

Big Data



د/معاذ عبده الصبرى

-
1. Data Mining vs. Big Data
 2. Big Data Life Cycle
 - Big Data Generation
 - ✓ **Big Data Source**
 - ✓ **Different Types of Data**
 - Data Collection
 - Data Preprocessing
 - Big Data Analytics
 - ✓ **Exploratory Data Analysis**
 - ✓ **Data Modelling**
 - ✓ **Model Evaluation**
 - Visualizing Big Data

Data Mining vs. Big Data

- التنصيب في البيانات هو عملية اكتشاف الأنماط أو الاتجاهات أو المعلومات المفيدة من مجموعات البيانات الكبيرة باستخدام تكنيات مختلفة، بما في ذلك التحليل الإحصائي والتعلم الآلي والذكاء الاصطناعي.
- الهدف هو التركيز على إيجاد رؤى ومعرفة ذات معنى داخل البيانات، غالباً من خلال تحديد الأنماط أو العلاقات التي قد لا تكون واضحة على الفور.
- تشير البيانات الضخمة إلى مجموعات بيانات كبيرة ومعقدة للغاية والتي لا تكفي أدوات وأساليب معالجة البيانات التقليدية للتعامل معها بكفاءة.
- الهدف من البيانات الضخمة هو معالجة وتخزين وتحليل كميات هائلة من البيانات لاستخراج رؤى قيمة ودعم اتخاذ القرار.
- باختصار، بينما يركز التنصيب على البيانات على استخلاص المعرفة والأنماط من البيانات، بينما البيانات الضخمة تتعامل مع تحديات إدارة وتحليل مجموعات البيانات الكبيرة والمتنوعة. غالباً ما يكون التنصيب عن البيانات أسلوباً يستخدم في السياق الأوسع لتحليلات البيانات الضخمة، حيث يتطلب حجم البيانات وتعقيدها أدوات وأساليب متخصصة للتحليل والتفسير الفعال.

Data Quality: Why Preprocess the Data?

Measures for data quality: A multidimensional view

- ✓ Accuracy: correct or wrong, accurate or not
- ✓ Completeness: not recorded, unavailable, ...
- ✓ Consistency: some modified but some not, dangling, ...
- ✓ Timeliness: timely update?
- ✓ Believability: how trustable the data are correct?
- ✓ Interpretability: how easily the data can be understood?

Big Data Life Cycle

Big Data Life Cycle

- تحقق البيانات الضخمة فوائد كبيرة، بدءاً من الأفكار التجارية المبتكرة مثل الطرق الحديثة لعلاج الأمراض والتغلب على جميع التحديات الطبية. وتنشأ هذه التحديات نظراً لتطور التكنولوجيا وتوليد الكثير من البيانات اليوم. ولقد تم إنشاء تقنيات للبيانات الضخمة قادرة على الالتقاط والتحليل بشكل فعال. وتتضمن البنية التحتية للبيانات الضخمة نماذج حوسية جديدة يمكنها معالجة الحسابات الموزعة والمتوازية مع تخزين وأداء قابلين للتطوير بشكل كبير. مثل أداة البيانات الضخمة Hadoop (إطار العمل)، وHDFS (التخزين)، وMapReduce (المعالجة). يوضح الشكل 1.10 دورة حياة البيانات الضخمة.
 يتم التقاط البيانات التي تصل بسرعة عالية من مصادر متعددة بتنسيقات بيانات مختلفة وتخزينها في منصة مثل HDFS وNoSQL ثم تتم معالجتها مسبقاً لجعل البيانات مناسبة للتحليل. ثم بعد ذلك يتم تمرير البيانات للمعالجة مسبقاً وتخزينها في منصة التخزين إلى طبقة التحليلات، حيث تتم معالجة البيانات باستخدام أدوات البيانات الضخمة مثل MapReduce وإجراء التحليل على البيانات المعالجة للكشف عن المعرفة المخفية منها.

Big Data Life Cycle

و تعد التحليلات وخوارزميات التعلم الآلي من المفاهيم المهمة في دورة حياة البيانات الضخمة. فعلى سبيل المثال تحليل النص هي نوع من التحليل الذي يتم إجراؤه على البيانات النصية غير المنظمة. كما يتم إجراء التحليل التنبؤي لسلوك المستهلك وتحليل اهتمامات المستهلك على البيانات النصية المستخرجة من مصادر مختلفة عبر الإنترن트 مثل وسائل التواصل الاجتماعي و مواقع البيع بالتجزئة عبر الإنترن트 وغير ذلك الكثير. لقد جعل التعلم الآلي تحليلات النص ممكنة.

يتم تمثيل البيانات التي تم تحليلها بشكل مرئي بواسطة أدوات التصور مثل Tableau لتسهيل فهمها من قبل المستخدم النهائي لاتخاذ القرارات السليمة والمناسبة.

وتشمل دورة حياة البيانات الضخمة المراحل التي تنتهي إليها معالجة وإدارة مجموعات البيانات الكبيرة والمعقدة. في حين أن الأطر والمنهجيات المحددة قد تختلف، فإن دورة حياة البيانات الضخمة النموذجية تتضمن المراحل التالية:

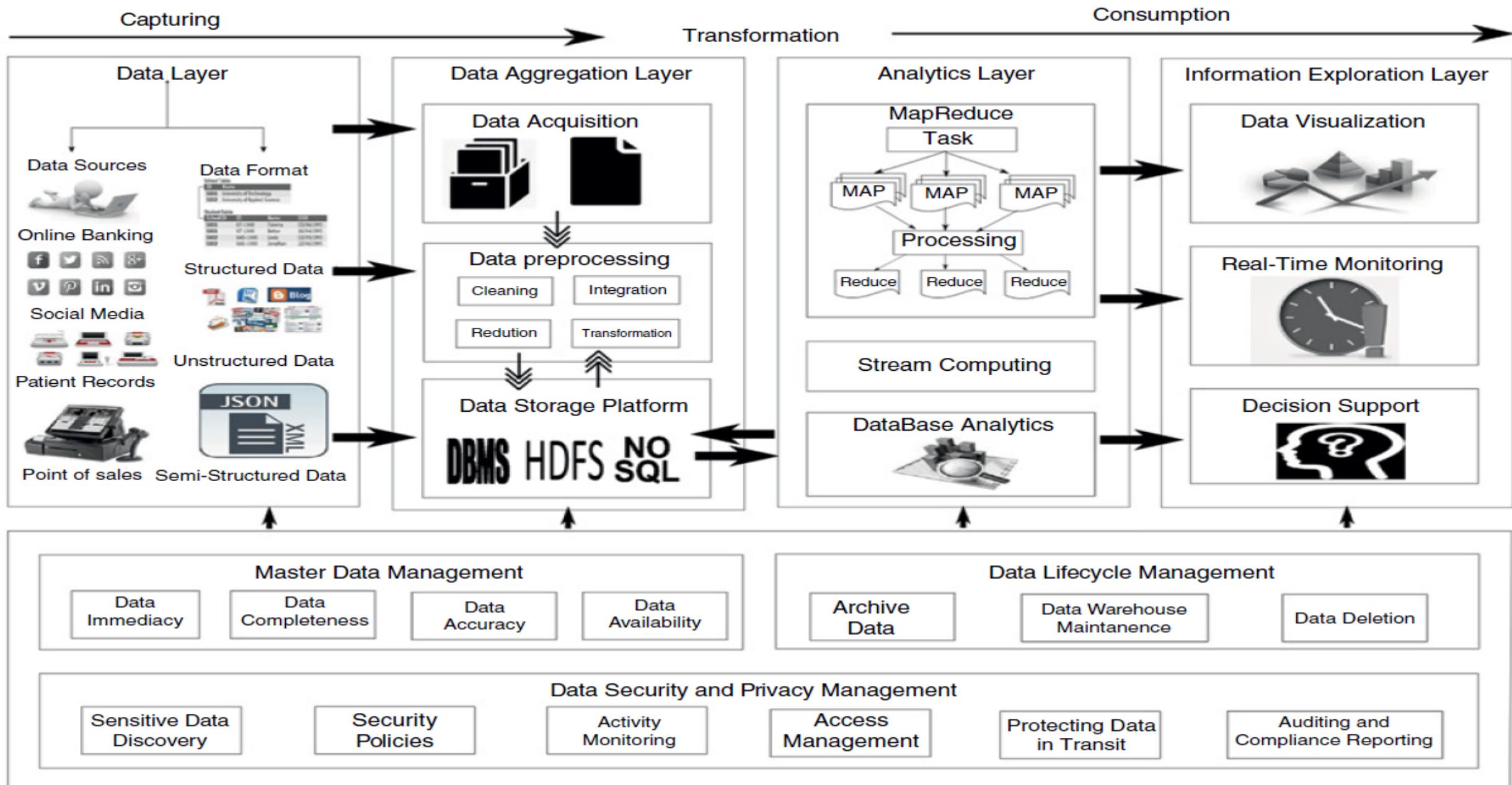


Figure 1.10 Big data life cycle.

Big Data Life Cycle

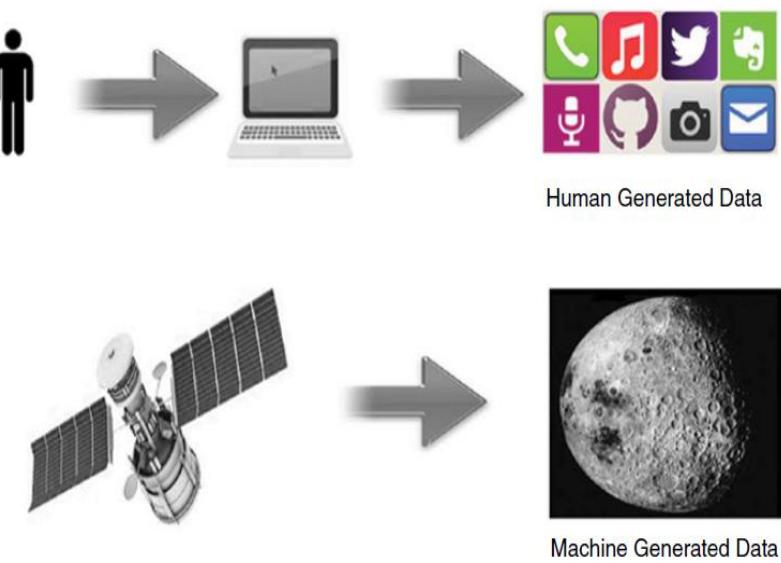


Figure 1.6 Human- and machine-generated data.

وتشمل دورة حياة البيانات الضخمة المراحل التي تنتهي إليها معالجة وإدارة مجموعات البيانات الكبيرة والمعقدة. في حين أن الأطر والمنهجيات المحددة قد تختلف، فإن دورة حياة البيانات الضخمة النموذجية تتضمن المراحل التالية:

1. توليد البيانات الكبيرة

المرحلة الأولى من دورة حياة البيانات الضخمة هي توليد البيانات. يتسع حجم البيانات الناتجة من مصادر متعددة تدريجياً. تمت مناقشة مصادر هذا الحجم الكبير من البيانات في "مصادر البيانات الضخمة".

يتم إنشاء البيانات بواسطة الآلة أو بواسطة الإنسان. تشير البيانات التي ينشئها الإنسان إلى البيانات التي يتم انتاجها كنتيجة لتفاعلات بين البشر والآلات. وتعد رسائل البريد الإلكتروني والمستندات ونشرات *Facebook* بعضًا من هذه البيانات. تقوم تطبيقات الكمبيوتر أو الأجهزة بإنشاء بيانات يتم إنشاؤها آلياً دون تدخل بشري نشط. حيث تعد هذه البيانات المستمدة من أجهزة الاستشعار وأنظمة الإنذار بال Kovarث وأنظمة التنبؤ بالطقس وبيانات الأقمار الصناعية بعضًا منها والتي يتم إنشاؤها بواسطة الآلة. يمثل الشكل 1.6 البيانات التي أنشأها البشر في وسائل التواصل الاجتماعي المختلفة، ورسائل البريد الإلكتروني المرسلة، والصور التي التقظوها، وبيانات الآلة التي أنشأها القمر الصناعي.

مصادر البيانات الضخمة

Big Data Source

- تأتي البيانات الضخمة من مصادر متعددة، وتحتاج إلى معالجة وتحليل لاستخراج المعرفة منها. فيما يلي بعض المصادر الشائعة للبيانات الضخمة:
 - وسائل التواصل الاجتماعي Social Media : تولد منصات مثل Facebook و Twitter و LinkedIn و Instagram كميات هائلة من البيانات في شكل منشورات وتغريدات وصور وتفاعلات.
 - بيانات المعاملات Transaction Data: تولد معاملات البيع بالتجزئة والمشتريات عبر الإنترنت والمعاملات المالية كميات كبيرة من البيانات، مما يوفر رؤى حول سلوك العملاء واتجاهات السوق.
 - بيانات الاستشعار Sensor Data: تنتج أجهزة إنترنت الأشياء IoT، مثل أجهزة الاستشعار الذكية والأجهزة القابلة للارتداء والأجهزة المتصلة، تدفقات مستمرة من البيانات، وتلتقط معلومات حول العالم المادي.
 - بيانات الويب Web and Clickstream Data تعد البيانات الناتجة عن تفاعلات المستخدم على مواقع الويب، بما في ذلك النقرات ومسارات التنقل والوقت الذي يقضيه على الصفحات، ذات قيمة لفهم سلوك المستخدم وتحسين وظائف موقع الويب.

مصادر البيانات الضخمة Big Data Source

- ملفات السجل Log Files: تسجل سجلات الخادم وسجلات التطبيقات وسجلات النظام الأحداث والأنشطة، وتتوفر معلومات حول أداء النظام والأخطاء وأنشطة المستخدم.
- البيانات البيومترية Biometric Data: تساهم بصمات الأصابع والتعرف على الوجه والبيانات البيومترية الأخرى في البيانات الضخمة في مجالات مثل الأمن والرعاية الصحية والتحقق من الهوية.
- البيانات الجينومية Genomic Data: في أبحاث الرعاية الصحية وعلم الجينوم، تساهم مجموعات البيانات الكبيرة من المعلومات الجينومية في فهم الأنماط الجينية والطب الشخصي.
- البيانات الحكومية وال العامة Government and Public Data: تقوم الوكالات الحكومية بإنشاء مجموعات كبيرة من البيانات المتعلقة بالتركيبة السكانية والخدمات العامة والسجلات الإدارية، والتي يمكن أن تكون ذات قيمة للبحث والتحليل.
- محتوى النص والوسائط المتعددة Text and Multimedia Content: تساهم المستندات ورسائل البريد الإلكتروني وملفات الصوت والفيديو في البيانات الضخمة، خاصة مع تزايد انتشار إنشاء المحتوى ومشاركته.

