
- Velocity: Data streams in at an unprecedented speed, such as social media updates, sensor data, and financial transactions. Realtime or near-real-time processing is essential to extract valuable insights promptly.
- Veracity: Big data often contains noisy, incomplete, or inaccurate information. Ensuring data quality and dealing with uncertainties can be a substantial challenge in mining bigdata effectively.
- Value: Identifying valuable insights from big data can be challenging. With vast amounts of data, it's easy to get lost in irrelevant information.
- Privacy and Security: The sheer amount of data increases the risk of privacy breaches and security threats. Safeguarding sensitive information while allowing datamining is a significant challenge
- Scalability: Traditional data mining algorithms may not scale efficiently to handle big data. Developing scalable algorithms and distributed computing solutions is essential.
- Interoperability: Integrating big data tools and platforms into existing IT infrastructures can be complex. Ensuring that new technologies work seamlessly with legacy systems is a challenge.
- Regulatory Compliance: Big data mining must adhere to various regulations and legal constraints, such as GDPR in Europe or HIPAA in healthcare. Ensuring compliance while mining data can be challenging.
- Resource Constraints: Processing and storing big data require significant computational and storage resources. Ensuring access to these resources can be a challenge, especially for smaller organizations.
- Ethical Concerns: As data mining becomes more pervasive, ethical concerns regarding data privacy, surveillance, and potential biases in algorithms need to be addressed.
- Complexity: Big data projects often involve complex data preparation, feature engineering, and modelling processes. Managing this complexity and ensuring the interpretability of results can be challenging.

What is applied Statistical Analysis

- "Applied statistical analysis refers to the use of statistical methods and techniques to analyse real-world data in order to make informed decisions, draw conclusions, and solve practical problems. It involves the application of statistical principles to various fields such as science, engineering, business, healthcare, social sciences, and more. The primary goal of applied statistical analysis is to extract meaningful information from data, uncover patterns and trends, test hypotheses, and make predictions or recommendations based on the data." – ChatGPT
- Applied statistics is the root of data analysis. The practice of applied statistics involves analysing data to help define and determine business needs. Modern workplaces are overwhelmed with big data and are looking for statisticians, data analysts, data scientists, and other professional switch applied statistics knowledge who can organize, analyse, and use data to solve real-world problems." - Michigan Tech
- Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy." - IBM

What are key aspects of Applied Statistical Analysis?

- Data Collection
- Data Cleaning and Preprocessing
- Descriptive Statistics
- Exploratory Data Analysis
- Inferential Statistics
- Regression Analysis
- Data Visualization
- Interpretation and Reporting



Gather Data

diabetes_dataset.csv

diabetes_dataset.csv | Diabetes_study.ipynb

Delimiter: ,

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5
1	59	2	32.1	101	157	93.2	38	4	4.8598
2	48	1	21.6	87	183	103.2	70	3	3.8918
3	72	2	30.5	93	156	93.6	41	4	4.6728
4	24	1	25.3	84	198	131.4	40	5	4.8903
5	50	1	23	101	192	125.4	52	4	4.2905
6	23	1	22.6	89	139	64.8	61	2	4.1897
7	36	2	22	90	160	99.6	50	3	3.9512
8	66	2	26.2	114	255	185	56	4.55	4.2485
9	60	2	32.1	83	179	119.4	42	4	4.4773
10	29	1	30	85	180	93.4	43	4	5.3845
11	22	1	18.6	97	114	57.6	46	2	3.9512
12	56	2	28	85	184	144.8	32	6	3.5835
13	53	1	23.7	92	186	109.2	62	3	4.3041

```
import pandas as pd
data = pd.read_csv('diabetes_dataset.csv')
```

Data Points and Features

```
type(data)
```

```
pandas.core.frame.DataFrame
```

```
data.shape
```

```
(442, 11)
```

10 Features in order:

- AGE age in years
- SEX sex
- BMI body mass index
- BP average blood pressure
- S1 tc, total serum cholesterol
- S2 ldl, low-density lipoproteins
- S3 hdl, high-density lipoproteins
- S4 tch, total cholesterol / HDL
- S5 ltg, possibly log of serum triglycerides level
- S6 glu, blood sugar level

Target variable:

- Y response

Cleaning Data

```
pd.isnull(data).any()
```

```
AGE    False
SEX    False
BMI    False
BP     False
S1     False
S2     False
S3     False
S4     False
S5     False
S6     False
Y      False
dtype: bool
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   AGE     442 non-null    int64
 1   SEX     442 non-null    int64
 2   BMI     442 non-null    float64
 3   BP      442 non-null    float64
 4   S1      442 non-null    int64
 5   S2      442 non-null    float64
 6   S3      442 non-null    float64
 7   S4      442 non-null    float64
 8   S5      442 non-null    float64
 9   S6      442 non-null    int64
 10  Y       442 non-null    int64
dtypes: float64(6), int64(5)
memory usage: 38.1 KB
```

Data Exploration

Data visualization also help us make sense of the data at the Exploration Stage

```
data.head()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151
1	48	1	21.6	87.0	183	103.2	70.0	3.0	3.8918	69	75
2	72	2	30.5	93.0	156	93.6	41.0	4.0	4.6728	85	141
3	24	1	25.3	84.0	198	131.4	40.0	5.0	4.8903	89	206
4	50	1	23.0	101.0	192	125.4	52.0	4.0	4.2905	80	135

```
data.tail()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
437	60	2	28.2	112.00	185	113.8	42.0	4.00	4.9836	93	178
438	47	2	24.9	75.00	225	166.0	42.0	5.00	4.4427	102	104
439	60	2	24.9	99.67	162	106.6	43.0	3.77	4.1271	95	132
440	36	1	30.0	95.00	201	125.2	42.0	4.79	5.1299	85	220
441	36	1	19.6	71.00	250	133.2	97.0	3.00	4.5951	92	57

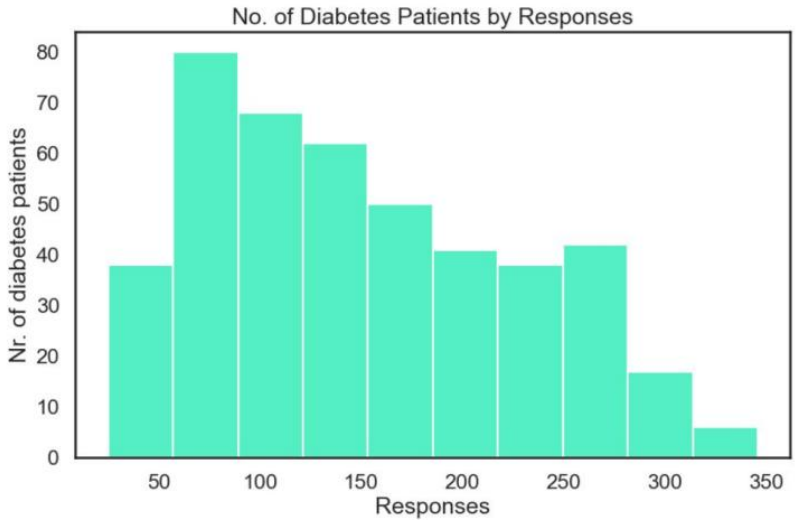
```
data.count()
```

```
AGE     442
SEX     442
BMI     442
BP      442
S1      442
S2      442
S3      442
S4      442
S5      442
S6      442
Y       442
dtype: int64
```

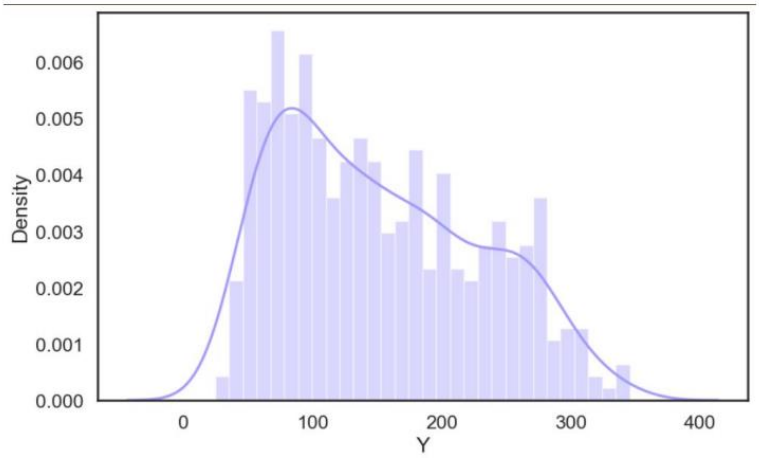
Visualizing Data

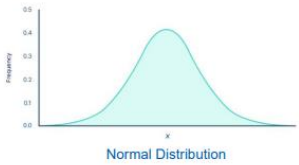
- Data visualization also help us make sense of the data at the Exploration Stage
- Used for data distribution and outliers
- Useful for identifying Normal vs Skewed Distributions

```
plt.figure(figsize=(10, 6))
plt.hist(data['Y'], ec='#ffffff', color='#55efc4')
plt.xlabel('Responses')
plt.ylabel('Nr. of diabetes patients')
plt.title('No. of Diabetes Patients by Responses')
plt.show()
```



```
plt.figure(figsize=(10, 6))
sns.distplot(data['Y'], bins=30, color='#a29bfe')
plt.show()
```

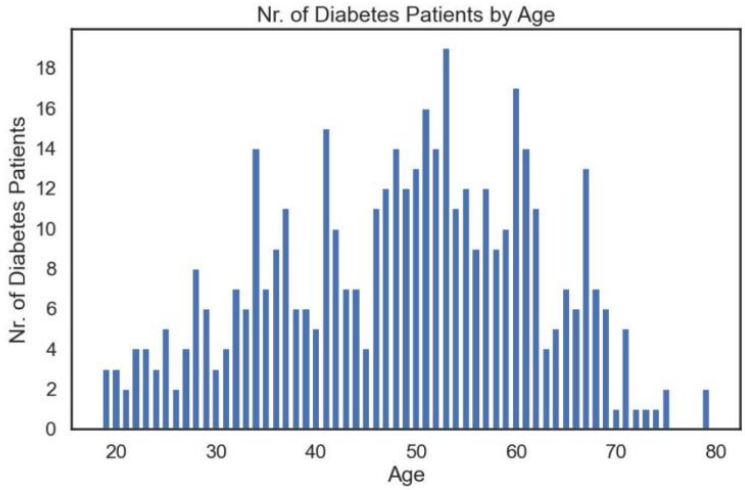




Visualising Data (continued…)

```
data['AGE'].mean()
```
48.51809954751131

```
data['AGE'].median()
```
50.0




```
frequency = data['AGE'].value_counts()

plt.figure(figsize=(10, 6))
plt.xlabel('Age')
plt.ylabel('Nr. of Diabetes Patients')
custom_y_ticks = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
plt.yticks(custom_y_ticks)
plt.title('Nr. of Diabetes Patients by Age')
plt.bar(frequency.index, height=frequency)
plt.show()
```