مصادر البيانات الضخمة Big Data Source

- بيانات الجهاز المحمول Mobile Device Data: تقوم الهواتف المحمولة بـتوليد البيانات من خلال تتبع الموقع واستخدام التطبيقات والاتصالات، مما يوفر نظرة ثاقبة لسلوك المستخدم وتقضياته.
 - بيانات الطقس والبيئة Weather and Environmental Data: تولد أجهزة استشعار المناخ والأقمار الصناعية ومحطات الطقس مجموعات بيانات كبيرة تستخدّم للتنبؤ بالطقس والرصد البيئي والبحث.
 - البيانات اللوجستية وسلسلة التوريد Logistics and Supply Chain Data: تساهُم البيانات المتعلقة بحركة البضائع وعمليات سلسلة التوريد وإدارة المخزون في تحسين الخدمات اللوجستية وتحسين الكفاءة.
- تساهم هذه المصادر بشكل جماعي في مجموعات البيانات الضخمة التي تقع تحت مظلة البيانات الضخمة. إن تحليل واستخلاص الأفكار من هذه البيانات المتنوعة والضخمة يمكن أن يوفر معلومات قيمة للشركات والباحثين وصناع القرار في مختلف الصناعات.

Big Data Source مصادر البيانات الضخمة

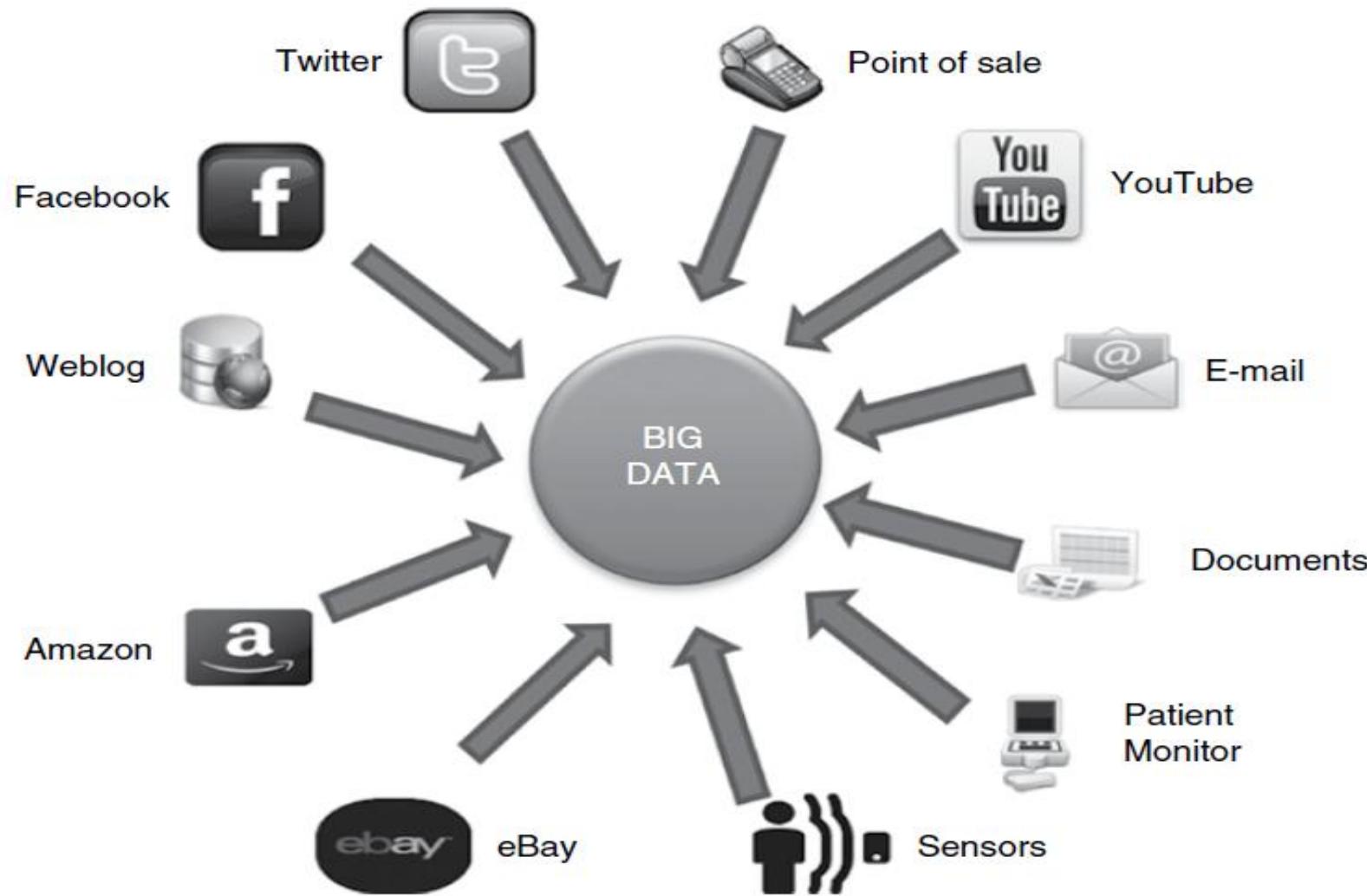


Figure 1.5 Sources of big data.

Types of Data Different

يمكن أن تمثل الأنواع البدائية التالية من البيانات الضخمة بيانات تم إنشاؤها بواسطة الآلة وأخرى تم إنشاؤها بواسطة الإنسان:

- **البيانات المنظمة** : Structured Data

البيانات التي يمكن تخزينها في قاعدة بيانات علاقية بتنسيق جدول مع صفوف وأعمدة تسمى البيانات المنظمة. تُظهر البيانات المنظمة التي يتم إنشاؤها غالباً من قبل مؤسسات الأعمال درجة عالية من التنظيم، ويمكن معالجتها بسهولة باستخدام أدوات استخراج البيانات، ويمكن الاستعلام عنها واسترجاعها باستخدام حقل المفتاح الأساسي. تتضمن أمثلة البيانات المنظمة تفاصيل الموظف والمعاملات المالية. يوضح الشكل 1.7 مثلاً للبيانات المنظمة، وهو جدول تفاصيل الموظف مع معرف الموظف كمفتاح.

Employee ID	Employee Name	Sex	Salary
334332	Daniel	Male	\$2300
334333	John	Male	\$2000
338332	Michael	Male	\$2800
339232	Diana	Female	\$1800
337891	Joseph	Male	\$3800
339876	Agnes	Female	\$4000

Figure 1.7 Structured data – employee details of an organization.

Types of Data Different

البيانات غير المنظمة Unstructured Data

تسمى البيانات الأولية وغير المنظمة والتي لا تتناسب مع أنظمة قواعد البيانات العلائقية بالبيانات غير المنظمة. ما يقرب من 80٪ من البيانات التي تم إنشاؤها غير منتظمة. تتضمن أمثلة البيانات غير المنظمة الفيديو والصوت والصور ورسائل البريد الإلكتروني والملفات النصية ونشرات وسائل الإعلام الاجتماعية. عادةً ما توجد البيانات غير المنظمة إما في ملفات نصية أو ملفات ثنائية. لا تحتوي البيانات الموجودة في الملفات الثنائية على أي بنية داخلية يمكن تحديدها، على سبيل المثال، الصوت أو الفيديو أو الصور. البيانات الموجودة في الملفات النصية هي رسائل البريد الإلكتروني ونشرات وسائل التواصل الاجتماعي وملفات pdf ومستندات معالجة النصوص.

يوضح الشكل 1.8 البيانات غير المنظمة، وهي نتيجة بحث Google.

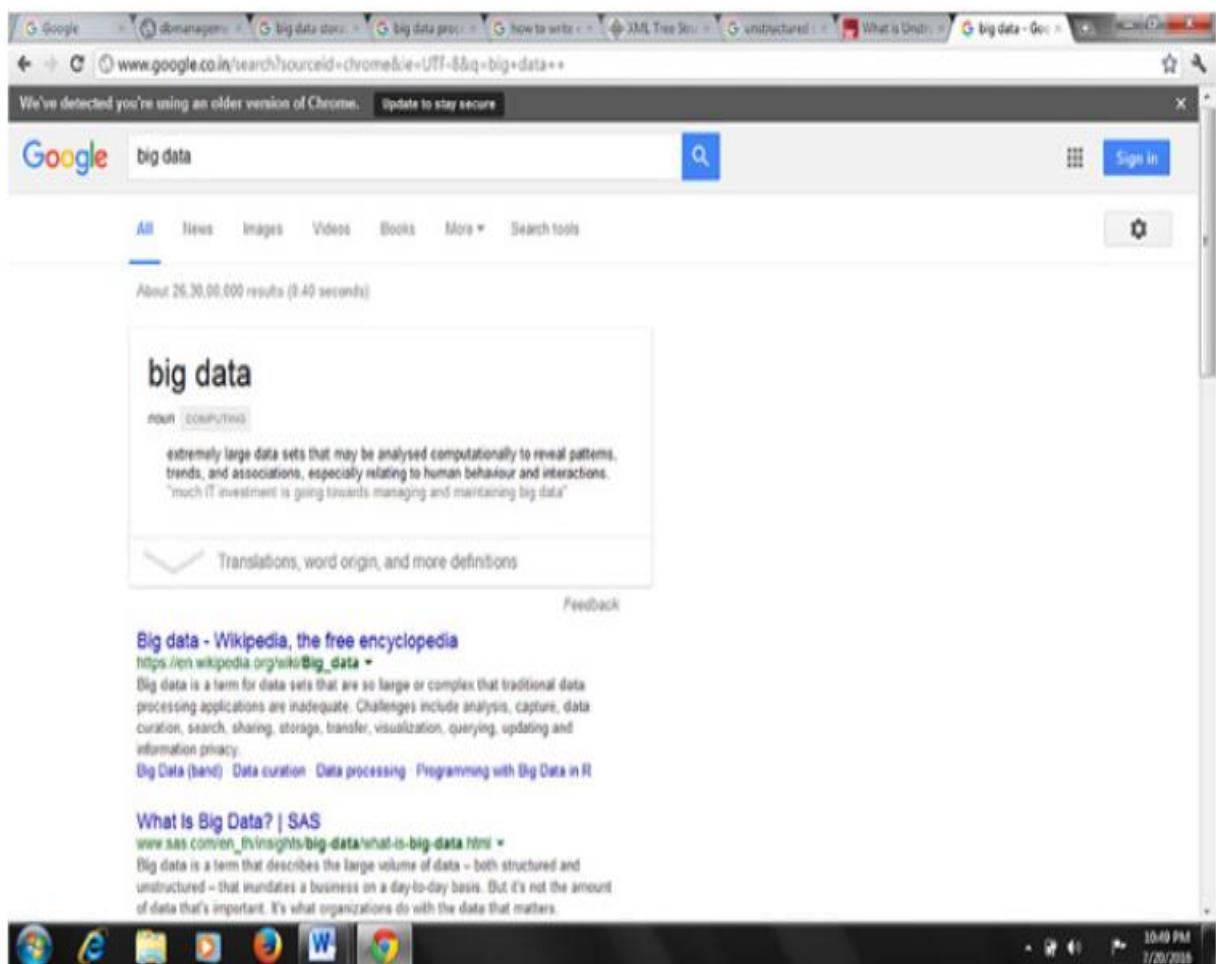


Figure 1.8 Unstructured data—the result of a Google search.

Different Types of Data

• البيانات شبه المنظمة Semi-structured Data

البيانات شبه المنظمة هي تلك التي لها بنية ولكنها لا تتناسب مع قاعدة البيانات العلائقية. يتم تنظيم البيانات شبه المنظمة، مما يسهل تحليلها عند مقارنتها بالبيانات غير المنظمة. JSON و XML هما مثالان للبيانات شبه المنظمة. الشكل 1.9 هو ملف XML يمثل تفاصيل الموظف في المؤسسة.

```
<?xml version = "1.0"?>
<Company>
<Employee>
    <EmployeeId>339876</EmployeeId>
    <FirstName>Joseph</FirstName>
    <LastName>Agnes</LastName>
    <Sex>Female</Sex>
    <Salary>$4000</Salary>
</Employee>
</Company>
```

Figure 1.9 XML file with employee details.

Data Collection

Data collection is crucial in Data Science, requiring high-quality and sufficient data. Challenges arise when gathering data from diverse sources. Gather data from various sources:

- ✓ Databases
- ✓ APIs
- ✓ Web scraping
- ✓ Public datasets
- ✓ Ensure data relevance and quality.