Descriptive Statistics

```
data['Y'].min()
```
25

```
data['Y'].max()
```
346

```
data.min()
```

AGE	19.0000
SEX	1.0000
BMI	18.0000
BP	62.0000
S1	97.0000
S2	41.6000
S3	22.0000
S4	2.0000
S5	3.2581
S6	58.0000
Y	25.0000

dtype: float64

```
data.max()
```

AGE	79.000
SEX	2.000
BMI	42.200
BP	133.000
S1	301.000
S2	242.400
S3	99.000
S4	9.090
S5	6.107
S6	124.000
Y	346.000

dtype: float64

```
data.head()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151
1	48	1	21.6	87.0	183	103.2	70.0	3.0	3.8918	69	75
2	72	2	30.5	93.0	156	93.6	41.0	4.0	4.6728	85	141
3	24	1	25.3	84.0	198	131.4	40.0	5.0	4.8903	89	206
4	50	1	23.0	101.0	192	125.4	52.0	4.0	4.2905	80	135

```
data.tail()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
437	60	2	28.2	112.00	185	113.8	42.0	4.00	4.9836	93	178
438	47	2	24.9	75.00	225	166.0	42.0	5.00	4.4427	102	104
439	60	2	24.9	99.67	162	106.6	43.0	3.77	4.1271	95	132
440	36	1	30.0	95.00	201	125.2	42.0	4.79	5.1299	85	220
441	36	1	19.6	71.00	250	133.2	97.0	3.00	4.5951	92	57

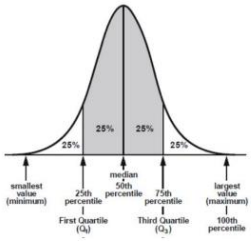
```
data.describe()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
count	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000
mean	48.518100	1.468326	26.375792	94.647014	189.140271	115.439140	49.788462	4.070249	4.641411	91.260181	152.133484
std	13.109028	0.499561	4.418122	13.831283	34.608052	30.413081	12.934202	1.290450	0.522391	11.496335	77.093005
min	19.000000	1.000000	18.000000	62.000000	97.000000	41.600000	22.000000	2.000000	3.258100	58.000000	25.000000
25%	38.250000	1.000000	23.200000	84.000000	164.250000	96.050000	40.250000	3.000000	4.276700	83.250000	87.000000
50%	50.000000	1.000000	25.700000	93.000000	186.000000	113.000000	48.000000	4.000000	4.620050	91.000000	140.500000
75%	59.000000	2.000000	29.275000	105.000000	209.750000	134.500000	57.750000	5.000000	4.997200	98.000000	211.500000
max	79.000000	2.000000	42.200000	133.000000	301.000000	242.400000	99.000000	9.090000	6.107000	124.000000	346.000000

```
data.mean()
AGE     48.518100
SEX      1.468326
BMI     26.375792
BP      94.647014
S1     189.140271
S2     115.439140
S3      49.788462
S4       4.070249
S5       4.641411
S6      91.260181
Y      152.133484
dtype: float64
```

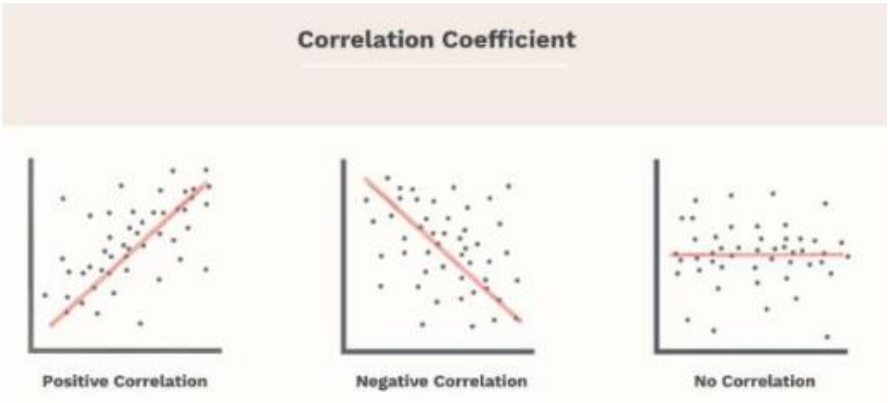
```
data.median()
AGE     50.00000
SEX      1.00000
BMI     25.70000
BP      93.00000
S1     186.00000
S2     113.00000
S3      48.00000
S4       4.00000
S5       4.62005
S6      91.00000
Y      140.50000
dtype: float64
```

- Mean: The average
- Median: The midpoint of the distribution. I.e., the middle value



Correlation

- Correlation is a statistical measure that describes the extent to which two variables change together (i.e., their relationship).
- 1 is a perfect positive correlation
- 0 means there is no correlation
- -1 is a perfect negative correlation
- Correlation (a linear relationship): is important because we want to include features that have the right strength and direction (i.e., features that are correlated with the target)
-



Correlation

```
data['Y'].corr(data['BMI'])
```

```
0.5864501344746885
```

```
data['BMI'].corr(data['AGE'])
```

```
0.18508466614655544
```

10 Features in order:

- AGE age in years
- SEX sex
- BMI body mass index
- BP average blood pressure
- S1 tc, total serum cholesterol
- S2 ldl, low-density lipoproteins
- S3 hdl, high-density lipoproteins
- S4 tch, total cholesterol / HDL
- S5 ltg, possibly log of serum triglycerides level
- S6 glu, blood sugar level

Target variable:

- Y response

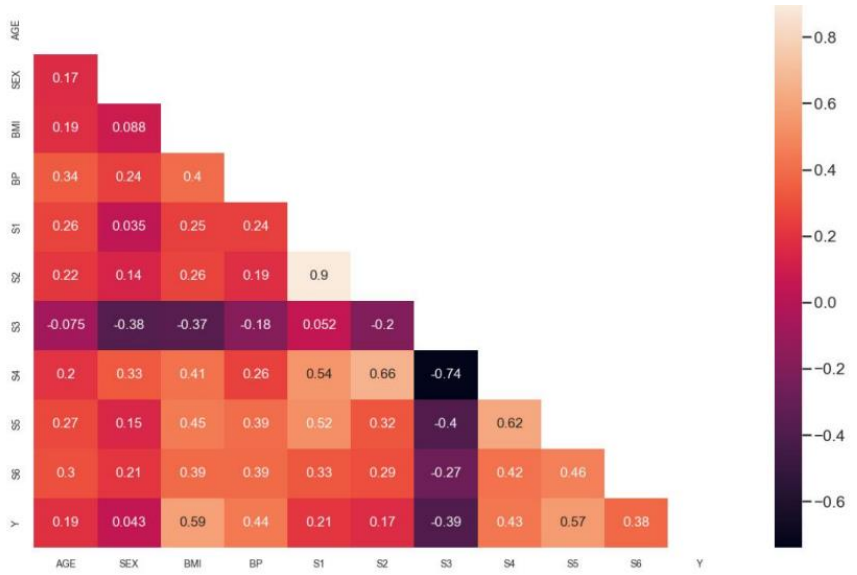
```
data.corr()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
AGE	1.000000	0.173737	0.185085	0.335428	0.260061	0.219243	-0.075181	0.203841	0.270774	0.301731	0.187889
SEX	0.173737	1.000000	0.088161	0.241010	0.035277	0.142637	-0.379090	0.332115	0.149916	0.208133	0.043062
BMI	0.185085	0.088161	1.000000	0.395411	0.249777	0.261170	-0.366811	0.413807	0.446157	0.388680	0.586450
BP	0.335428	0.241010	0.395411	1.000000	0.242464	0.185548	-0.178762	0.257650	0.393480	0.390430	0.441482
S1	0.260061	0.035277	0.249777	0.242464	1.000000	0.896663	0.051519	0.542207	0.515503	0.325717	0.212022
S2	0.219243	0.142637	0.261170	0.185548	0.896663	1.000000	-0.196455	0.659817	0.318357	0.290600	0.174054
S3	-0.075181	-0.379090	-0.366811	-0.178762	0.051519	-0.196455	1.000000	-0.738493	-0.398577	-0.273697	-0.394789
S4	0.203841	0.332115	0.413807	0.257650	0.542207	0.659817	-0.738493	1.000000	0.617859	0.417212	0.430453
S5	0.270774	0.149916	0.446157	0.393480	0.515503	0.318357	-0.398577	0.617859	1.000000	0.464669	0.565883
S6	0.301731	0.208133	0.388680	0.390430	0.325717	0.290600	-0.273697	0.417212	0.464669	1.000000	0.382483
Y	0.187889	0.043062	0.586450	0.441482	0.212022	0.174054	-0.394789	0.430453	0.565883	0.382483	1.000000


```
mask = np.zeros_like(data.corr())
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True
mask
```

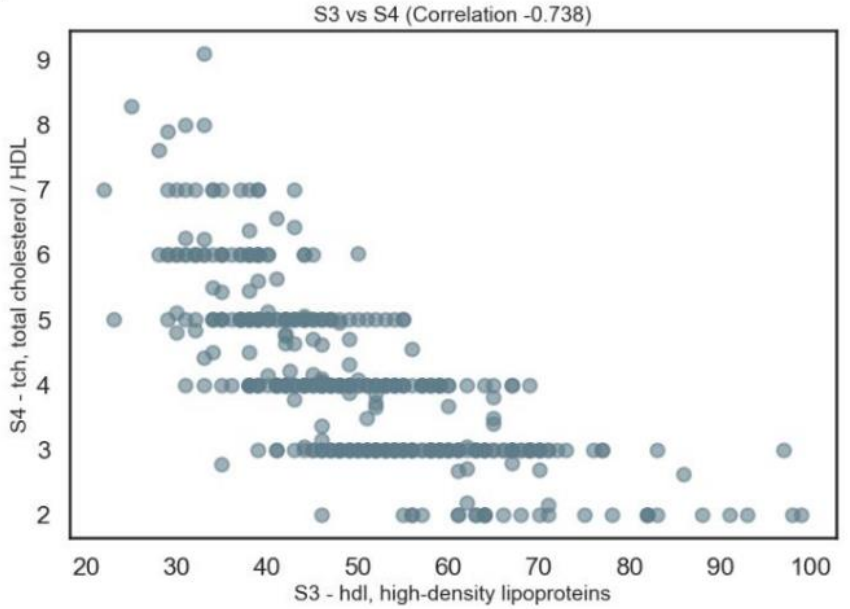
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]])
```

```
plt.figure(figsize=(16, 10))
sns.heatmap(data.corr(), mask=mask, annot=True, annot_kws={'size': 14})
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