□ جمع البيانات Data Collection

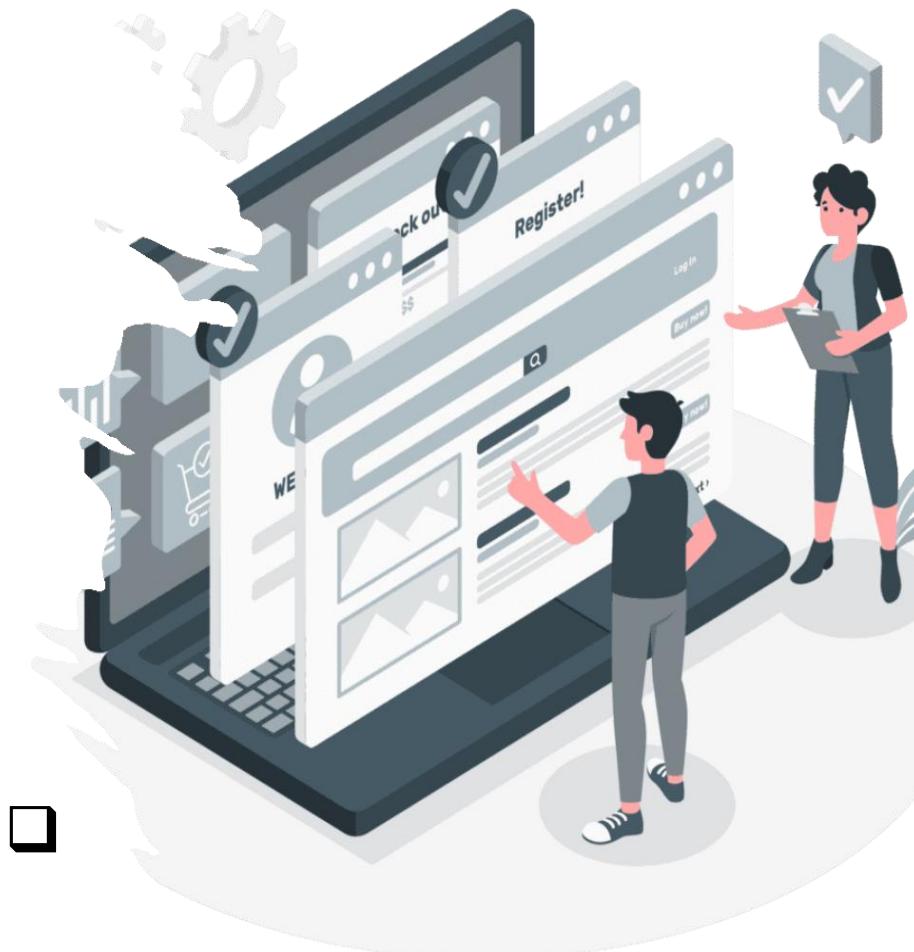
تبدأ العملية بجمع البيانات من مصادر مختلفة، مثل أجهزة الاستشعار ووسائل التواصل الاجتماعي والموقع الإلكتروني وقواعد البيانات وغيرها. يمكن أن تكون هذه البيانات بتنسيقات منظمة أو شبه منتظمة أو غير منتظمة

تتضمن مرحلة تجميع البيانات في دورة حياة البيانات الضخمة جمع البيانات الأولية، ونقل البيانات إلى منصة التخزين، ومعالجتها مسبقاً. إن الحصول على البيانات في عالم البيانات الضخمة يعني الحصول على بيانات كبيرة الحجم تصل بوتيرة متزايدة باستمرار. يتم نقل البيانات الأولية التي تم جمعها بهذه الطريقة إلى بنية تحتية للتخزين مناسبة لدعم المعالجة والتطبيقات التحليلية المختلفة.

Data Pre-processing

is a crucial step in the data science life cycle as it ensures the quality and reliability of the collected data before further analysis. Without proper processing, the data may contain errors or inconsistencies that can lead to inaccurate results and flawed decision-making.

- ✓ Handle missing values, outliers, and inconsistencies.
- ✓ Transform and normalize data.
- ✓ Prepare data for analysis.



□ تنظيف البيانات Data Preprocessing

بمجرد جمع البيانات، تخضع للمعالجة المسبقة لضمان جودتها وملاءمتها للتحليل. يتضمن ذلك

معالجة القيم المفقودة ومعالجة القيم المتطرفة وتنظيف البيانات لتعزيز سلامتها.

تحليل البيانات الضخمة 3.Big Data Analytics

تحليلات البيانات الضخمة هي مزيج من تكنيات البيانات الضخمة وأدوات التحليل.

التحليلات ليست مفهوماً جديداً: العديد من التكنيات التحليلية، وهي تحليل الانحدار والتعلم الآلي، موجودة منذ سنوات

عديدة. يعد تشابك تكنيات البيانات الضخمة مع البيانات المستمدة من مصادر جديدة وتقنيات تحليل البيانات مفهوماً تم

تطويره حديثاً. الأنواع المختلفة للتحليلات هي

1. التحليلات الوصفية Descriptive Analytics
2. والتحليلات التنبؤية Predictive Analytics
3. التحليلات الإرشادية Prescriptive Analytics

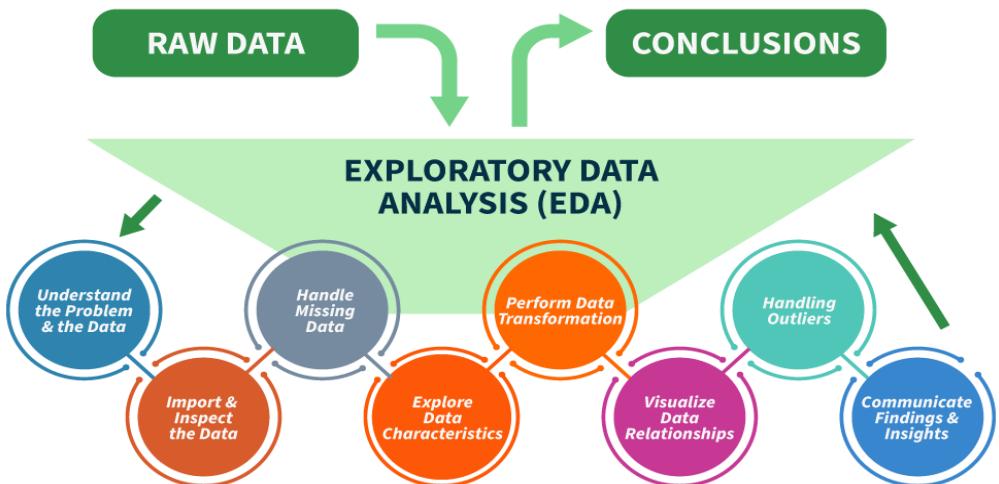
تحليل البيانات الاستكشافي (EDA)

٤٦

□ استكشاف البيانات Data Exploration

يتضمن تحليل البيانات الاستكشافية تصور البيانات وتلخيصها للحصول على رؤى أولية. تساعد الإحصائيات الوصفية والمخططات والرسوم البيانية على فهم توزيع المتغيرات وتحديد الأنماط والكشف عن العلاقات المحتملة.

Steps for Performing Exploratory Data Analysis



- ✓ Use statistical and visual methods to understand data. استخدم الأساليب الإحصائية والبصرية لفهم البيانات.
- ✓ Identify patterns, correlations, and trends. تحديد الأنماط والارتباطات والاتجاهات.
- اختيار الميزة Feature Selection: يتم تحديد الميزات (المتغيرات) ذات الصلة أو تصميمها لتعزيز المعلومات المتوفرة للتحليل. تتضمن هذه الخطوة اختيار المتغيرات الأكثر تأثيراً وإنشاء ميزات جديدة قد تعمل على تحسين أداء النموذج.
- ✓ Generate hypotheses. توليد الفرضيات.

Data Modeling نمذجة البيانات

□ بناء نموذج : Model Building

يتم استخدام الخوارزمية المحددة لبناء نموذج على مجموعة بيانات التدريب. يتم تدريب هذا النموذج للتعرف على الأنماط وال العلاقات داخل البيانات.



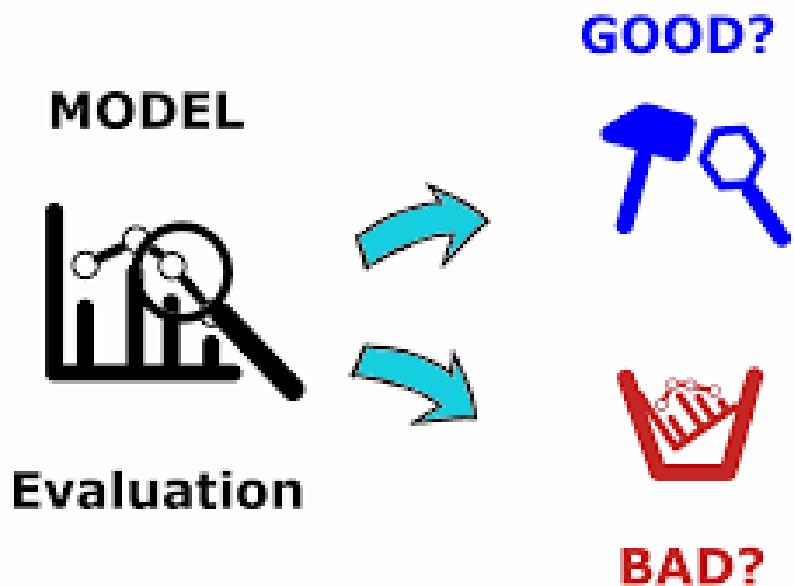
- Choose appropriate machine learning or statistical models.
- Train models on data.
- Optimize performance through tuning.
 - اختر نماذج التعلم الآلي أو الإحصائية المناسبة.
 - تدريب النماذج على البيانات.
 - تحسين الأداء من خلال الضبط.

Model Evaluation تقييم النماذج

يتم تقييم أداء النموذج باستخدام مجموعة بيانات اختبار منفصلة. تتضمن هذه الخطوة تقييم دقة النموذج وإحكامه واسترجاعه والمقاييس الأخرى ذات الصلة لضمان فعاليته في عمل التنبؤات أو الكشف عن الأنماط.

Test model performance using metrics such as:

- Accuracy
- Precision
- Recall
- F1 Score
- Validate results using cross-validation or test sets.



Deployment

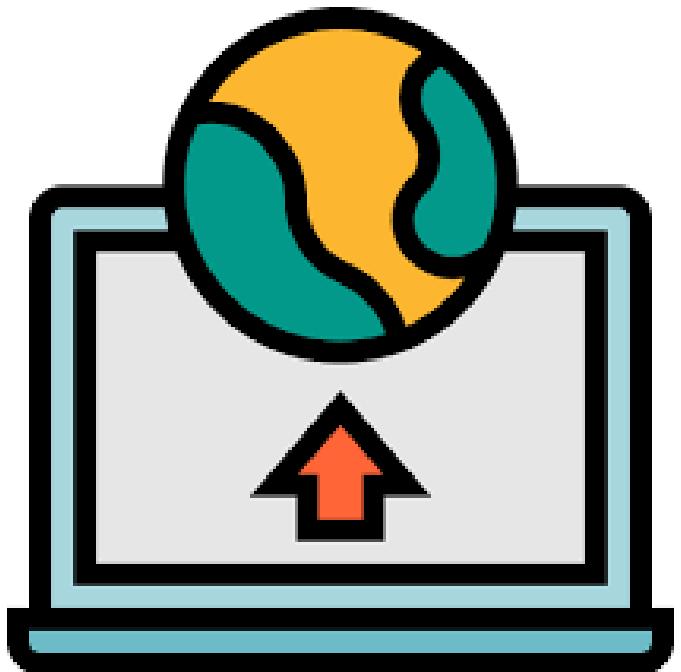
□ التطوير :Deployment

إذا كانت الأفكار ذات قيمة، فيمكن دمج النتائج في الأنظمة التشغيلية أو عمليات صنع القرار. تتضمن هذه الخطوة نشر المعرفة المكتسبة من استخراج البيانات للاستخدام العملي.

Integrate the model into production environments.

Build APIs or user interfaces.

Ensure scalability and security.



Monitoring and Maintenance

تُعد المراقبة والصيانة جانبين أساسيين في علم البيانات، إذ تضمنان دقة وموثوقية وفعالية الأنظمة القائمة على البيانات باستمرار. تتضمن المراقبة تتبع أداء النموذج وجودة البيانات في الوقت الفعلي، بينما تتضمن الصيانة اتخاذ إجراءات تصحيحية مثل إعادة تدريب النماذج أو تحديث البيانات. تمنع المراقبة والصيانة الفعالة تدهور النموذج وانحراف البيانات وتتضمن استمرار الأنظمة في تقديم القيمة بمرور الوقت.

Track model performance over time.

Retrain models as data changes.

Fix bugs and improve accuracy.



4. Visualizing Big Data

يؤدي التصور إلى اكتمال دورة حياة البيانات الضخمة، مما يساعد المستخدمين النهائيين على الحصول على رؤى من البيانات. من المديرين التنفيذيين إلى موظفي مراكز الاتصال، يرغب الجميع في استخلاص المعرفة من البيانات المجمعة لمساعدتهم في اتخاذ قرارات أفضل. بغض النظر عن حجم البيانات، فإن إحدى أفضل الطرق لتمييز العلاقات واتخاذ القرارات الحاسمة هي اعتماد أدوات تحليل البيانات وتصورها المتقدمة. الرسوم البيانية الخطية، والمخططات الشريطية، ومخططات التشتت، والمؤامرات الفقاعية، والمخططات الدائرية هي تقنيات تقليدية لتصور البيانات. تُستخدم الرسوم البيانية الخطية لتصوير العلاقة بين متغير وآخر.

5. Visualizing Big Data

تُستخدم **المخططات الشريطية** لمقارنة قيم البيانات التي تنتمي إلى فئات مختلفة ممثلة بأشرطة أفقية أو رأسية، والتي تمثل ارتفاعاتها القيمة الفعلية.

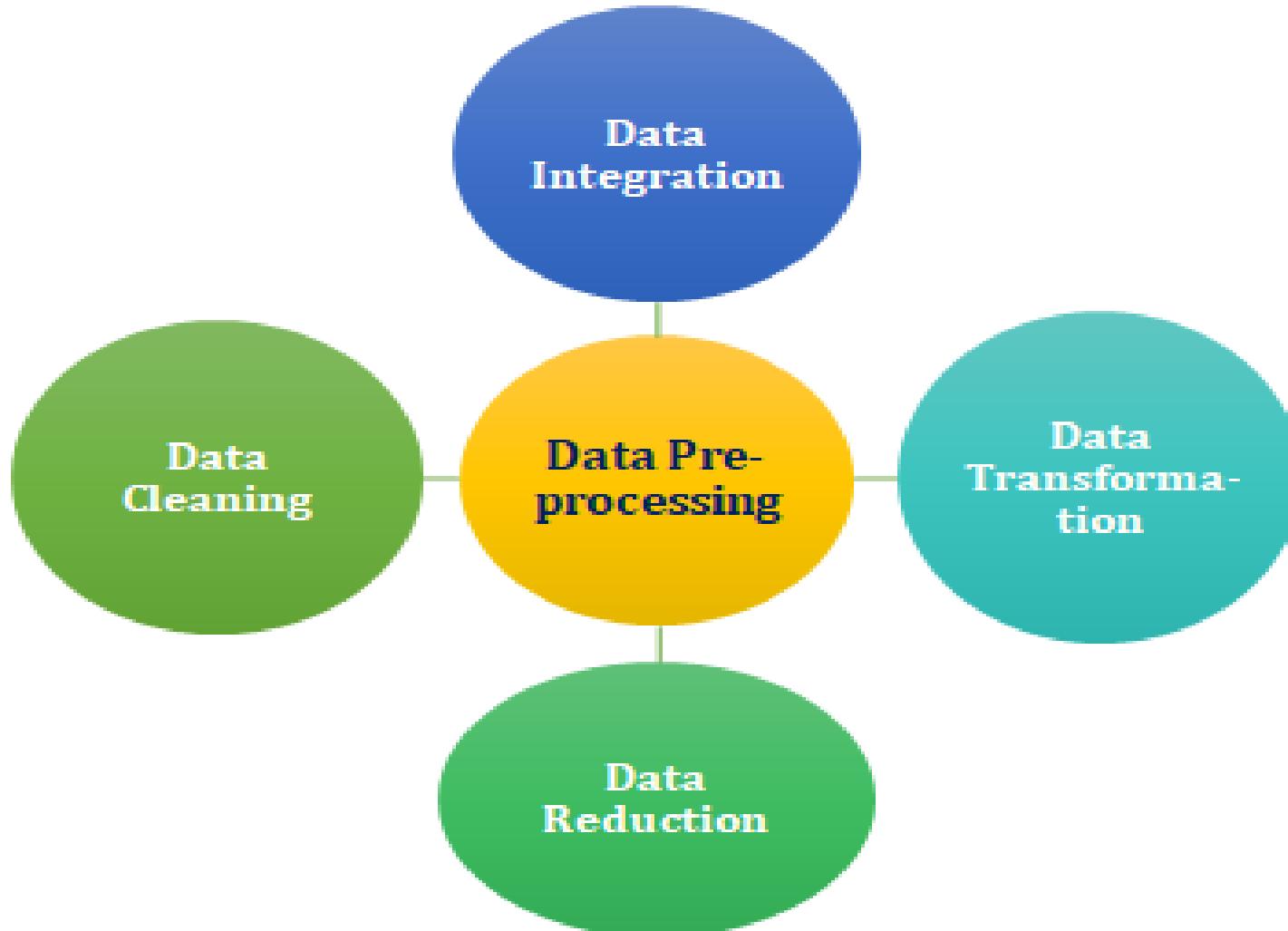
تُستخدم **مخططات التشتت** لإظهار العلاقة بين متغيرين X وY.

المخطط الفقاعي هو شكل مختلف لمخطط التشتت حيث يتم عرض العلاقات بين X وY بالإضافة إلى قيمة البيانات المرتبطة بحجم الفقاعة.

تُستخدم **المخططات الدائرية** عند مقارنة أجزاء من الظاهرة بأكملها.

Data Preprocessing

Data Preprocessing



Data Preprocessing

Data Preprocessing

المعالجة المسبقة للبيانات Data Preprocessing

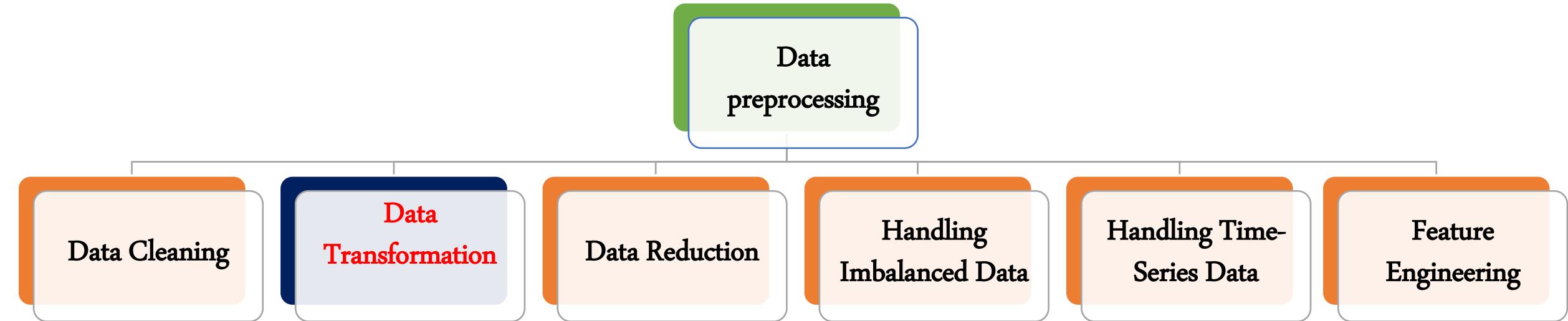
عملية مهمة يتم إجراؤها على البيانات الأولية لتحويلها إلى تنسيق مفهوم وتوفير الوصول إلى بيانات متسقة ودقيقة. البيانات الناتجة من مصادر متعددة خاطئة أحياناً وغير كاملة وغير متسقة بسبب حجمها الهائل ومصادرها غير المتجانسة، ومن غير المجدى تخزين بيانات عديمة الفائدة. بالإضافة إلى ذلك، تتطلب بعض التطبيقات التحليلية متطلبات حاسمة لجودة البيانات. وبالتالي، لتحليل البيانات بشكل فعال وفعال ودقيق، تعد المعالجة المسبقة المنهجية للبيانات أمراً ضرورياً. تتأثر جودة البيانات المصدر بعوامل مختلفة. على سبيل المثال، قد تحتوي البيانات على أخطاء مثل حقل الراتب ذو القيمة السالبة (على سبيل المثال، الراتب = -2000)، والذي ينشأ بسبب أخطاء الإرسال أو الأخطاء المطبعية أو إدخال بيانات خاطئ عن عمد من قبل المستخدمين الذين لا يرغبون في الكشف عن معلوماتهم الشخصية.

Data Preprocessing

يشير عدم الاتكمال إلى أن الحقل يفتقر إلى السمات محل الاهتمام (على سبيل المثال، التعليم = ")، والتي قد تأتي من حقل غير قابل للتطبيق أو أخطاء برمجية. يشير عدم الاتساق في البيانات إلى التناقضات في البيانات، مثل عدم تناقض تاريخ الميلاد والعمر. تنشأ حالات عدم الاتساق في البيانات عندما تكون البيانات التي تم جمعها من مصادر مختلفة بسبب عدم الاتساق في اصطلاحات التسمية بين البلدان المختلفة وعدم الاتساق في تنسيق الإدخال (على سبيل المثال، حقل التاريخ DD/MM عند تفسيره على أنه MM/DD). غالباً ما تحتوي مصادر البيانات على بيانات زائدة عن الحاجة في أشكال مختلفة، وبالتالي يجب أيضاً إزالة التكرارات في البيانات في المعالجة المسبقة للبيانات لجعل البيانات ذات معنى وخلالية من الأخطاء.

هناك عدة خطوات تشارك في المعالجة المسبقة للبيانات:

Data preprocessing



Data preprocessing



تنظيف البيانات Data cleaning

• **تنظيف البيانات**، المعروف أيضًا باسم **تنقية البيانات**، هو عملية تحديد وتصحيح الأخطاء والتناقضات وعدم الدقة في مجموعات البيانات. الهدف من تنظيف البيانات هو **تحسين جودة البيانات**، وجعلها أكثر دقة واقتداراً وموثوقية للتحليل أو لأغراض أخرى. تتضمن المهام الشائعة في تنظيف البيانات:

► التعامل مع القيم المفقودة : Handling missing values

تحديد نقاط البيانات المفقودة ومعالجتها إما عن طريق احتساب القيم أو إزالة المثيلات ذات البيانات المفقودة.

تنظيف البيانات Data cleaning

► إزالة التكرارات : Removing duplicates

تحديد وإزالة السجلات المكررة للتأكد من أن كل ملاحظة في مجموعة البيانات فريدة من نوعها.

► تصحيح الأخطاء : Correcting inaccuracies

تحديد الأخطاء في البيانات وتصحيحها، مثل الأخطاء المطبعية أو القيم غير الصحيحة أو القيم المتطرفة.

► توحيد التنسيقات : Standardizing formats

ضمان الاتساق في تنسيقات البيانات والوحدات والتمثيلات لتسهيل التحليل والمقارنات.

تنظيف البيانات Data cleaning

► التعامل مع القيم المتطرفة : Handling outliers

تحديد ومعالجة القيم المتطرفة التي قد تؤدي إلى تحريف نتائج التحليل أو أداء النموذج.

► التعامل مع التناقضات : Dealing with inconsistencies

حل التناقضات في البيانات، مثل المعلومات المترادفة أو القيم غير المتطابقة.

يعد تنظيف البيانات خطوة حاسمة في عملية إعداد البيانات قبل التحليل أو النمذجة. تساهم البيانات النظيفة

وعلية الجودة في الحصول على رؤى وقرارات أكثر دقة وموثوقية.

1. التعامل مع القيم المفقودة Handling missing values

هناك عدة حلول ممكنة للتعامل مع البيانات المفقودة، ويعتمد اختيار الطريقة على طبيعة البيانات وأهداف التحليل. فيما يلي بعض الأساليب الشائعة:

- **الحذف** :Deletion

► حذف القائمة **Listwise deletion**: إزالة الصفوف بأكملها التي تحتوي على قيم مفقودة. هذا أمر بسيط ولكنه قد يؤدي إلى فقدان معلومات قيمة.

- **الإسناد** :Imputation

► حساب المتوسط أو الوسيط أو الوضع **Mean, median, or mode imputation** : استبدل القيم المفقودة بالمتوسط أو الوسيط أو وضع القيم المرصودة لهذا المتغير.

► تعبئة للأمام أو تعبئة للخلف **Forward fill or backward fill**: نشر آخر قيمة تم ملاحظتها للأمام أو القيمة الملاحظة التالية للخلف لملء القيم المفقودة في سلسلة زمنية.

التعامل مع القيم المفقودة Handling missing values

- احتساب الانحدار الخطي Linear regression imputation: توقع القيم المفقودة باستخدام نموذج الانحدار الخطي بناءً على متغيرات أخرى.
- احتساب أقرب جiran K-nearest neighbors (KNN) imputation: استبدل القيم المفقودة بمتوسط قيم أقرب جiran.
- استيفاء Interpolation:
 - الاستيفاء المستند إلى الوقت Time-based interpolation: استخدم المعلومات المستندة إلى الوقت لتقدير القيم المفقودة في سلسلة زمنية.
 - الإسناد المتعدد Multiple Imputation:
 - إنشاء مجموعات بيانات متعددة Generate multiple datasets: احتساب القيم المفقودة عدة مرات لإنشاء مجموعات بيانات متعددة، وتحليل كل مجموعة بيانات على حدة، ودمج النتائج لمراعاة عدم اليقين في التضمين.
 - النمذجة التنبؤية Predictive Modeling:
 - إنشاء نماذج للتنبؤ بالقيم المفقودة Build models to predict missing values: استخدم حواجز ميات التعلم الآلي للتنبؤ بالقيم

التعامل مع القيم المفقودة Handling missing values

- **الطرق الخاصة بالمجال** :Domain-Specific Methods
 - **أساليب مخصصة تعتمد على معرفة المجال** :Custom methods based on domain knowledge
 - تشير الأساليب الخاصة بالمجال لتنظيف البيانات إلى الأساليب والتقنيات المصممة خصيصاً للخصائص والمتطلبات والمعرفة المحددة لمجال أو صناعة معينة، والمعروفة أيضاً باسم المجال. بمعنى آخر، تم تصميم هذه الأساليب لمواجهة تحديات تنظيف البيانات الفريدة لمجال أو تطبيق معين. فيما يلي بعض الأمثلة لتوضيح مفهوم الأساليب الخاصة بالمجال لتنظيف البيانات:
 - ❖ **تنظيف البيانات الجغرافية المكانية** : Geospatial Data Cleaning
 - قد تتطلب مجموعات البيانات الجغرافية المكانية أساليب تنظيف متخصصة. قد يتضمن ذلك التعامل مع الإحداثيات المفقودة، والتحقق من القيم المتطرفة المكانية، والتأكد من توافق المعالم الجغرافية مع جغرافيا العالم الحقيقي.

التعامل مع القيم المفقودة

Handling missing values

اعتماداً على طبيعة البيانات والمجال المحدد، قد يتم تطوير أساليب مخصصة للتعامل مع القيم المفقودة.

يعتمد اختيار الطريقة على عوامل مثل كمية ونمط البيانات المفقودة، وتوزيع المتغير، ونوع التحليل الذي سيتم إجراؤه، والافتراضات التي يمكن وضعها حول آلية البيانات المفقودة. من الضروري أن تدرس بعناية الآثار المترتبة على كل نهج وأن تكون على دراية بالتحيزات المحتملة التي تقدمها الطريقة المختارة. بالإضافة إلى ذلك، ينبغي الحفاظ على التوثيق والشفافية حول كيفية التعامل مع البيانات المفقودة لضمان إمكانية تكرار وتفسير التحليلات.