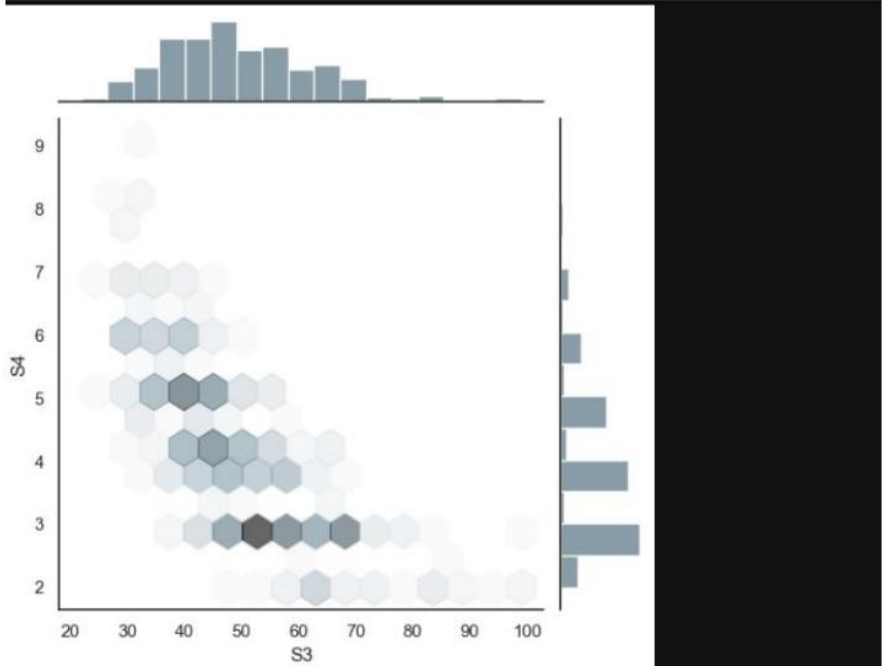
Correlation (continued…)

```
# Scatter plot between S4 and S3 Correlation
s3_s4_corr = round(data['S3'].corr(data['S4']), 3)
plt.figure(figsize=(9, 6))
plt.scatter(x=data['S3'], y=data['S4'], alpha=0.6, s=80, color='#607D8B')
plt.title(f'S3 vs S4 (Correlation {s3_s4_corr})', fontsize=14)
plt.xlabel('S3 - hdl, high-density lipoproteins', fontsize=14)
plt.ylabel('S4 - tch, total cholesterol / HDL', fontsize=14)
plt.show()
```



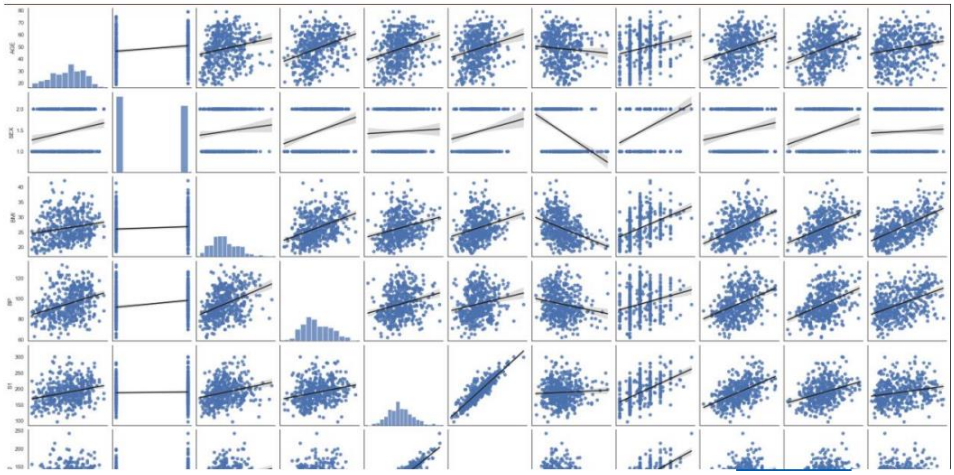
Correlation (continued…)

```
sns.set()
sns.set_context('notebook')
sns.set_style('white')
sns.jointplot(x=data['S3'], y=data['S4'], height=6, color='#607D8B', kind='hex', joint_kws={'alpha': 0.6})
plt.show()
```



Correlation (continued…)

```
%%time

sns.pairplot(data, kind='reg', plot_kws={'line_kws':{'color':'#212121'}})
plt.show()
```



Training & Test Dataset Split

```
responses = data['Y']
features = data.drop('Y', axis=1)

X_train, X_test, y_train, y_test = train_test_split(features, responses,
                                                    test_size=0.2, random_state=30)


len(X_train)/len(features)
```

```
len(X_test)/len(features)
```

```
0.20135746606334842
```

Shuffling randomizes the order of your data. ♣ For data split there is no one size fits all approach

Multivariable Linear Regression

```
regr = LinearRegression()
regr.fit(X_train, y_train)

print('Training data r-squared:', regr.score(X_train, y_train))
print('Test data r-squared:', regr.score(X_test, y_test))

print('Intercept', regr.intercept_)
pd.DataFrame(data=regr.coef_, index=X_train.columns, columns=['coef'])
```

Linear regression helps us understand how changes in one or more variables are associated with changes in another variable

Simple linear regression: between 2 variables (e.g., target (y) and feature (x)). ♣ Multivariate linear regression: between multiple variables (e.g., target (y) and features (x1, x2, x3 …xn))

To evaluate your model you use statistics such as:

♣ r-squared

♣ p-values

♣ MSE

♣ VIF

♣ BIC

```
Training data r-squared: 0.5483730165268 65
Test data r-squared: 0.41370714229360894
Intercept -10.534842392647715
```

	coef
AGE	0.002635
SEX	-0.788625
BMI	0.227139
BP	0.056038
S1	-0.087029
S2	0.071969
S3	0.048395
S4	0.268017
S5	3.932265
S6	-0.014507

Note: we were able to explain approximately 53% (r-squared) of the variance of the response with just 10 features.

♣ The performance of the model on the test data is going to be worse that that of the training data.

♣ Negative coefficients means that there is an inverse relationship between that feature and the dependent variable. You interpret coefficients in the context of your specific dataset and problem domain.

```
Training data r-squared: 0.5264783626678893
Test data r-squared: 0.4653044632644133
Intercept -368.2185304349134
```

	coef
AGE	0.029063
SEX	-23.059093
BMI	5.602677
BP	1.254404
S1	-1.334738
S2	0.955752
S3	0.588029
S4	3.158733
S5	78.918445
S6	0.226053

Data Transformations

```
data['Y'].skew()
```

```
0.44056293407014124
```

Logarithms (log) helps transform the data before you fit the linear regression line - useful if the data is skew

Other data transformations besides log transformation for linear regression include: ♣ Square Root Transformation ♣ Exponential Transformation ♣ Box-Cox Transformation ♣ Yeo-Johnson Transformation

```
y_log = np.log(data['Y'])
y_log.tail()
```

```
437    5.181784
438    4.644391
439    4.882802
440    5.393628
441    4.043051
Name: Y, dtype: float64
```

```
y_log.skew()
```

```
-0.3325670604728491
```

```
response = np.log(data['Y'])
features = data.drop(['Y'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(features, response,
                                                    test_size=0.4, random_state=20)

regr = LinearRegression()
regr.fit(X_train, y_train)

print('Training data r-squared:', regr.score(X_train, y_train))
print('Test data r-squared:', regr.score(X_test, y_test))

print('Intercept', regr.intercept_)
pd.DataFrame(data=regr.coef_, index=X_train.columns, columns=['coef'])
```

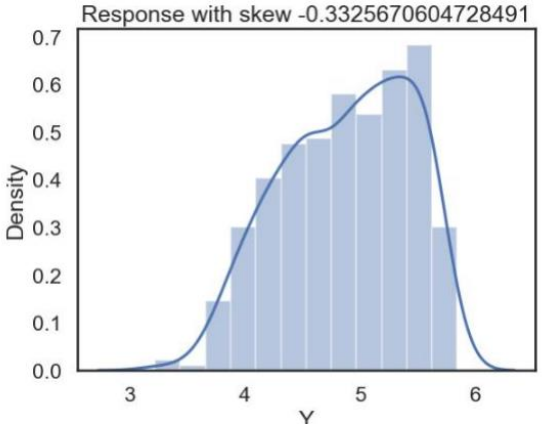
```
Training data r-squared: 0.5250952185956388
Test data r-squared: 0.3853085475235067
Intercept 1.080693882659484
```

```
response = np.sqrt(data['Y'])
features = data.drop('Y', axis=1)

X_train, X_test, y_train, y_test = train_test_split(features, response,
                                                    test_size=0.4, random_state=20)

regr = LinearRegression()
regr.fit(X_train, y_train)

print('Training data r-squared:', regr.score(X_train, y_train))
print('Test data r-squared:', regr.score(X_test, y_test))

print('Intercept', regr.intercept_)
pd.DataFrame(data=regr.coef_, index=X_train.columns, columns=['coef'])
```

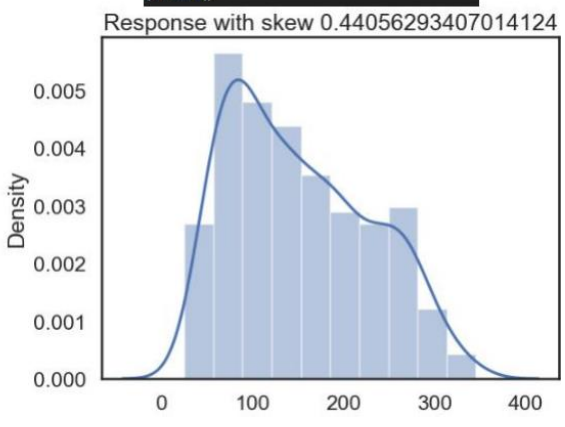
```
Training data r-squared: 0.5483730165268 65
Test data r-squared: 0.41370714229360894
Intercept -10.534842392647715
```

Data Transformation (continued…)

```
sns.distplot(y_log)
plt.title(f'Response with skew {y_log.skew()}')
plt.show()
```

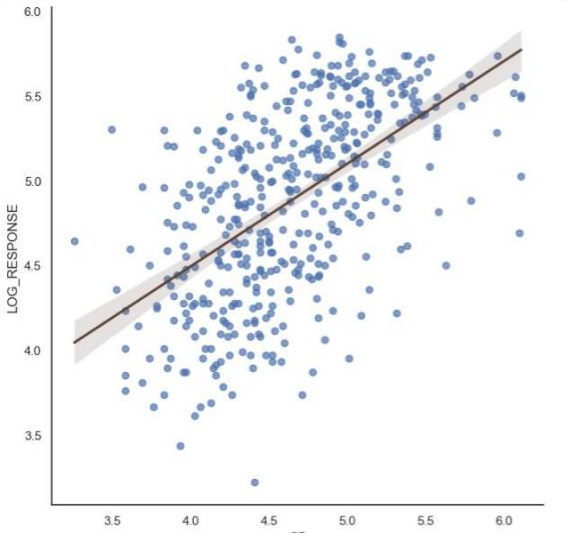



```
tmpdata = data['Y']

sns.distplot(tmpdata)
plt.title(f'Response with skew {tmpdata.skew()}')
plt.show()
```