التعامل مع القيم المفقودة Handling missing values

```
import pandas as pd
import numpy as np
# قمنا بإنشاء مجموعة جديدة من البيانات تحتوي على بيانات خالية كما هو موضح في المثال التالي
# Create a sample DataFrame with missing values
data = {'Name': ['Alice', 'Bob', 'Charlie', np.nan, 'Eve'],
        'Age': [25, 30, np.nan, 35, 28],
        'Salary': [50000, 60000, 70000, np.nan, 55000]}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Alice	25.0	50000.0
1	Bob	30.0	60000.0
2	Charlie	NaN	70000.0
3	NaN	35.0	NaN
4	Eve	28.0	55000.0

This creates a DataFrame with missing values denoted by np.nan (NumPy's representation of missing/undefined data).

التعامل مع القيم المفقودة Handling missing values

1. إزالة الصفوف التي تحتوي على قيم مفقودة: Removing rows with missing values

```
# إزالة الصفوف التي تحتوي على قيم مفقودة: Removing rows with missing values
# Drop rows with missing values
df_cleaned = df.dropna()

print("\nDataFrame after removing rows with missing values:")
print(df_cleaned)
```

DataFrame after removing rows with missing values:

	Name	Age	Salary
0	Alice	25.0	50000.0
1	Bob	30.0	60000.0
4	Eve	28.0	55000.0

التعامل مع القيم المفقودة

Handling missing values

هذا المثال يتم اسناد وتعينة القيم الخالية بقيمة المدى #

```
import numpy as np
import pandas as pd
# A dictionary with list as values
GFG_dict = {'C++': [10, 20, 30, 40],
             'php': [25, np.Nan, np.Nan, 29],
             'python': [15, 14, 17, 11],
             'math': [21, 22, 23, 25]}

# Create a DataFrame from dictionary
gfg = pd.DataFrame(GFG_dict)
# Finding the mean of the column having NaN
mean_value = gfg['php'].mean()
# Replace NaNs in column S2 with the
# mean of values in the same column
gfg['php'].fillna(value=mean_value, inplace=True)
print('Updated Dataframe:')
print(gfg)
```

Updated Dataframe:

	C++	php	python	math
0	10	25.0	15	21
1	20	27.0	14	22
2	30	27.0	17	23
3	40	29.0	11	25

2. إسناد القيم المفقودة: Imputing missing values

التعامل مع القيم المفقودة Handling missing values

3. الاستيفاء هو طريقة لتقدير القيم المفقودة بناءً على القيم المرصودة في مجموعة البيانات. في سياق بيانات السلسل الزمنية، يمكن أن يكون الاستيفاء مفيداً لملء القيم المفقودة بين النقاط الزمنية المرصودة. فيما يلي مثال باستخدام Python و pandas للاستيفاء المستند إلى الوقت:

```
main.py
1 import pandas as pd
2 import numpy as np
3
4 date_rng = pd.date_range(start='2022-01-01', end='2025-01-01', freq='D')
5
6 data = {'value': [1, 2, np.nan, 4, 5, np.nan, np.nan, 8, 9, 10]}
7 df = pd.DataFrame(data, index=date_rng[:10])
8
9 print("Original DataFrame")
10 print(df)
```

Original DataFrame	
	value
2022-01-01	1.0
2022-01-02	2.0
2022-01-03	NaN
2022-01-04	4.0
2022-01-05	5.0
2022-01-06	NaN
2022-01-07	NaN
2022-01-08	8.0
2022-01-09	9.0
2022-01-10	10.0

```
main.py
1 import pandas as pd
2 import numpy as np
3
4 date_rng = pd.date_range(start='2022-01-01', end='2025-01-01', freq='D')
5
6 data = {'value': [1, 2, np.nan, 4, 5, np.nan, np.nan, 8, 9, 10]}
7 df = pd.DataFrame(data, index=date_rng[:10])
8
9 df['value_interpolated'] = df['value'].interpolate()
10 print(df)
```

	value	value_interpolated
2022-01-01	1.0	1.0
2022-01-02	2.0	2.0
2022-01-03	NaN	3.0
2022-01-04	4.0	4.0
2022-01-05	5.0	5.0
2022-01-06	NaN	6.0
2022-01-07	NaN	7.0
2022-01-08	8.0	8.0
2022-01-09	9.0	9.0
2022-01-10	10.0	10.0

التعامل مع القيم المفقودة Handling missing values

```
# sklearn
#sklearn
# sklearn
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Create a sample DataFrame with missing values
data = {'Name': ['Ali', 'Qasm', 'Asma', np.nan, 'Marya'],
        'Age': [25, 30, np.nan, 35, 28],
        'Salary': [50000, 60000, 70000, np.nan, 55000]}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25.0	50000.0
1	Qasm	30.0	60000.0
2	Asma	NaN	70000.0
3	NaN	35.0	NaN
4	Marya	28.0	55000.0

في هذا المثال نلاحظ استخدام مكتبة جديدة وهي `sklearn`. تعتبر هذه المكتبة من اهم المكتبات المستخدمة حديثاً في التعلم الالي. يمكن استخدام النمذجة التنبؤية لتنظيف البيانات في بايثون، وتحديداً لحساب القيم المفقودة. أحد الأساليب الشائعة هو بناء نماذج تنبؤية للتنبؤ بالقيم المفقودة بناءً على المتغيرات الأخرى في مجموعة البيانات.

فيما يلي مثال باستخدام نموذج تنبؤي بسيط، مثل الانحدار الخطى، لإدراج القيم المفقودة في بايثون بمساعدة مكتبة `scikit-Learn`.

التعامل مع القيم المفقودة Handling missing values

```
# Create a copy of the DataFrame for imputation
df_imputed = df.copy()
# Identify columns with missing values
columns_with_missing = df_imputed.columns[df_imputed.isnull().any()]
# Loop through columns with missing values and impute using linear regression
for column in columns_with_missing:
    # Create a linear regression model for imputation
    model = LinearRegression()
    # Identify independent variables (features) and target variable (column with missing values)
    features = df_imputed.columns[df_imputed.columns != column]
    target = column
    # Split data into observed and missing values
    observed_data = df_imputed.dropna(subset=[target])
    missing_data = df_imputed[df_imputed[target].isnull()]
    # Fit the model on observed data
    model.fit(observed_data[features], observed_data[target])
    # Predict missing values
    predictions = model.predict(missing_data[features])
    # Impute missing values in the DataFrame
    df_imputed.loc[df_imputed[target].isnull(), target] = predictions
print("\nDataFrame after predictive modeling imputation:")
print(df_imputed)
```

2. التعامل مع القيم المتكررة Removing duplicates

```
import pandas as pd
جامعة الجزيرة - اب اليمن برمجة متقدمة بايثون#
احد مجموعه او عينة من البيانات والتي تحتوي على قيم متكررة كما هو موضح في هذا المثال#
# Create a sample DataFrame with duplicate values
data = {'Name': ['Ali', 'Mohammed', 'Qasem', 'Ali', 'Qasem', 'Mona'],
        'Age': [25, 30, 35, 25, 28, 30],
        'Salary': [50000, 60000, 70000, 50000, 55000, 60000]}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25	50000
1	Mohammed	30	60000
2	Qasem	35	70000
3	Ali	25	50000
4	Qasem	28	55000
5	Mona	30	60000

تعد إزالة التكرارات من مجموعة البيانات مهمة شائعة لتنظيف البيانات. في بايثون، توفر مكتبة الباندا طريقة ملائمة للتعامل مع القيم المكررة. فيما يلي مثال لكيفية إزالة التكرارات من DataFrame باستخدام الباندا:

التعامل مع القيم المتكررة Removing duplicates

```
# Remove duplicates based on all columns
#استخدام دوال حذف هذه البيانات
df_no_duplicates = df.drop_duplicates()

print("\nDataFrame after removing duplicates:")
print(df_no_duplicates)
```

DataFrame after removing duplicates:

	Name	Age	Salary
0	Ali	25	50000
1	Mohammed	30	60000
2	Qasem	35	70000
4	Qasem	28	55000
5	Mona	30	60000

التعامل مع القيم المتكررة Removing duplicates

```
# Remove duplicates based on the 'Name' column
```

كما لاحظنا في المثال السابق انه في اسم مكرر ما زال موجود ولذلك من اجل حذف هذا الاسم سيتم التعامل معه كالتالي

```
df_no_duplicates_name = df.drop_duplicates(subset=['Name'])
```

```
print("\nDataFrame after removing duplicates based on 'Name':")
```

```
print(df_no_duplicates_name)
```

DataFrame after removing duplicates based on 'Name':

	Name	Age	Salary
0	Ali	25	50000
1	Mohammed	30	60000
2	Qasem	35	70000
5	Mona	30	60000

3. تصحيح الأخطاء

جامعة الجزيرة - اباليمن برمجة متقدمة بايثون#

#5- لاحظ في هذا المثال وجود القيمة

```
import pandas as pd  
# Create a sample DataFrame with inaccuracies  
data = {'Name': ['Ali', 'Qasm', 'Alia', 'Asma', 'Yasmean'],  
        'Age': [25, 30, -5, 35, 28],  
        'Salary': [50000, 60000, 70000, 75000, 55000]}
```

```
df = pd.DataFrame(data)  
print("Original DataFrame:")  
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25	50000
1	Qasm	30	60000
2	Alia	-5	70000
3	Asma	35	75000
4	Yasmean	28	55000

يتضمن تصحيح الأخطاء في مجموعة البيانات تحديد الأخطاء في قيم البيانات وتصحيحها. في بايثون، تُستخدم مكتبة الباندا بشكل شائع لمعالجة البيانات وتنظيفها. لنستعرض مثلاً لتصحيح DataFrame: الأخطاء في

3. تصحيح الأخطاء

```
# Correct inaccuracies in the 'Age' column by replacing negative values with NaN
df['Age'] = df['Age'].apply(lambda x: x if x >= 0 else pd.NA)

print("\nDataFrame after correcting inaccuracies:")
print(df)
```

DataFrame after correcting inaccuracies:

	Name	Age	Salary
0	Ali	25	50000
1	Qasm	30	60000
2	Alia	<NA>	70000
3	Asma	35	75000
4	Yasmean	28	55000

- في هذا المثال، أولاً استخدمنا الدالة `lambda` مع دالة `Apply()` لاستبدال قيم العمر السلبية بـ `pd.NA`.
قيمة خاصة تمثل قيمة مفقودة. اعتماداً على طبيعة الأخطاء، يمكنك اختيار استراتيجيات تصحيح مختلفة.

4. توحيد التنسيقات Standardizing formats

جاءة الجزيرة - اب اليمن برمجة متقدمة بايثون

```
import pandas as pd

# Create a sample DataFrame with inconsistent formats
data = {'Name': ['Ali', 'Qasm', 'Alia', 'Asma', 'Yasmean'],
        'Age': ['25', 30, '35', 'thirty', '28'],
        'Salary': ['50,000', '60,000', '70,000', '75,000', '55,000']}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25	50,000
1	Qasm	30	60,000
2	Alia	35	70,000
3	Asma	thirty	75,000
4	Yasmean	28	55,000

- This creates a DataFrame with inconsistent formats in the 'Age' and 'Salary' columns:

يتضمن توحيد التنسيقات في مجموعة البيانات ضمان التنسق في تمثيل البيانات عبر أعمدة أو صفوف مختلفة. وهذا مهم بشكل خاص عند التعامل مع المتغيرات الفئوية، أو التواريخ، أو أي أنواع بيانات أخرى حيث يمكن أن يسهل التوحيد التحليل والمقارنات. في بايثون، تُستخدم مكتبة الباندا عادةً لمعالجة البيانات وتنظيفها. دعنا نتناول مثلاً على توحيد تنسيقات :

4. توحيد التنسيقات Standardizing formats

في هذا المثال، استخدمنا الدالة `pd.to_numeric()` لتحويل العمود "العمر" إلى قيم رقمية .
تستبدل معلمة الأخطاء 'coerce'=أي قيم غير قابلة للتحويل بـ `NaN`.
استخدمنا الدالة `str.replace()` لإزالة الفواصل ثم قمنا بتحويل القيم إلى أرقام فاصلة عائمة باستخدام `astype(float)`.

```
# Standardize formats: Convert 'Age' to numeric and remove commas from 'Salary'  
df['Age'] = pd.to_numeric(df['Age'], errors='coerce') # Convert 'Age' to numeric, coerce errors to NaN  
df['Salary'] = df['Salary'].str.replace(',', '').astype(float) # Remove commas and convert 'Salary' to float  
  
print("\nDataFrame after standardizing formats:")  
print(df)
```

DataFrame after standardizing formats:

	Name	Age	Salary
0	Ali	25.0	50000.0
1	Qasm	30.0	60000.0
2	Alia	35.0	70000.0
3	Asma	NaN	75000.0
4	Yasmean	28.0	55000.0

التعامل مع القيم المتطرفة : Handling outliers

جامعة الجزيرة - اب اليمن برمجة متقدمة بايثون#

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Create a sample DataFrame with outliers
data = {'Name': ['Ali', 'Qasm', 'Alia', 'Asma', 'Yasmean'],
        'Age': [25, 30, 35, 200, 28],
        'Salary': [50000, 60000, 70000, 75000, 200000]}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25	50000
1	Qasm	30	60000
2	Alia	35	70000
3	Asma	200	75000
4	Yasmean	28	200000

تتضمن معالجة القيم المتطرفة تحديد القيم المتطرفة ومعالجتها في مجموعة البيانات. يمكن للقيم المتطرفة أن تؤثر بشكل كبير على نتائج التحليلات الإحصائية ونماذج التعلم الآلي، لذا من المهم التعامل معها بعناية. في بايثون، يتم استخدام مكتبة الباندا والأساليب الإحصائية المختلفة بشكل شائع للكشف عن الحالات الشاذة ومعالجتها. دعونا نوضح مثلاً للتعامل مع القيم المتطرفة:

التعامل مع القيم المتطرفة : Handling outliers

جامعة الجزيرة - اباليمن برمجة متقدمة بايثون#

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Create a sample DataFrame with outliers
data = {'Name': ['Ali', 'Qasm', 'Alia', 'Asma', 'Yasmean'],
        'Age': [25, 30, 35, 200, 28],
        'Salary': [50000, 60000, 70000, 75000, 200000]}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

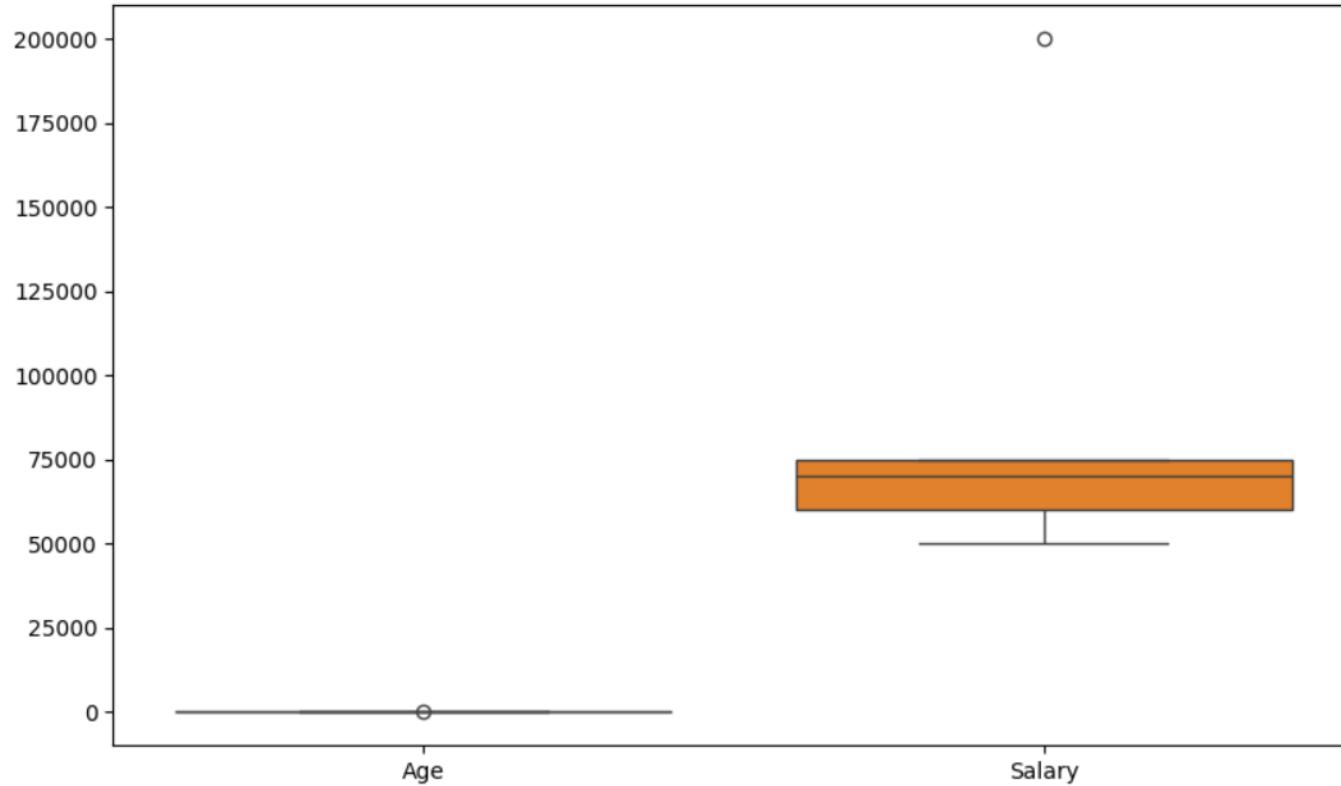
	Name	Age	Salary
0	Ali	25	50000
1	Qasm	30	60000
2	Alia	35	70000
3	Asma	200	75000
4	Yasmean	28	200000

This creates a DataFrame with outliers in the 'Age' and 'Salary' columns:

تتضمن معالجة القيم المتطرفة تحديد القيم المتطرفة ومعالجتها في مجموعة البيانات. يمكن للقيم المتطرفة أن تؤثر بشكل كبير على نتائج التحليلات الإحصائية ونماذج التعلم الآلي، لذا من المهم التعامل معها بعناية. في بايثون، يتم استخدام مكتبة الباندا والأساليب الإحصائية المختلفة بشكل شائع للكشف عن الحالات الشاذة ومعالجتها. دعونا نوضح مثلاً للتعامل مع القيم المتطرفة:

التعامل مع القيم المتطرفة : Handling outliers