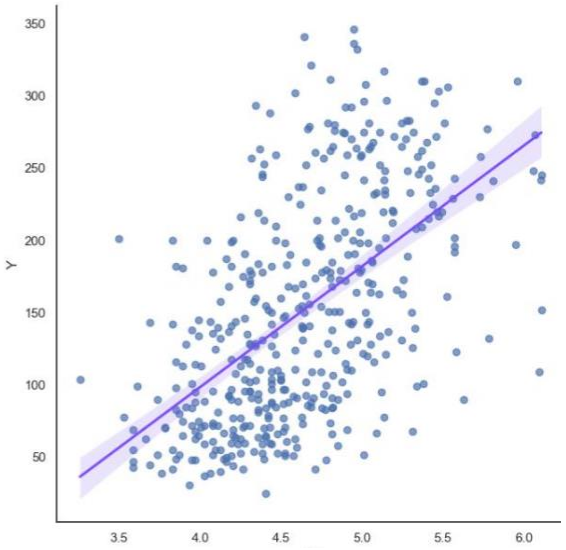


Data Transformation (continued…)

```
# after transformed Log Data
transformed_data = features
transformed_data['LOG_RESPONSE'] = y_log

sns.lmplot(x='S5', y='LOG_RESPONSE', data=transformed_data, height=7, scatter_kws={'alpha': 0.7}, line_kws={'color':'#5D4837'})
plt.show()
```




```
# before transformed Log Data
sns.lmplot(x='S5', y='Y', data=data, height=7, scatter_kws={'alpha': 0.7}, line_kws={'color':'#7
plt.show()
```



P Values & Evaluating Coefficients

```
X_incl_const = sm.add_constant(X_train)

model = sm.OLS(y_train, X_incl_const)
results = model.fit()

pd.DataFrame({'coef': results.params, 'p-value': round(results.pvalues, 3)})
```

	coef	p-value
const	-368.218530	0.000
AGE	0.029063	0.905
SEX	-23.059093	0.000
BMI	5.602677	0.000
BP	1.254404	0.000
S1	-1.334738	0.031
S2	0.955752	0.096
S3	0.588029	0.487
S4	3.158733	0.624
S5	78.918445	0.000
S6	0.226053	0.457

P ≤ 0,05 is considered statistically significant.

Nb: It is preferred to have a more simple model to explain things, therefore, simpler models are considered better than complex models.

♣ So, to simplify your model you can drop features that are insignificant. But, this should not be done “willy- nilly”. Because, even a feature with a high/bad p-value can add value to the model as a whole. To drop features requires some contemplation

Bayesian Information Criteria (BIC)

- BIC is a model selection criterion for a finite list of models
- A way to measure complexity between models
- The lower the BIC value the better the model (a lower value is better)

```
print('BIC is', results.bic)
print('r-squared is', results.rsquared)
pd.DataFrame({'coef': results.params, 'p-value': round(results.pvalues, 3)})
```

```
BIC is 2900.7543929935687
```

```
print('BIC is', results.bic)
pd.DataFrame({'coef': results.params, 'p-value': round(results.pvalues, 3)})
```

```
BIC is 305.1614591304148
```

```
print('BIC is', results.bic)

pd.DataFrame({'coef': results.params, 'p-value': round(results.pvalues, 3)})
```

```
BIC is 1214.431162240603
```

Variance Inflation Factor (VIF)

- We are testing for Multicollinearity
- VIF is a measure of collinearity amongst the features within a regression model
- A high VIF value indicates that a feature is highly correlated with other features, potentially leading to unstable coefficient estimates

```
X_incl_const.columns
```

```
Index(['const', 'AGE', 'SEX', 'BMI', 'BP', 'S1', 'S2', 'S3', 'S4', 'S5', 'S6'], dtype='object')
```

```
for i in range(X_incl_const.shape[1]):
    print(variance_inflation_factor(exog=X_incl_const, exog_idx=i))
```

```
671.8771190666284
1.1487198884621623
1.2553595575721868
1.4774401394787282
1.4332140437255596
57.98699505658825
39.391753886328544
14.313441346908945
9.783714465306861
10.571305534829273
1.4323006290342313
```

- VIF = 1: A VIF of 1 indicates no multicollinearity. It means that the feature is not correlated with any other feature in the model.
- VIF < 5: A VIF below 5 generally suggests low to moderate multicollinearity. In this range, you typically don't need to be overly concerned about multicollinearity
- VIF >= 5: A VIF of 5 or greater suggests high multicollinearity. This indicates that the feature is highly correlated with other features in the model, and it may be problematic
- VIF > 10: A VIF greater than 10 is a strong indication of severe multicollinearity. In such cases, it's essential to address multicollinearity, such as by removing one or more correlated features, through feature selection, or by applying dimensionality reduction techniques.

Mean Square Error (MSE)

- MSE calculates how close the regression line is to the data points by considering the predicted value of the observation and eliminates the arbitrariness associated with the residual sum of the squares
- MSE equation measures the average squared error of our predictions between the actual output and the model's prediction
- The lower the MSE value the lower the variance of error

```
# Mean Squared Error & R-Squared
full_normal_mse = round(results.mse_resid, 3)
full_normal_rsquared = round(results.rsquared, 3)
```

```
pd.DataFrame({'R-Squared': [full_normal_rsquared],
              'MSE': [full_normal_mse],
              'RMSE': np.sqrt([full_normal_mse])},
             index=['Full Normal Response Model'])
```

	R-Squared	MSE	RMSE
Full Normal Response Model	0.475	3217.196	56.720331

```
pd.DataFrame({'R-Squared': [reduced_log_rsquared, full_normal_rsquared, omitted_var_rsquared],
              'MSE': [reduced_log_mse, full_normal_mse, omitted_var_mse],
              'RMSE': np.sqrt([reduced_log_mse, full_normal_mse, omitted_var_mse])},
             index=['Reduced Log Model', 'Full Feature Set Model', 'Reduced Log with Omitted Features Model' ])
```

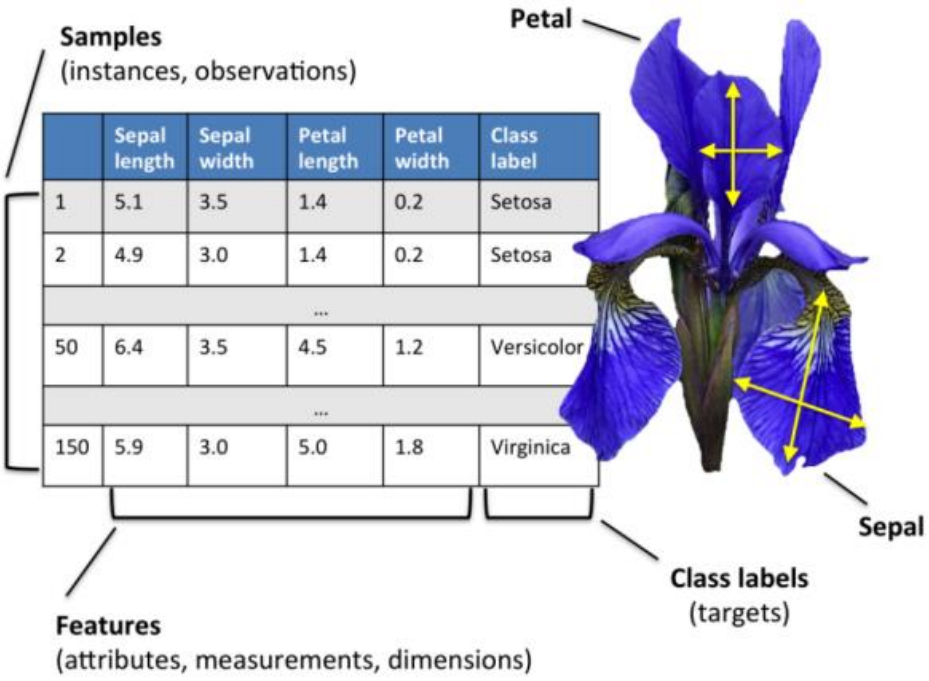
	R-Squared	MSE	RMSE
Reduced Log Model	0.527	0.039	0.197484
Full Feature Set Model	0.564	0.507	0.712039
Reduced Log with Omitted Features Model	0.527	0.038	0.194936

We use Residuals to check if model and our assumptions are valid

Residuals should be normally distributed with no identifiable pattern (read up)

Using the Iris Dataset with K-means

The Iris dataset is a classic dataset in machine learning, ideal for clustering since it contains numeric features and well-defined clusters (species of iris flowers).



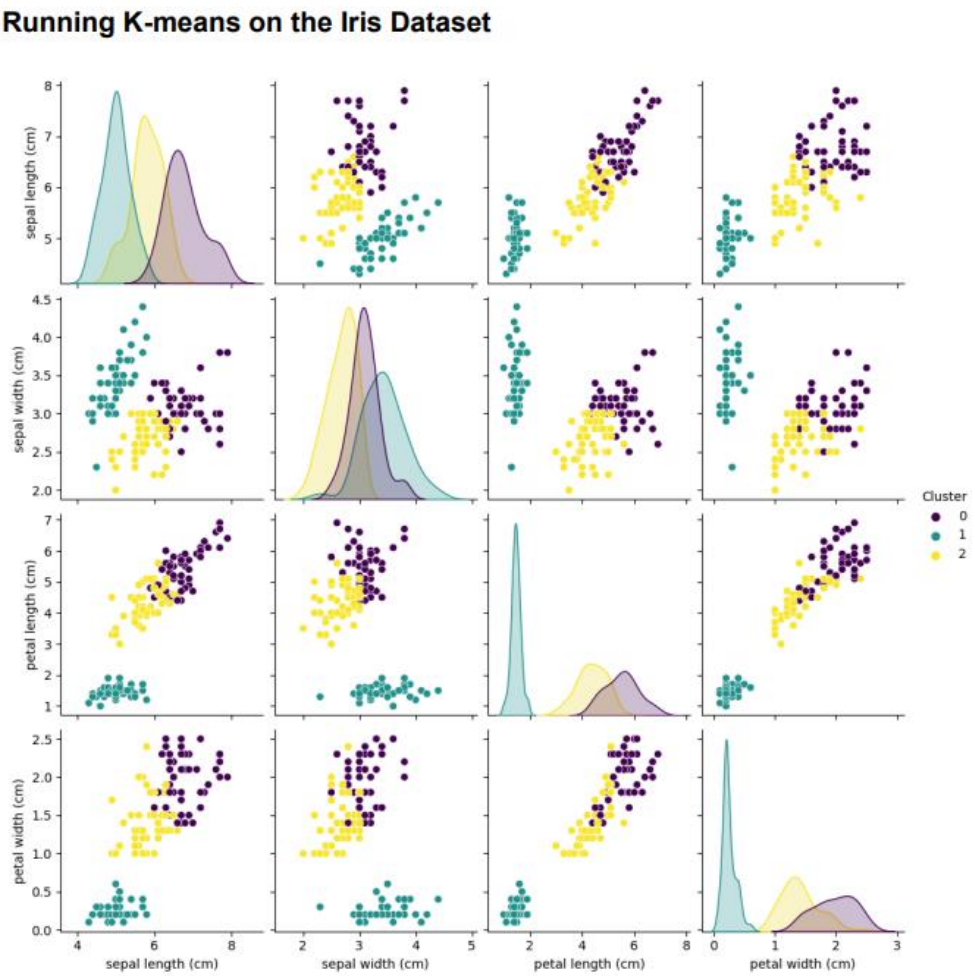
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt
# Load the Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
# Standardize the data
scaler = StandardScaler()
iris_scaled = scaler.fit_transform(iris_df)
# Run K-means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
```

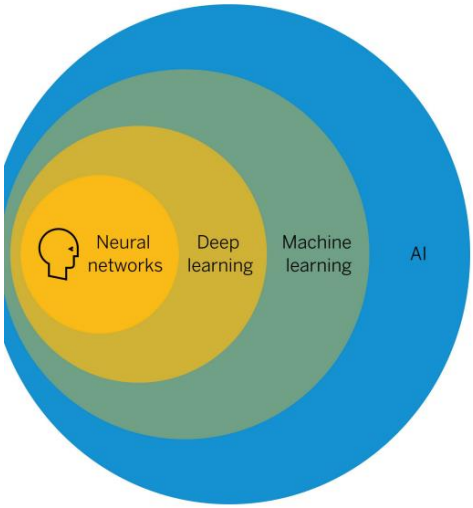
```
iris_df['Cluster'] = kmeans.fit_predict(iris_scaled)

# Visualize the clusters

sns.pairplot(iris_df, hue='Cluster', palette='viridis')

plt.show()

# Inspect cluster centers

print("Cluster Centers:")

print(kmeans.cluster_centers_)
```

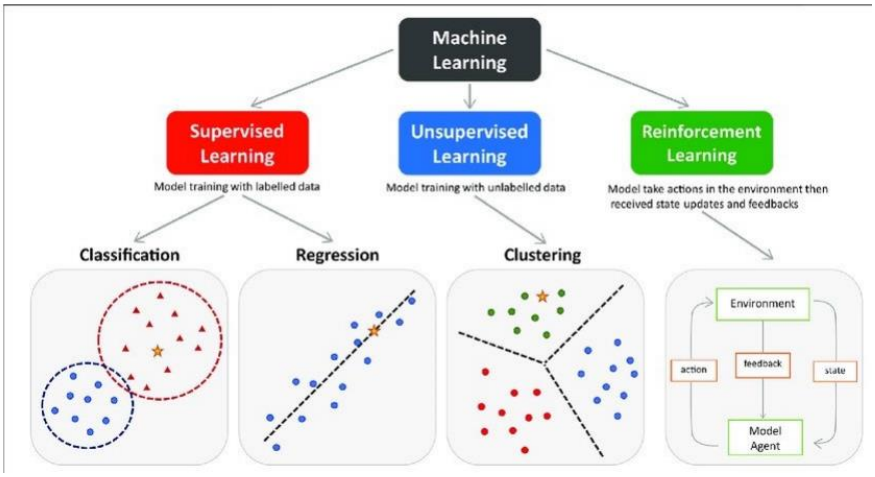
Explanation

- **Iris Dataset**: The Iris dataset is loaded using sklearn.datasets.load_iris(). It contains four features (sepal length, sepal width, petal length, and petal width) and is often used to demonstrate clustering.
- **Scaling**: The features are standardised to ensure each feature contributes equally to the clustering.
- **K-means Clustering**: K-means is applied with 3 clusters (since there are three species of iris in the dataset).
- **Visualisation**: A pairplot is used to visualise the clusters across different feature combinations.
- **Cluster Centres**: The centres of the clusters are printed for analysis.



What is machine Learning





What are key aspects of Machine Learning

- Data
- Features
- Algorithms
- Learning types (Supervised, Unsupervised, Reinforcement, etc...)
- Model Training
- Evaluation Metrics
- Overfitting and Underfitting
- Model Selection
- Deployment

Natural Language Processing and Computational Lexicography

What is Natural Language Processing (NLP)?

Natural Language Processing (NLP) is a branch of artificial intelligence that enables machines to understand, interpret, and respond to human language in a valuable way. It involves the interaction between computers and humans using natural language, making it crucial for applications like voice assistants, chatbots, translation tools, and sentiment analysis. The importance of NLP lies in its ability to bridge the gap between human communication and machine understanding, allowing for more intuitive interactions with AI systems in various industries such as healthcare, customer service, and education

Key NLP Tasks:

Tokenization

- **Definition**: Tokenization is the process of breaking down text into smaller units such as words, phrases, or sentences. These tokens are the building blocks for further analysis and processing
- **Importance**: Tokenization is essential because it helps in structuring unstructured text for various NLP tasks, such as sentiment analysis or machine translation.

Part-of-Speech Tagging (POS Tagging)

- **Definition**: This process involves assigning grammatical tags (e.g., nouns, verbs, adjectives) to each word in a sentence
- **Importance**: POS tagging helps in understanding the syntactic structure of a sentence, which is critical for tasks like parsing, machine translation, and text generation

Named Entity Recognition (NER)

- **Definition**: NER identifies and classifies entities in text, such as names of people, organizations, locations, and dates
- **Importance**: NER is used in tasks like information extraction, question answering, and knowledge graph construction, helping machines identify and understand real-world entities in text

Text Classification

- **Definition**: Text classification involves categorizing text into predefined labels or categories, such as spam detection, topic classification, or sentiment categorization
- **Importance**: This task is vital for filtering information and automating processes, such as sorting emails or analysing customer feedback

Sentiment Analysis

- **Definition**: Sentiment analysis detects emotions or opinions within text, determining whether the sentiment expressed is positive, negative, or neutral.
- **Importance**: Sentiment analysis is widely used in areas like social media monitoring, customer reviews, and market research, where understanding public opinion is crucial

These NLP tasks are foundational in modern AI applications, enabling machines to process and respond to human language more naturally and effectively

Computational Lexicography

Computational Lexicography is the study and development of electronic dictionaries and lexicons using computational methods. It involves leveraging algorithms and digital tools to create, organize, and manage large-scale dictionaries and lexical resources. These lexicons are then used in various natural language processing (NLP) applications to enhance machine understanding of language.

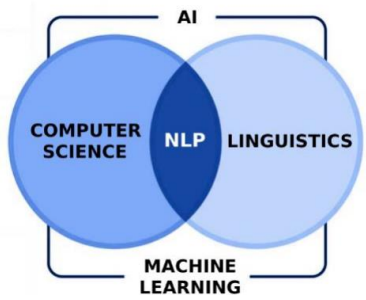
Importance in NLP

Dictionaries and lexicons are foundational resources in NLP as they help machines comprehend word meanings, relationships, synonyms, and grammatical structures.

They are essential for tasks such as machine translation, text mining, and semantic analysis, where accurate word definitions and contexts are critical for language understanding.

Computational lexicography enables the creation of rich, structured data that can be processed by algorithms, making it easier to develop more efficient and accurate NLP systems.

What are key aspects of NLP?

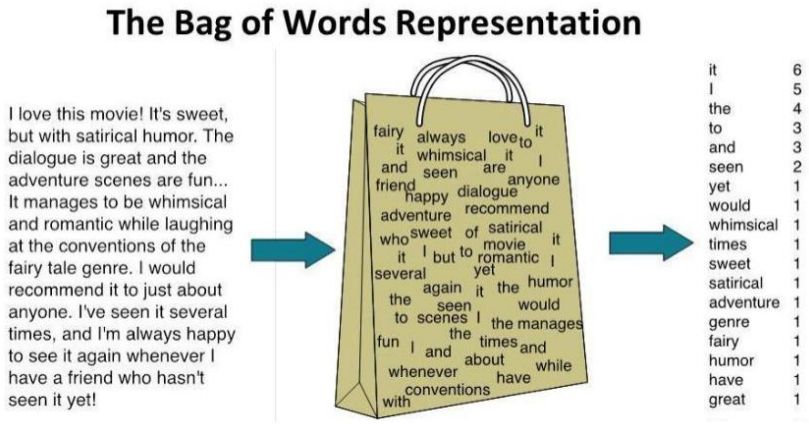


Key Techniques in NLP

Bag of Words (BoW)

Definition: Bag of Words is a simple and widely used technique in NLP that represents a text as an unordered collection of words, disregarding grammar and word order but keeping track of word frequencies. In this model, a document is represented as a vector, where each word corresponds to a dimension, and the value represents the frequency of the word in that document.

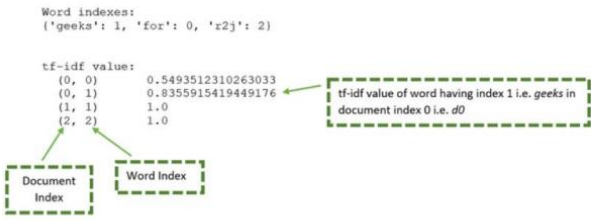
Use Case: BoW is commonly used in text classification tasks, such as spam detection or sentiment analysis, where understanding the presence or absence of certain words is more important than their order



TF-IDF (Term Frequency-Inverse Document Frequency)

Definition: TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection or corpus. It combines two factors: term frequency (TF), which measures how often a word appears in a document, and inverse document frequency (IDF), which measures how common or rare the word is across the entire corpus. A higher TF-IDF score means a word is more significant in distinguishing the document from others.

Use Case: TF-IDF is useful for information retrieval, ranking the importance of words in documents, and document similarity tasks, such as recommending articles based on content.



From the above image the below table can be generated:

Document	Word	Document Index	Word Index	tf-idf value
d0	for	0	0	0.549
d0	geeks	0	1	0.8355
d1	geeks	1	1	1.000
d2	r2j	2	2	1.000

BM25

Definition: BM25 is an advanced ranking function used in information retrieval systems like search engines. It builds on the TF-IDF model but improves the ranking of documents by considering term saturation (diminishing returns as terms are repeated) and document length normalization. BM25 ranks documents based on the relevance of their content to a query.

Use Case: BM25 is widely used in search engines to assess the relevance of documents and provide more accurate search

Word Embeddings (e.g., Word2Vec, GloVe)

Definition: Word embeddings are dense vector representations of words in a continuous vector space, capturing semantic relationships between words. Word2Vec and GloVe are popular algorithms for generating word embeddings.

Unlike Bag of Words or TF-IDF, word embeddings take into account the context in which words appear, allowing them to capture more nuanced relationships, such as "king" being closer to "queen" than to "man."

Use Case: Word embeddings are used in tasks like machine translation, question answering, and text similarity, where understanding the semantic meaning of words is essential

These techniques provide foundational methods for processing and analyzing text in NLP, enabling machines to better understand and generate human language

Role of Lexicons in NLP

Lexicons are essential resources in Natural Language Processing (NLP), offering predefined lists of words along with relevant linguistic properties, such as their meanings, sentiment scores, and part-of-speech tags.

Lexicons serve as foundational tools for analyzing and understanding text in tasks like machine translation, text classification, and sentiment analysis.

They provide structured information that helps algorithms interpret the meaning and emotional tone of words, which is crucial for accurately processing language in tasks like text mining and semantic analysis.

Lexicon-based approaches are especially valuable for tasks that require interpretability, as the lexicons provide explicit mappings between words and their meanings.

Sentiment Analysis Lexicons

SentiWordNet

○ Definition: SentiWordNet is an extension of WordNet, assigning sentiment scores (positive, negative, and neutral) to each word. It is widely used in sentiment analysis tasks for extracting the emotional tone of words in various contexts.

○ Use Case: This lexicon is commonly used for applications requiring fine-grained sentiment detection, such as product reviews or movie rating analysis.

VADER (Valence Aware Dictionary and sEntiment Reasoner)

● Definition: VADER is a lexicon and rule-based model designed to handle sentiment analysis, particularly suited for social media content due to its ability to handle informal language, emoticons, and abbreviations.

● Use Case: VADER is often used for real-time sentiment analysis in platforms like Twitter or Facebook, where informal language prevails, and quick assessment of public sentiment is necessary.

AFINN-111

● Definition: AFINN-111 is a lexicon containing words rated for valence on a scale from negative to positive. It is primarily used for binary or multi-class sentiment classification tasks.

● Use Case: It is commonly applied in sentiment analysis tasks for customer feedback or survey responses where simplicity and quick evaluation of sentiment are essential.

Lexicon-Based vs. Machine Learning Approaches

Lexicon-Based Approaches

● **Advantages**: These methods rely on predefined word lists, making them easy to implement and interpret. They are particularly effective in smaller datasets or applications requiring transparency and traceability, as they provide clear mappings between words and sentiments.

● Limitations: Lexicon-based methods may lack flexibility when encountering out-of-vocabulary words or complex contexts. They often struggle with detecting nuanced sentiments or slang not covered by the lexicon.

Machine Learning Approaches

● **Advantages**: ML models are more flexible and can learn patterns from data, making them more adaptable to complex contexts, idiomatic expressions, and evolving language use. They generally perform better in large-scale datasets and can automatically capture more nuanced sentiments.

● **Limitations**: Machine learning models can be less interpretable, requiring more resources for training and validation.

Additionally, they often require extensive labeled data and may suffer from overfitting or bias if not properly managed.

In summary, while lexicon-based methods offer simplicity and interpretability, machine learning models provide greater flexibility and accuracy, making the choice between the two dependents son the specific needs and constraints of the NLP task

Text Preprocessing

Text preprocessing is a critical step in Natural Language Processing (NLP), transforming raw text into a format that can be easily analysed by algorithms. It involves various techniques to clean, structure, and prepare text data before it can be used for tasks such as text classification, sentiment analysis, or machine translation.

Tokenization:

● **Definition**: Tokenization refers to the process of splitting text into smaller units, known as tokens. These tokens can be words, subwords, or even characters, depending on the application. For example, a sentence like "Natural Language Processing is fascinating!" could be tokenized into individual words:

["Natural", "Language", "Processing", "is", "fascinating", "!"].

● **Importance**: Tokenization helps break down large chunks of text into manageable pieces, making it easier for NLP models to process and analyze language at a finer level, such as word-level or sentence-level analysis.

Stop Words Removal:

● **Definition**: Stop words are common words that usually carry little meaning in text analysis (e.g., "and," "the," "is," "in"). Removing these words reduces the noise in the data and helps focus the analysis on the more meaningful words that convey key information.