```
# Visualize the data using box plots
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['Age', 'Salary']])
plt.title('Boxplot of Age and Salary')
plt.show()
```



يؤدي هذا إلى إنشاء مخطط مربع يوضح التوزيع والقيم المتطرفة المحتملة في عمودي "العمر" و"الراتب".

التعامل مع القيم المتطرفة : Handling outliers

```
C:\Windows\System32>pip install seaborn
Collecting seaborn
  Downloading seaborn-0.13.1-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (1.26.3)
Requirement already satisfied: pandas>=1.2 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (2.1.4)
Collecting matplotlib!=3.6.1,>=3.4 (from seaborn)
  Downloading matplotlib-3.8.2-cp310-cp310-win_amd64.whl.metadata (5.9 kB)
Collecting contourpy>=1.0.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading contourpy-1.2.0-cp310-cp310-win_amd64.whl.metadata (5.8 kB)
Collecting cycler>=0.10 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading fonttools-4.47.2-cp310-cp310-win_amd64.whl.metadata (160 kB)
    160.8/160.8 kB 535.7 kB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading kiwisolver-1.4.5-cp310-cp310-win_amd64.whl.metadata (6.5 kB)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.2.0)
Collecting pyparsing>=2.3.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading pyparsing-3.1.1-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.2->seaborn) (2023.4)
Requirement already satisfied: six>=1.5 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Downloading seaborn-0.13.1-py3-none-any.whl (294 kB)
  294.8/294.8 kB 1.1 MB/s eta 0:00:00
Downloading matplotlib-3.8.2-cp310-cp310-win_amd64.whl (7.6 MB)
  7.6/7.6 MB 1.2 MB/s eta 0:00:00
Downloading contourpy-1.2.0-cp310-cp310-win_amd64.whl (186 kB)
  186.7/186.7 kB 1.6 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.47.2-cp310-cp310-win_amd64.whl (2.2 MB)
  2.2/2.2 MB 1.7 MB/s eta 0:00:00
Downloading kiwisolver-1.4.5-cp310-cp310-win_amd64.whl (56 kB)
  56.1/56.1 kB 1.5 MB/s eta 0:00:00
Downloading pyparsing-3.1.1-py3-none-any.whl (103 kB)
  103.1/103.1 kB 1.5 MB/s eta 0:00:00
Installing collected packages: pyparsing, kiwisolver, fonttools, cycler, contourpy, matplotlib, seaborn
Successfully installed contourpy-1.2.0 cycler-0.12.1 fonttools-4.47.2 kiwisolver-1.4.5 matplotlib-3.8.2 pyparsing-3.1.1 seaborn-0.13.1
```

التعامل مع القيم المتطرفة : Handling outliers

```
C:\Windows\System32>pip show seaborn
Name: seaborn
Version: 0.13.1
Summary: Statistical data visualization
Home-page:
Author:
Author-email: Michael Waskom <mwaskom@gmail.com>
License:
Location: c:\users\dell\appdata\local\programs\python\python310\lib\site-packages
Requires: matplotlib, numpy, pandas
Required-by:
```

التعامل مع القيم المتطرفة : Handling outliers

افضل الحلول للتعامل مع القيم المتطرفة هو من خلال استبدالها بقيم أكثر ملاءمة. سنستخدم طريقة المدى الرباعي (IQR) لتحديد القيم المتطرفة واستبدلتها:

```
[6]: # Function to handle outliers using IQR method
def handle_outliers_iqr(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return series.apply(lambda x: lower_bound if x < lower_bound else (upper_bound if x > upper_bound else x))
# Apply the function to handle outliers in 'Age' and 'Salary'
df['Age'] = handle_outliers_iqr(df['Age'])
df['Salary'] = handle_outliers_iqr(df['Salary'])
print("\nDataFrame after handling outliers:")
print(df)
```

DataFrame after handling outliers:

	Name	Age	Salary
0	Ali	25.0	50000.0
1	Qasm	30.0	60000.0
2	Alia	35.0	70000.0
3	Asma	45.5	75000.0
4	Yasmean	28.0	97500.0

في هذا المثال، قمنا بتعريف دالة (handle_outliers_iqr) تستخدم طريقة IQR لتحديد القيم المتطرفة واستبدلتها. يتم استبدال القيم الموجودة أسفل الحد الأدنى أو أعلى من الحد الأعلى بالحد المعنوي. ثم قمنا بتطبيق هذه الوظيفة على عمودي "العمر" و"الراتب".

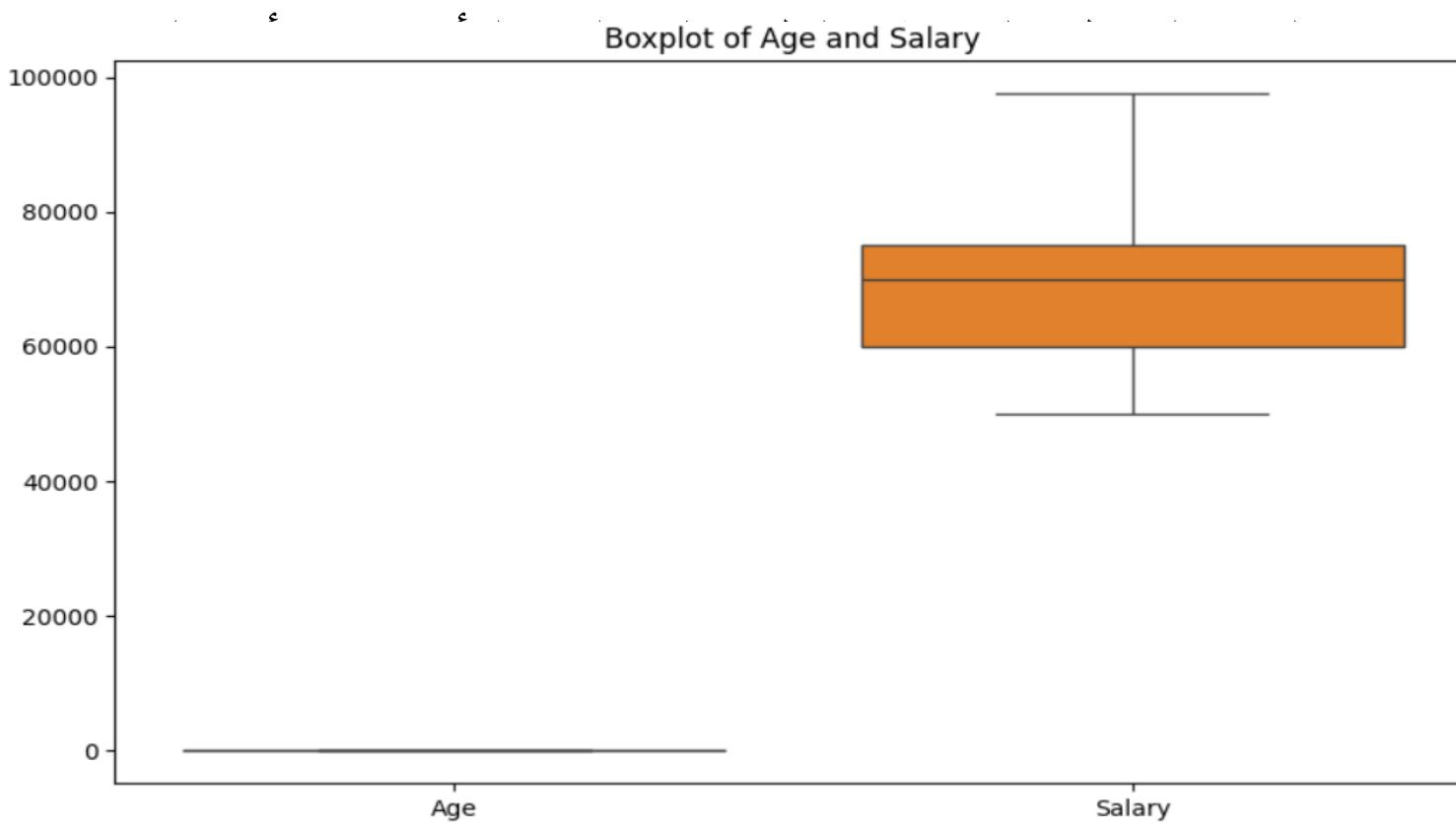
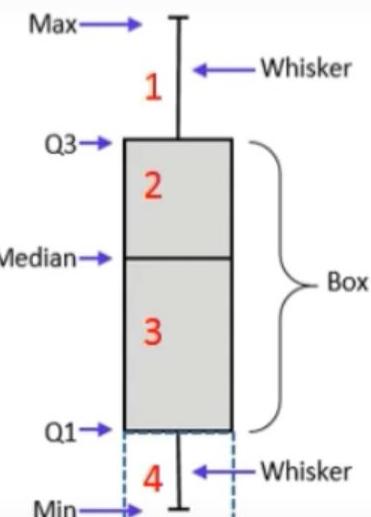
التعامل مع القيم المتطرفة : Handling outliers

والآن، تم استبدال القيم المتطرفة في أعمدة "العمر" و"الراتب" بقيم تقع ضمن الحدود المحددة بواسطة

طريقة IQR.

كثيراً ما يُقال
عن المخطط الذي

يوضح لنا القيم حيث
تحتوي على التالي :

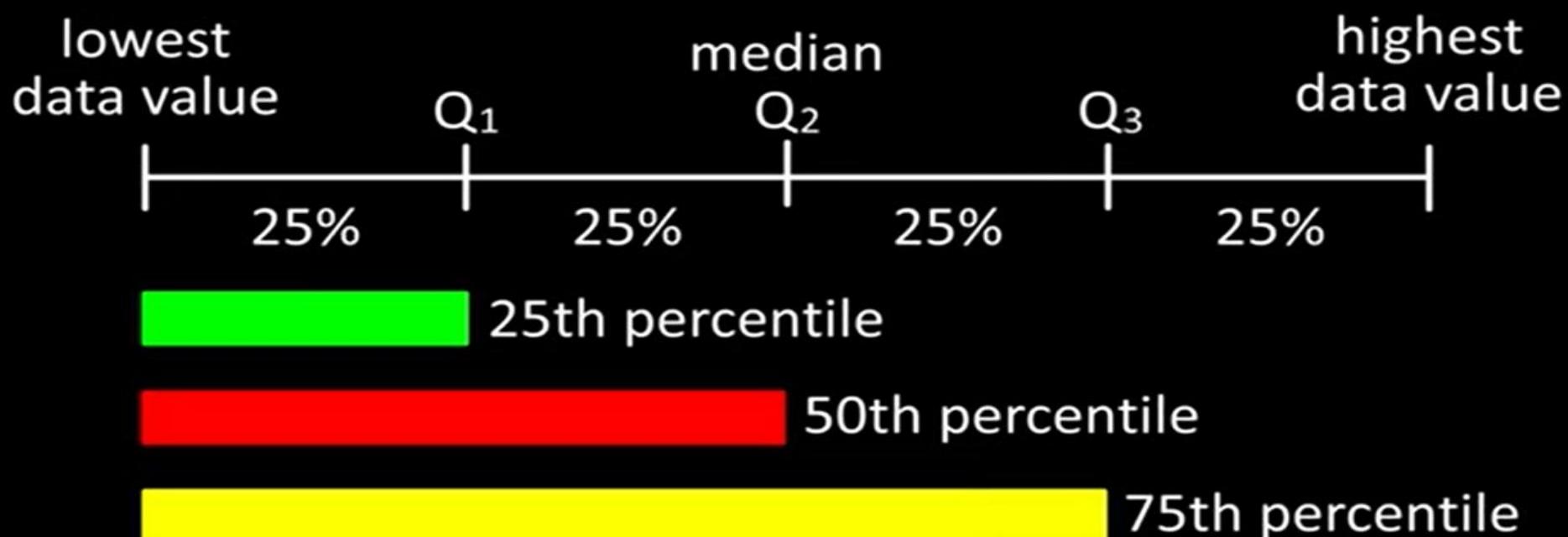


يساعد على التوازن
للتحليل أو النموذج

Quartiles, The Interquartile Range, And Outliers

quartiles

- divide a data set into 4 equal groups
- are marked Q_1 , Q_2 , and Q_3

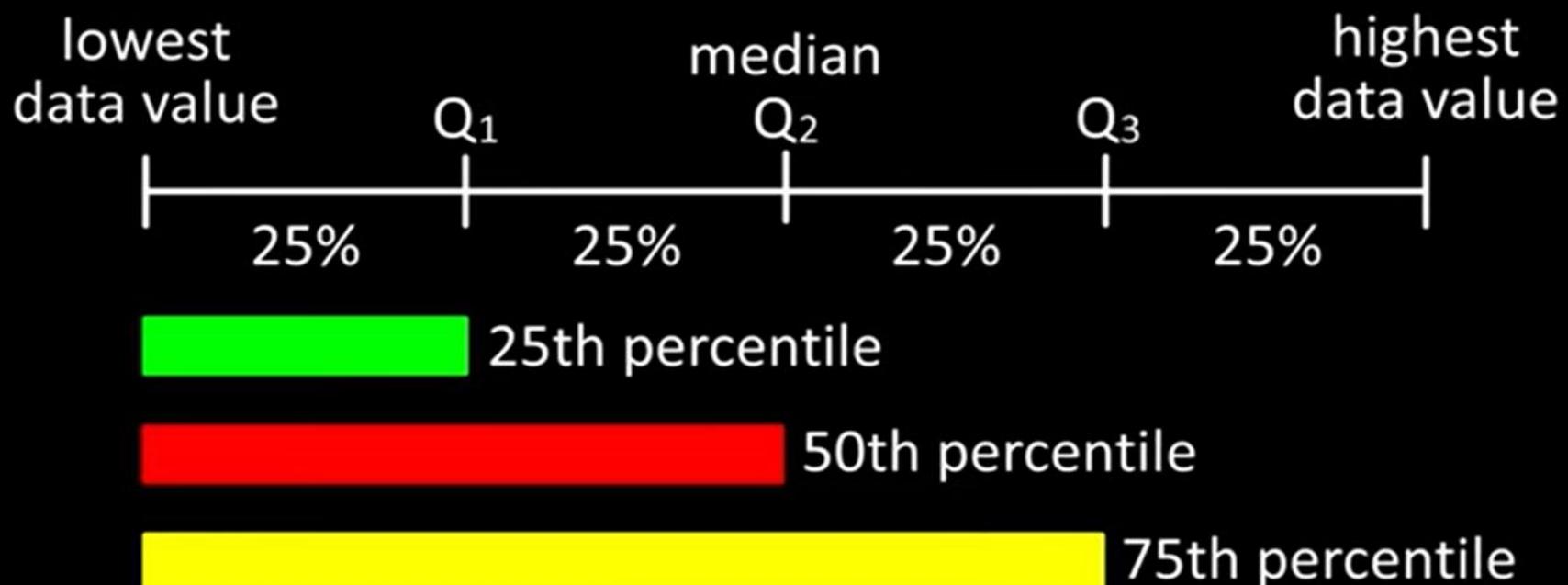


Finding data values that correspond to Q_1 , Q_2 , Q_3

Quartiles, The Interquartile Range, And Outliers

quartiles

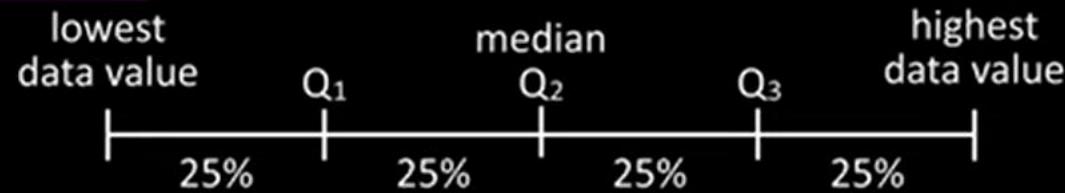
- divide a data set into 4 equal groups
 - are marked Q_1 , Q_2 , and Q_3



Finding data values that correspond to Q_1 , Q_2 , Q_3

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3



arrange the data smallest to largest

5 6 7 8 9 10 11 12 14 15 16 17 18 19 21 23 24 25 27 31

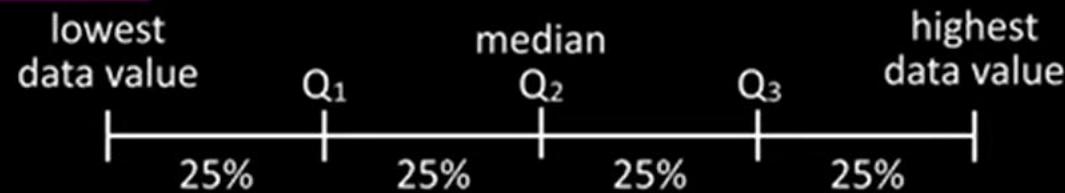
median

$$\text{median} = Q_2 =$$

20 values in
the data set

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3



arrange the data smallest to largest

5 6 7 8 9 10 11 12 14 15 16 17 18 19 21 23 24 25 27 31

median

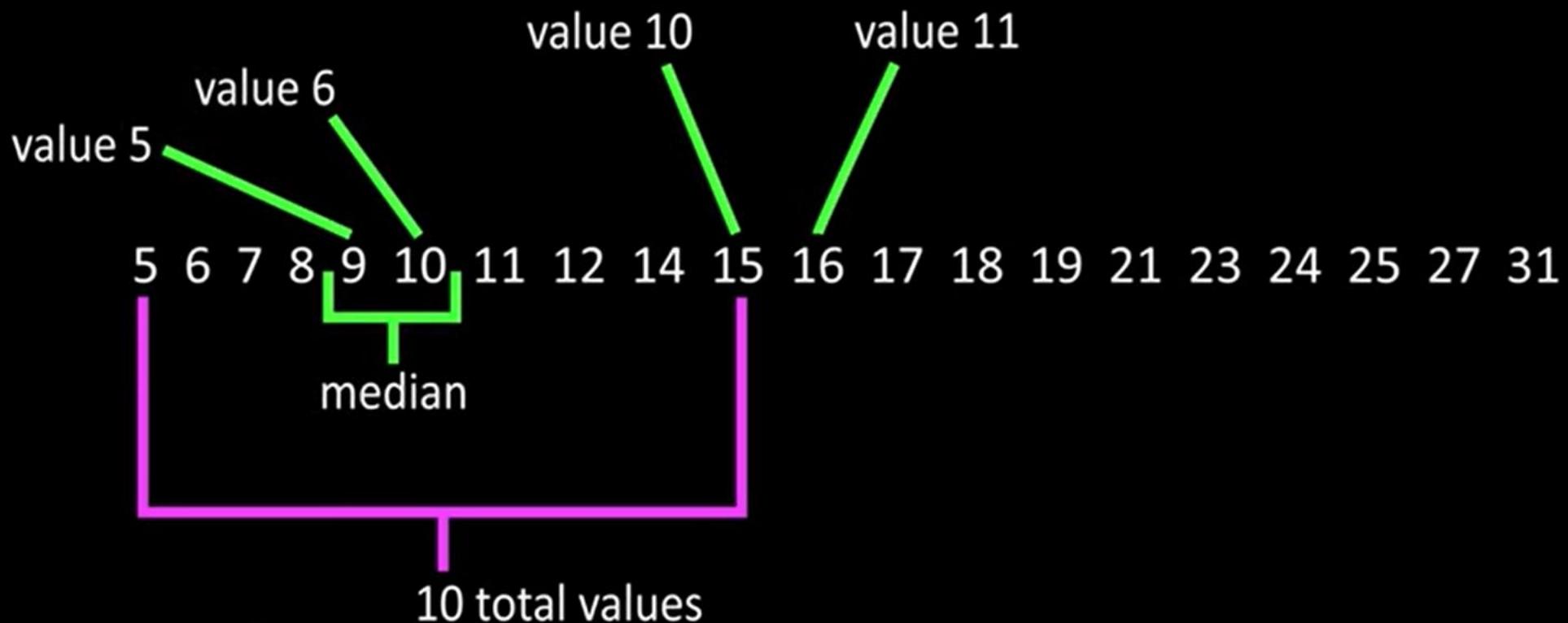
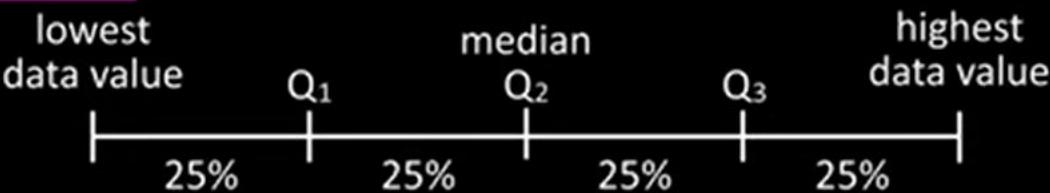
$$\frac{15 + 16}{2} = 15.5$$

$$\text{median} = Q_2 = 15.5$$

20 values in
the data set

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

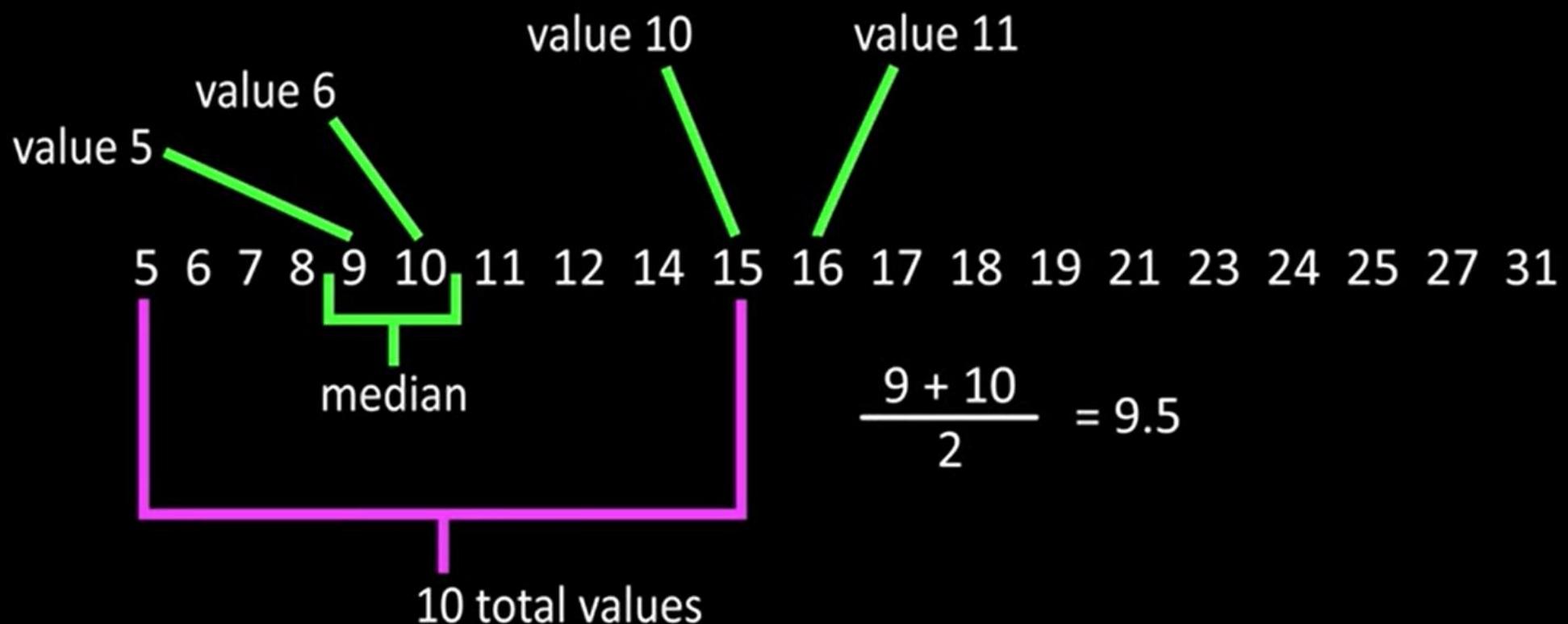
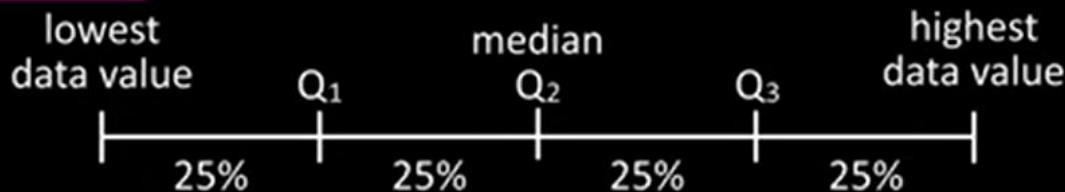


$$\text{median} = Q_2 = 15.5$$

median of values that fall below $Q_2 = Q_1 =$

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

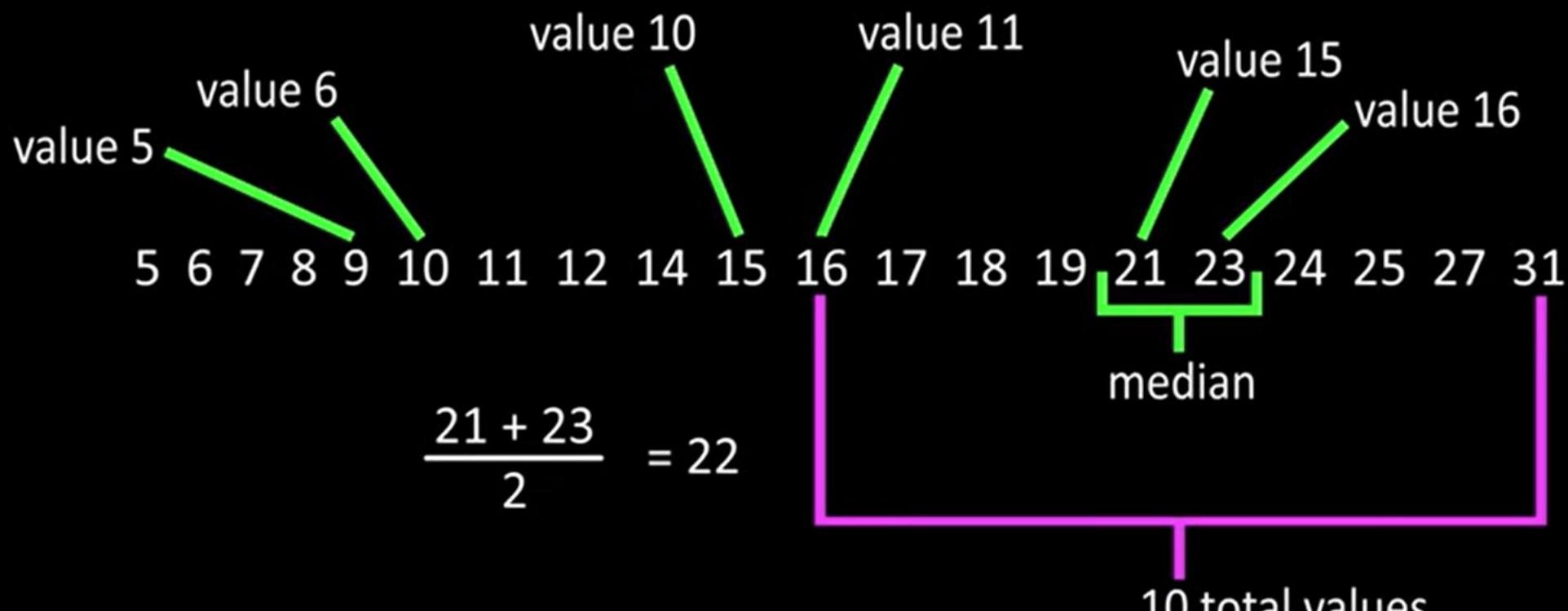
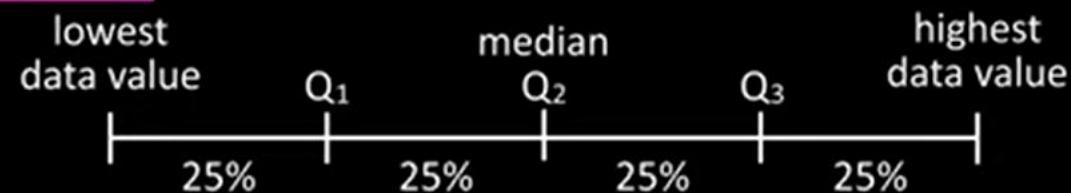


$$\text{median} = Q_2 = 15.5$$

$$\text{median of values that fall below } Q_2 = Q_1 =$$

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3



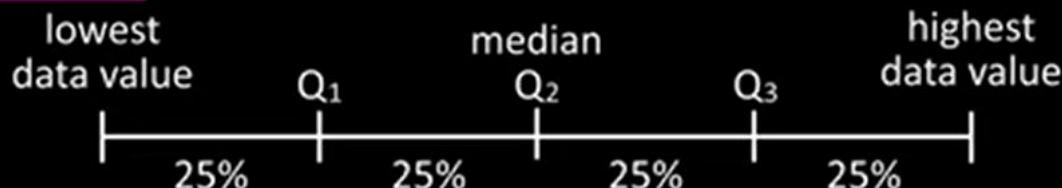
$$\text{median} = Q_2 = 15.5$$

$$\text{median of values that fall below } Q_2 = Q_1 = 9.5$$

$$\text{median of values that fall above } Q_2 = Q_3 = 22$$

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3



5 6 7 8 9 10 11 12 14 15 16 17 18 19 21 23 24 25 27 31

$$Q_1 = 9.5$$

$$Q_2 = 15.5$$

$$Q_3 = 22$$

Interquartile Range (IQR)

- difference between 3rd and 1st quartiles

$$\text{IQR} = Q_3 - Q_1$$

$$\text{IQR} = 22 - 9.5 = 12.5$$

Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

Interquartile Range (IQR)

$$IQR = Q_3 - Q_1$$

$$IQR = 22 - 9.5 = 12.5$$

$$Q_1 = 9.5$$

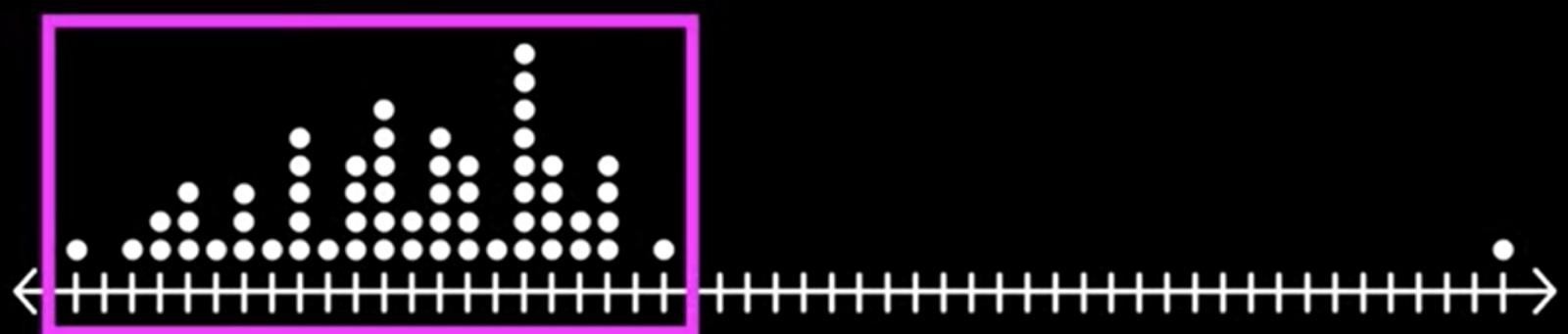
$$Q_2 = 15.5 \quad 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 14 \ 15 \ 16 \ 17 \ 18 \ 19 \ 21 \ 23 \ 24 \ 25 \ 27 \ 31$$

$$Q_3 = 22$$

IQR to identify outliers

Outlier - an extremely high or low value compared to other values in the data set

***can effect the mean and standard deviation



Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

Interquartile Range (IQR)

$$\text{IQR} = Q_3 - Q_1$$

$$\text{IQR} = 22 - 9.5 = 12.5$$

$$Q_1 = 9.5$$

$$Q_2 = 15.5$$

$$Q_3 = 22$$

5 6 7 8 9 10 11 12 14 15 16 **17** 18 19 21 23 24 25 27 31

5 6 7 8 9 10 11 12 14 15 16 18 19 21 23 24 25 27 31 **170**

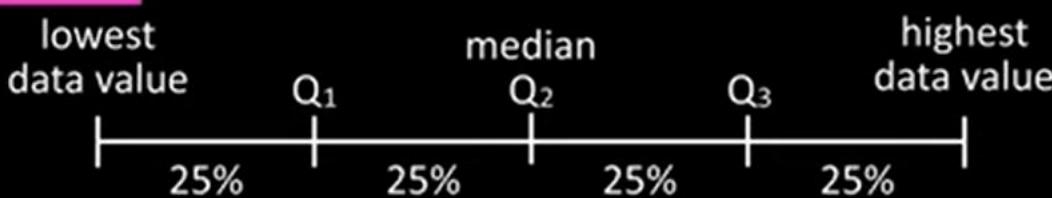
IQR to identify outliers

$$Q_1 = 9.5$$

$$Q_3 = 23.5$$

$$\text{IQR} = 23.5 - 9.5 = 14$$

$$\text{multipy IQR by } 1.5 = 14 \times 1.5 = 21$$



Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

Interquartile Range (IQR)

$$\text{IQR} = Q_3 - Q_1$$

$$\text{IQR} = 22 - 9.5 = 12.5$$

$$Q_1 = 9.5$$

$$Q_2 = 15.5$$

$$Q_3 = 22$$

5 6 7 8 9 10 11 12 14 15 16 **17** 18 19 21 23 24 25 27 31

5 6 7 8 9 10 11 12 14 15 16 18 19 21 23 24 25 27 31 **170**

IQR to identify outliers

$$Q_1 = 9.5$$

$$Q_3 = 23.5$$

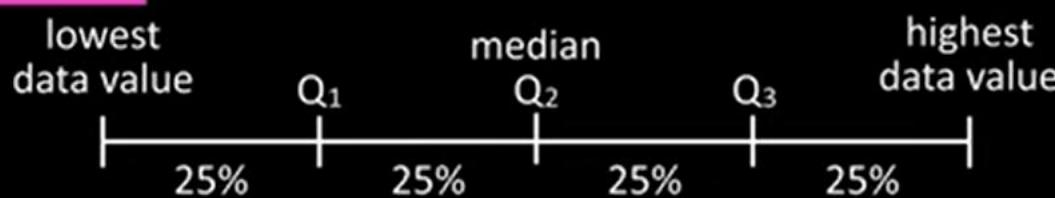
$$\text{IQR} = 23.5 - 9.5 = 14$$

range of (-11.5 to 44.5)

$$\text{multiply IQR by } 1.5 = 14 \times 1.5 = 21$$

$$Q_1 - 21 = -11.5$$

$$Q_3 + 21 = 44.5$$



Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

Interquartile Range (IQR)

$$\text{IQR} = Q_3 - Q_1$$

$$\text{IQR} = 22 - 9.5 = 12.5$$

$$Q_1 = 9.5$$

$$Q_2 = 15.5$$

$$Q_3 = 22$$

5 6 7 8 9 10 11 12 14 15 16 **17** 18 19 21 23 24 25 27 31

5 6 7 8 9 10 11 12 14 15 16 18 19 21 23 24 25 27 31 **170**

IQR to identify outliers

$$Q_1 = 9.5$$

$$Q_3 = 23.5$$

$$\text{IQR} = 23.5 - 9.5 = 14$$

range of (-11.5 to 44.5)

$$\text{multipy IQR by } 1.5 = 14 \times 1.5 = 21$$

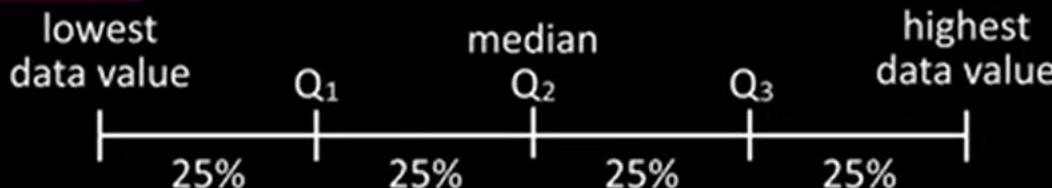
- any number outside of the range is an outlier

$$Q_1 - 21 = -11.5$$

outlier =

$$Q_3 + 21 = 44.5$$

any data value smaller than $Q_1 - 1.5(\text{IQR})$



Quartiles, The Interquartile Range, And Outliers

Finding data values that correspond to Q_1 , Q_2 , Q_3

Interquartile Range (IQR)

$$IQR = Q_3 - Q_1$$

$$IQR = 22 - 9.5 = 12.5$$

$$Q_1 = 9.5$$

$$Q_2 = 15.5$$

$$Q_3 = 22$$

5 6 7 8 9 10 11 12 14 15 16 **17** 18 19 21 23 24 25 27 31

5 6 7 8 9 10 11 12 14 15 16 18 19 21 23 24 25 27 31 **170**

IQR to identify outliers

$$Q_1 = 9.5$$

$$Q_3 = 23.5$$

$$IQR = 23.5 - 9.5 = 14$$

range of (-11.5 to 44.5)

$$\text{multipy IQR by } 1.5 = 14 \times 1.5 = 21$$

- any number outside of the range is an outlier

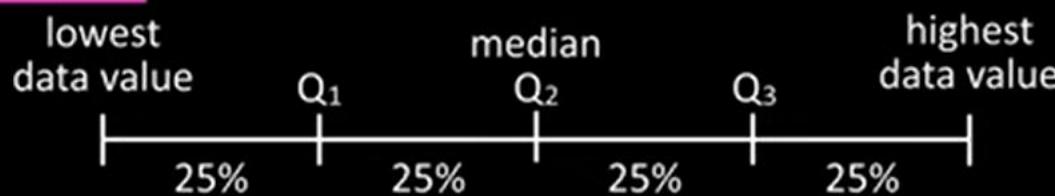
$$Q_1 - 21 = -11.5$$

outlier =

$$Q_3 + 21 = 44.5$$

any data value smaller than $Q_1 - 1.5(IQR)$

any data value larger than $Q_3 + 1.5(IQR)$



التعامل مع التناقضات : Dealing with inconsistencies