● **Importance**: By removing stop words, algorithms can improve performance in tasks such as text classification or sentiment analysis because they no longer waste resources on processing common but uninformative words. However, stop word removal should be done carefully, as some contexts may require these words.

Handling Polysemy (Polysemy Dilemma)

● Definition: Polysemy refers to the phenomenon where a single word has multiple meanings. For example, the word "bank" can mean a financial institution or the side of a river. Handling polysemy is essential for accurately interpreting text because the correct meaning depends on the context in which the word appears.

● Solutions: Techniques like Word Sense Disambiguation (WSD) are employed to resolve polysemy by using the surrounding words (context) to infer the correct meaning. Machine learning models like Word2Vec can also help handle polysemy by learning different word embeddings for the various meanings of a word based on context.

These preprocessing techniques ensure that raw text is transformed into a structured format, allowing machine learning algorithms to work more efficiently and produce more accurate results.

NLP and Machine Translation

Machine translation (MT) is a key application of Natural Language Processing (NLP), aiming to automatically translate text or speech from one language to another. Over the years, several methodologies have been developed for text translation, each with its own strengths and limitations.

1. Text Translation Methods:

○ **Rule-based Systems**: Early machine translation systems relied on handcrafted linguistic rules and bilingual dictionaries. These systems translated text by applying syntactic, semantic, and grammatical rules of the source and target languages. While they provided high-quality translations in certain cases, they were limited by their dependency on predefined rules, which made them difficult to scale across diverse languages and contexts.

Statistical Machine Translation (SMT): SMT emerged as a more data-driven approach, where translation models were built using large bilingual corpora. Instead of relying on rules, SMT used probabilities to predict the most likely translation based on patterns observed in parallel texts. Phrases or words in one language were matched to their translations in another language, and the best translations were selected based on statistical models. However, SMT struggled with complex sentence structures and context beyond short phrases

Neural Machine Translation (NMT): NMT represents the state of the art in machine translation. It uses deep learning models, particularly Recurrent Neural Networks (RNNs) or Transformer models, to encode entire sentences or paragraphs into high-dimensional vector representations. These vectors capture the meaning of the input text, allowing the model to generate more accurate and contextually appropriate translations. NMT has shown remarkable improvements in fluency and coherence compared to SMT, particularly with the rise of the Transformer-based architecture like Google's BERT and OpenAI's GPT models.

Challenges in Machine Translation:

● Translating Polysemic Words: One of the major challenges in machine translation is handling polysemic words—words with multiple meanings depending on the context. For instance, the English word "bark" can mean the sound a dog makes or the outer layer of a tree. Choosing the correct meaning in translation is essential to preserving the meaning of the sentence. NMT

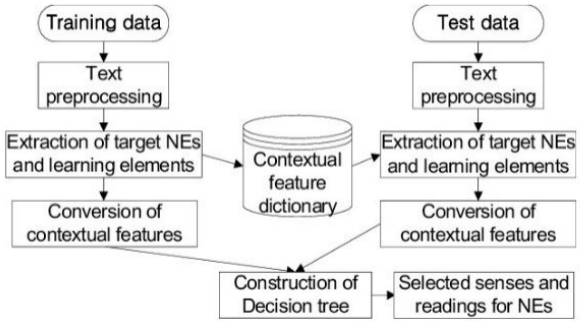
models mitigate this issue by learning contextual representations of words, but even advanced systems sometimes struggle with ambiguous terms.

● Maintaining Meaning Across Languages: Maintaining meaning during translation can be complex, especially for languages with different grammatical structures, idiomatic expressions, or cultural references. Some languages, for example, use more context-specific information than others. For example, translating gender-neutral sentences from English into gendered languages like French or Spanish requires additional contextual understanding to maintain meaning and avoid errors.

Modern machine translation has evolved from rule-based systems to powerful NMT techniques. Despite the significant advancements, challenges such as polysemy and preserving cultural meaning across languages remain ongoing areas of research in NLP

Linguistics:

> In linguistics, a *corpus* is a collection of linguistic data (usually contained in a computer database) used for research, scholarship, and teaching. Also called a *text corpus*. Plural: *corpora*.



Tools and Libraries for NLP and Lexicography

● Natural Language Toolkit (NLTK): Popular Python library for building NLP applications.

● SpaCy: Another library used for faster NLP processing.

● Stanford NLP: A suite of NLP tools for text analysis.

Computational Lexicography in Practice

1. Building and Maintaining Lexicons:

Computational lexicography involves the creation of electronic dictionaries or lexicons using **corpus data**—large collections of text that reflect real-world language usage. The process starts with collecting extensive text corpora, which are then analyzed to identify word frequencies, meanings, usage patterns, and contexts. The goal is to build lexicons that are accurate, comprehensive, and representative of how language is used. Techniques like corpus linguistics are often employed to extract lexical information such as word senses, collocations, and grammatical behavior, making the lexicons useful for tasks like machine translation, speech recognition, and sentiment analysis.

Once a lexicon is created, it needs to be continually maintained. Words evolve over time, new terms emerge (e.g., from technology or pop culture), and meanings shift, making regular updates necessary. Corpus-based approaches allow lexicographers to detect these changes, helping to keep the lexicon relevant and up-to-date. Additionally, the maintenance process often involves adding multilingual support, integrating synonyms and antonyms, and enriching the entries with semantic annotations, such as word senses, synonyms, and usage notes.

2. Annotating and Enriching Dictionaries with Semantic Information:

To enhance the utility of lexicons in computational tasks, lexicographers annotate dictionaries with semantic information. This includes assigning word senses, labelling parts of speech, and identifying semantic relations like synonymy, antonymy, and hypernymy. For instance, resources like WordNet are built around semantic networks, where words are connected through relationships that provide context and depth to their meanings. This enrichment allows for better natural language understanding (NLU) by enabling more nuanced interpretations of text, crucial for tasks like semantic search, question answering, and sentiment analysis.

3. Use of Crowdsourcing and AI to Build Large-Scale Lexicons:

Crowdsourcing and AI are becoming increasingly essential in the development of large-scale lexicons. Platforms like Wiktionary and Wordnik have utilized crowdsourcing to expand their entries by allowing users to contribute definitions, examples, and usage notes. This democratization of lexicography enables rapid growth in the lexicon, especially for emerging words and regional dialects. AI and machine learning techniques are also being used to automatically extract lexical data from corpora and web sources, significantly accelerating the lexicon-building process. AI-driven models can learn patterns in language, such as common collocations or context-dependent meanings, and help scale up the creation of lexicons for low-resource languages or specialized domains (e.g., medical terminology).

These innovations make it possible to develop more expansive and contextually rich lexicons, supporting a wide range of NLP applications and improving the quality of language-related tasks like machine translation and text classification.

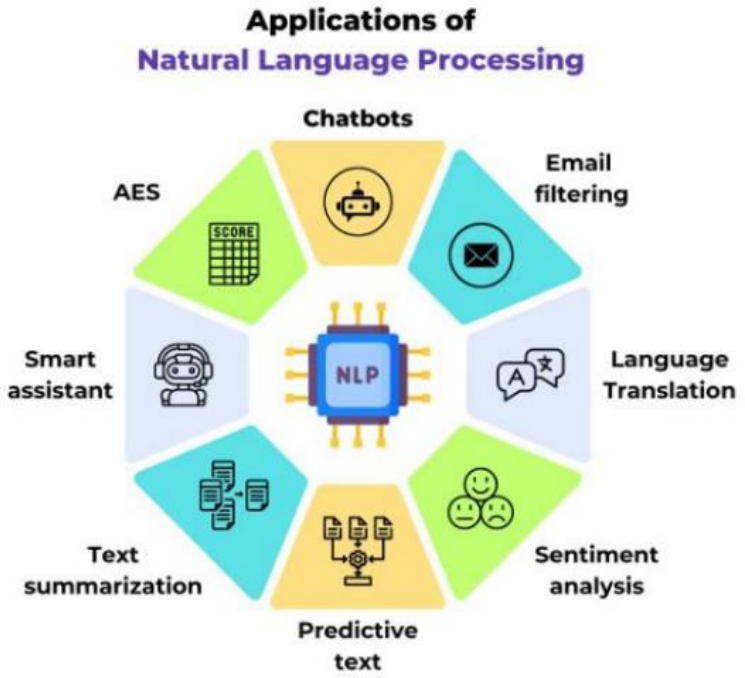
Applications of NLP and Computational Lexicography

● Search Engines: How computational lexicons improve search accuracy.

● Chatbots and Virtual Assistants: NLP-based interaction using lexicons for understanding.

● Machine Translation: Role of lexicons and syntactic parsing in translation systems.

● Text Summarization: Reducing large texts into concise summaries using NLP techniques



Challenges and Future Directions

● Ambiguity in Language: Handling polysemy, homonymy, and context sensitivity.

● Multilingual Lexicons: Building resources that support multiple languages effectively.

● Explainable NLP: Making NLP systems more interpretable and transparent, especially when using lexicons.

The evaluation of computational lexicography models involves assessing their accuracy, coverage, consistency, and usefulness for various linguistic

and NLP tasks. Below are key methods used to evaluate these models:

1. Lexicon Coverage:

Coverage measures how well the lexicon includes the vocabulary used in a specific corpus or domain. A high-coverage lexicon should contain most of the words and expressions found in the target corpus, including rare or domain-specific terms. Evaluators often compare the lexicon to reference corpora or test sets to calculate the percentage of words included.

2. Precision and Recall:

Like in machine learning, precision and recall are used to evaluate the quality of lexical entries. Precision measures the correctness of the entries (i.e., how many of the words in the lexicon are accurate), while recall measures how many of the actual words from the target dataset are included in the lexicon. F1 score, the harmonic mean of precision and recall, is often used for balanced evaluation.

3. Sense Disambiguation and Accuracy:

For lexicons that include word senses or polysemy (multiple meanings of a word), it is crucial to evaluate how well the model assigns the correct meaning based on context. Models are evaluated by comparing their word sense assignments to a gold-standard dataset where human annotators have pre-identified the correct senses. Tools like Word Sense Disambiguation (WSD) systems are commonly used to benchmark accuracy.

4. Consistency and Redundancy:

Lexicons are evaluated for internal consistency, ensuring that similar words or concepts are treated similarly throughout the lexicon. Redundancy checks are also carried out to avoid duplication of entries or conflicting meanings, which can degrade the quality of NLP tasks like machine translation and sentiment analysis.

5. Application-Based Evaluation:

A common method is to test lexicon-based models within specific NLP tasks such as machine translation, sentiment analysis, or named entity recognition (NER). Evaluators measure the performance improvements provided by the lexicon compared to other models or against baseline methods without lexicons. For example, how well the lexicon enhances machine translation or sentiment analysis can be quantified by BLEU scores (for translation) or accuracy in sentiment classification.

Benchmarking against Standard Lexical Resources

Computational lexicons can be evaluated by comparing them with established resources such as WordNet, Oxford English Dictionary, or BabelNet. Metrics such as overlap in vocabulary, semantic relations, and coverage of word senses provide insights into the model's performance

Computational lexicography models are evaluated both quantitatively (using statistical metrics) and qualitatively (through human evaluation and usability testing in NLP applications). This combined approach ensures that lexicons are both linguistically sound and practically useful in a wide range of AI-driven language tasks.

VADER

Input	neg	neu	pos	compound
"This computer is a good deal."	0	0.58	0.42	0.44
"This computer is a very good deal."	0	0.61	0.39	0.49
"This computer is a very good deal!!"	0	0.57	0.43	0.58
This computer is a very good deal!! :-)"	0	0.44	0.56	0.74
This computer is a VERY good deal!! :-)"	0	0.393	0.61	0.82

```
lexique = {
    "amazing": 2.0,
    "good": 1.5,
    "okay": 0.0,
    "bad": -1.5,
    "terrible": -2.0
}

def analyse_sentiment(text):
    words = text.lower().split()

    score = sum(lexique.get(word, 0) for word in words)
    return score

text = "The movie was amazing and the plot was good"
print(analyse_sentiment(text))
# Output: 3.5
```

of Engineering, vironment and ıtion Technology

Lexicon

Sentence	Positive score	Negative score	Binary prediction
I can play chess	+1	−1	+1
I can play chess!!!	+2	−1	+1
I like to read science fictions	+2	−1	+1
I do not like to read science fictions	+1	−1	+1
I left early because the film was boring	+1	−2	−1
I hate you	+1	−4	−1
I really love you, but dislike your cold sister	+4	−3	+1

Explainable Artificial Intelligence (XAI) & Large Language Models (LLMs)

Explainable AI (XAI) refers to a set of techniques and methods that make the decisions, results, and outputs of AI or ML models interpretable and understandable by humans.

● The rise of complex AI/ML models (black-box), especially deep learning, has led to the need for transparency and trust.

● XAI addresses the "black-box" problem by providing insights into how models make decisions.

● XAI addresses the "black-box" problem by providing insights into how models make decisions.

Key Characteristics of Black-Box Models

A black-box model refers to an AI or machine learning model whose internal workings are not easily interpretable or transparent to users.

In such models, we can observe the inputs/datasets and outputs/results, but we cannot easily understand how the model processes the inputs to produce the outputs.

These models are often complex, involving numerous layers of computations and parameters, making it challenging to explain why they make specific predictions or decisions.

Examples of Black-Box Models

● Deep Neural Networks (DNNs): DNNs consist of many layers, and their internal connections make it nearly impossible for a human to follow the logic behind individual decisions.

● Ensemble Models: Models like random forests and gradient boosting combine multiple decision trees, making the overall model's decision process hard to understand, even though individual trees may be interpretable.

● Support Vector Machines (SVMs): SVMs are hard to interpret when using complex kernel functions to transform data into high-dimensional spaces

Why Black-Box Models are a Concern?

● Trust: Users may hesitate to trust a system they cannot understand, especially in critical applications like medical diagnosis, finance, or legal decisions.

● Accountability: In cases of errors or biases, it’s difficult to determine what caused the incorrect outcome or who is responsible.

● Ethical and Legal Issues: Regulations, such as GDPR, require transparency in automated decision-making systems, and black-box models often do not comply with these transparency requirements.

To address these challenges, Explainable AI (XAI) techniques are used to make black-box models more interpretable by providing explanations of how decisions are made. Techniques like LIME, SHAP, and Attention Visualizations allow users to gain insights into the model's decision process without fully exposing its complexity.

Why XAI is Crucial

● Trust and Accountability: In fields like healthcare, finance, and law, users need to understand AI decisions to trust them and ensure ethical usage.

● Debugging and Optimization: Helps data scientists and developers identify weaknesses or biases in models, improving their performance.

● Regulatory Compliance: Compliance with regulations like GDPR (General Data Protection Regulation) which require explanations for automated decisions

XAI Techniques

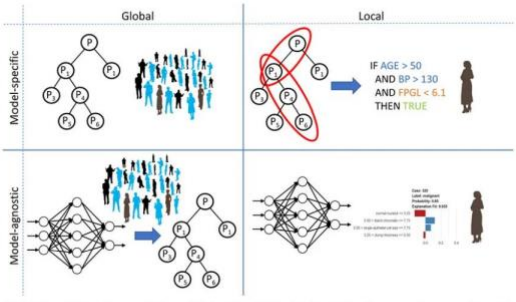
● Model-Specific vs Model-Agnostic

○ Model-Specific: Methods designed for specific types of models (e.g., decision trees, linear models) which are naturally interpretable.

○ Model-Agnostic: Techniques applicable to any machine learning model, especially black-box models like neural networks and random forests.

Said so, here we are going to focus on Post-Hoc Techniques, i.e. explanation methods that work on top of complex black-box methods. *So that anyone can have fun with the ML model he prefers.*

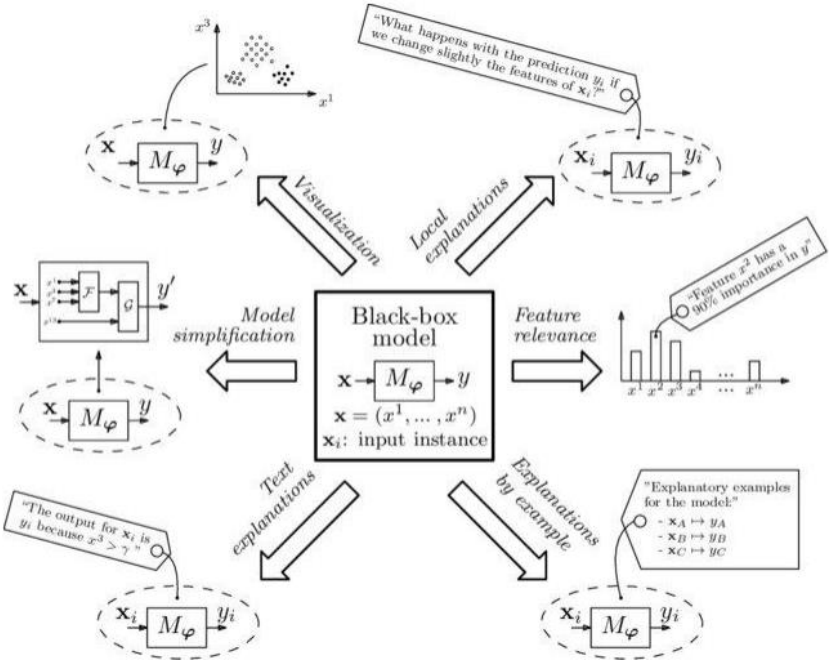
Post-Hoc Methods are partitioned following two main concepts: Model Agnostic/Model Specific and Local/Global techniques.



Model Agnostic techniques work for any kind of ML models, while Model Specific ones rely on a certain model structure. Global methods give an explanation for all the units in the dataset, whereas Local ones just for a bunch of dataset units (but you may always repeat the Local explanation on all the units of interest). Image credits to Stiglic

Pre-hoc and post-hoc Methods

Pre-hoc and post-hoc methods are two primary approaches in Explainable AI (XAI) used to explain the behaviour of machine learning models. They differ in when and how the explanations are generated, in relation to the model's training and prediction phases.



Pre-hoc Methods (Intrinsic Interpretability)

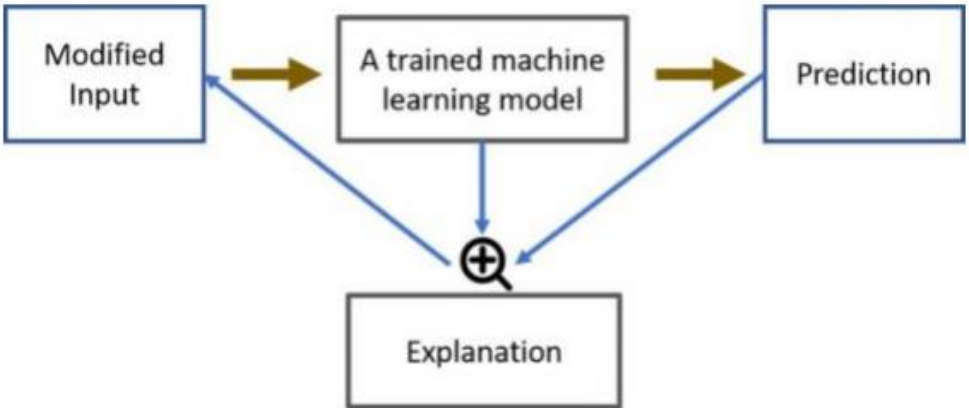
Pre-hoc methods, also known as intrinsic or built-in interpretability methods, are applied before or during model training. These methods focus on using inherently interpretable models, meaning that the models themselves are designed to be simple and transparent, allowing users to understand their decision-making process directly.

Characteristics of Pre-hoc Methods

● Model Simplicity: These methods rely on models that are simple enough for humans to easily interpret without the need for additional tools. Common models include:

- ○ Linear regression
- ○ Logistic regression
- ○ Decision trees
- ○ Rule-based models

Disadvantages: Pre-hoc models often sacrifice accuracy and performance in complex tasks in favour of interpretability



Post-hoc Methods (Post-training Interpretability)

Post-hoc methods, on the other hand, are applied after the model has been trained. These methods attempt to explain the decisions of complex, black-box models like deep neural networks, random forests, and ensemble methods, which are not inherently interpretable.

Post-hoc approaches provide explanations for specific predictions or model behaviours without altering the model itself.

Characteristics of Post-hoc Methods

- **Applied After Training**: These methods are used once the model has been trained, making them suitable for explaining black-box models.
- **Flexible Application**: They can be used with any type of model, making them more versatile than pre-hoc methods.
- **Local or Global Explanations**: Some post-hoc methods provide explanations for individual predictions (local), while others explain the overall model behaviour (global).

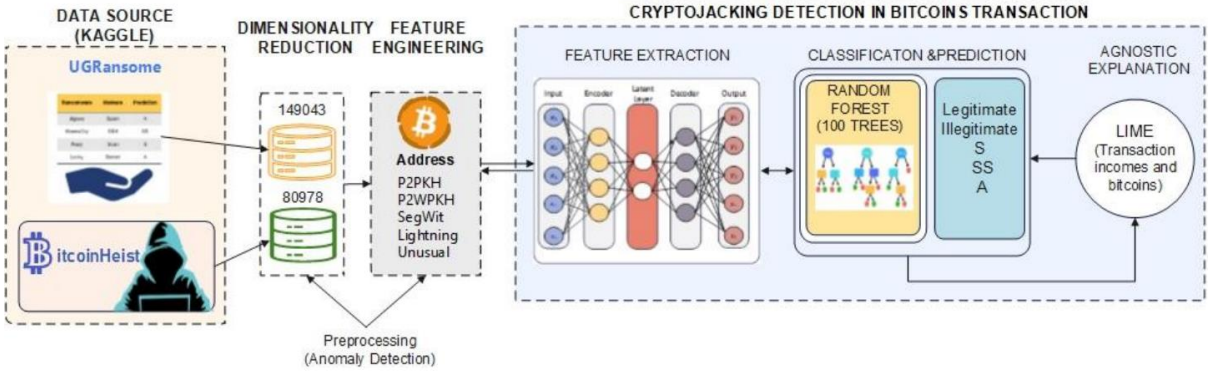
Common Post-hoc Techniques

- **LIME** (Local Interpretable Model-Agnostic Explanations): LIME approximates the model locally using simpler, interpretable models (e.g., linear models, Random Forest), which help explain a single prediction.
- **SHAP** (SHapley Additive exPlanations): SHAP assigns an importance score to each feature for a specific prediction based on cooperative game theory, offering local and global explanations.
- **Partial Dependence Plots (PDP)**: PDPs show how a feature affects the model's predictions on average, helping to explain the global behaviour of the model.
- Feature Importance Scores: These scores rank the features based on how much they contribute to the model's predictions.

Examples of Post-hoc Methods

- LIME: When a neural network makes a classification decision, LIME can be used to approximate the network's behavior around a particular input by fitting an interpretable model (like a linear model) locally.
- SHAP: After training a random forest, SHAP can be used to compute the contribution of each feature to a specific prediction, enabling more understandable explanations.

Examples of Post-hoc Methods



Advantages:

● Can explain highly accurate and complex models (e.g., deep learning models) without changing their structure.

● Flexible and can be applied to any black-box model.

Disadvantages:

● Explanations may not always be perfect or fully accurate representations of how the black-box model works.

● Computational cost can be high, especially for large models and datasets.

Pre-hoc vs. Post-hoc Comparison

Pre-hoc vs. Post-hoc Comparison

Aspect	Pre-hoc Methods	Post-hoc Methods
When Applied	During model design and training	After the model is trained
Model Type	Simple, interpretable models	Complex, black-box models
Interpretability	Inherently interpretable	Requires external methods for interpretation
Examples	Linear regression, Decision trees, Logistic regression	LIME, SHAP, PDP, Feature Importance
Trade-offs	May sacrifice performance for transparency	More powerful models but require extra effort to explain
Use Case	Suitable for simpler tasks where interpretability is critical	Suitable for complex tasks requiring high performance and later explanation
Explainability	Global (the whole model is	Can be local (specific to predictions) or global

Trade-offs in XAI

● **Accuracy vs Interpretability**: Simple models (e.g., decision trees) are interpretable but may be less accurate. Complex models (e.g., deep neural networks) are accurate but harder to explain.

● **Global vs Local Explanations**: Some techniques provide global explanations (understanding the model as a whole), while others focus on local explanations (understanding specific predictions).

Introduction to Large Language Models (LLMs)

What are LLMs?

● Large Language Models are AI models trained on vast amounts of text data to understand, generate, and manipulate human language. They are typically based on architectures like transformers and have billions of parameters, allowing them to perform complex natural language tasks.

Popular Examples of LLMs:

● GPT (Generative Pretrained Transformer): A family of models by OpenAI, including GPT-3 and GPT-4, known for tasks like text generation, translation, and summarization.

● BERT (Bidirectional Encoder Representations from Transformers): A transformer-based model that performs well on tasks like text classification and question answering.

● T5 (Text-to-Text Transfer Transformer): A unified model that frames all NLP tasks as text-to-text tasks, improving flexibility across various natural language tasks.

Key concepts in LLMS:

Transformers Architecture

- ○ Introduced in the paper "Attention is All You Need" (2017), transformers rely on attention mechanisms, which allow models to weigh the importance of different words in a sequence, improving performance on long-range dependencies.

Pretraining and Fine-tuning

- ○ Pretraining: LLMs are trained on large corpora of text data using unsupervised tasks (e.g., next-word prediction) to learn language representations.
- ○ Fine-tuning: Once pretrained, LLMs are adapted to specific tasks (e.g., sentiment analysis, summarization) using labelled datasets.

Self-Attention Mechanism

● A core component of transformers, self-attention allows the model to focus on different parts of the input sentence, capturing context more effectively.

Contextual Understanding

● Unlike earlier models (e.g., RNNs, LSTMs), LLMs capture bidirectional context, meaning they can understand the full context of a word based on its surrounding words, making them highly effective for complex NLP tasks.

Applications of LLMs

● Text Generation: LLMs can generate coherent and contextually appropriate text based on prompts, useful for chatbots, content creation, and automated storytelling.

● Translation and Summarization: Translate text from one language to another or summarize lengthy documents into concise versions.

● Sentiment Analysis: LLMs can analyze the sentiment behind text (e.g., positive, neutral, negative), useful in social media analysis, customer feedback, etc.

● Question Answering: Models like GPT-4 can answer factual questions, leveraging their vast knowledge base learned during pretraining.

The Intersection of XAI and LLMs

● Challenges with Interpretability in LLMs

  ○ Due to their massive size and complexity, LLMs are often considered black-box models, making it difficult to understand how they make specific decisions.

● Applying XAI to LLMs

  ○ Techniques like LIME and SHAP can be used to explain why a certain word or phrase was important in an LLM's prediction.

  ○ Attention Visualization: In transformer models, the self-attention weights can be visualized to show which words the model focused on when making prediction

Use Cases for XAI in LLMs

● **Bias Detection**: XAI methods can help identify and mitigate biases in LLMs (e.g., gender or racial biases) by analyzing how different inputs affect the model's predictions.

● **Interpretation of Generated Text**: For sensitive applications (e.g., legal document generation), XAI can help ensure that LLMs generate appropriate and fair text.

Limitations and Future Directions

● XAI Limitations

  ○ Most XAI methods provide approximations of model behavior, which may not fully explain the internal workings of complex models like LLMs.

  ○ Some XAI techniques can be computationally expensive or difficult to apply to models with billions of parameters.

● LLM Challenges

  ○ Data Bias: LLMs often reflect the biases present in the training data.

  ○ Overfitting to Language Patterns: LLMs might generate plausible but factually incorrect text because they optimize for fluency over factual accuracy.

Future of XAI in LLMs

● The development of more robust and scalable XAI methods that can be applied to larger, more complex models.

● Integration of ethical and fairness considerations into both LLM training and XAI explanations to create more trustworthy AI systems.

Explainable AI (XAI) and Large Language Models (LLMs) represent two critical developments in the AI landscape. While LLMs offer impressive capabilities in understanding and generating human language, XAI plays a vital role in ensuring that these systems are transparent, accountable, and trustworthy. The intersection of XAI with LLMs is an active area of research, aiming to make advanced AI systems interpretable without sacrificing their performance. Understanding the trade-offs and challenges involved will be key to developing future AI systems that are both powerful and understandable.