```
import pandas as pd

# Create a sample DataFrame with inconsistencies
data = {'Name': ['Ali', 'Qasm', 'Alia', 'Asma', 'Yasmean'],
        'Age': [25, 'thirty', 35, 40, 28],
        'Salary': [50000, 60000, '70,000', 75000, 55000]}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
```

Original DataFrame:

	Name	Age	Salary
0	Ali	25	50000
1	Qasm	thirty	60000
2	Alia	35	70,000
3	Asma	40	75000
4	Yasmean	28	55000

يتضمن التعامل مع حالات عدم الاتساق في مجموعة البيانات تحديد ومعالجة حالات عدم الاتساق أو الأخطاء في البيانات. يمكن أن تظهر التناقضات بطرق مختلفة، مثل **الأخطاء المطبعية** أو **غير المتطابقة** أو المعلومات المتضاربة. في بايثون، تُستخدم مكتبة الباندا بشكل شائع لمعالجة البيانات وتنظيفها. دعونا نستعرض مثلاً للتعامل مع التناقضات:

التعامل مع التناقضات : Dealing with inconsistencies

```
# Convert 'Age' to numeric, coerce errors to NaN
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Remove commas from 'Salary' and convert to numeric
df['Salary'] = df['Salary'].str.replace(',', '').astype(float)

print("\nDataFrame after dealing with inconsistencies:")
print(df)
```

DataFrame after dealing with inconsistencies:

	Name	Age	Salary
0	Ali	25.0	NaN
1	Qasm	NaN	NaN
2	Alia	35.0	70000.0
3	Asma	40.0	NaN
4	Yasmean	28.0	NaN

في هذا المثال، استخدمنا الدالة `pd.to_numeric()` لتحويل عمود "العمر" إلى قيم رقمية، وإجبار الأخطاء على `NaN` للقيم غير القابلة للتحويل. بالنسبة لعمود "الراتب"، أزلنا الفوائل باستخدام `str.replace()` ثم قمنا بتحويل القيم إلى أرقام الفاصلة العائمة باستخدام `astype(float)`.

التعامل مع التناقضات : Dealing with inconsistencies

الآن، يتم تمثيل عمود "العمر" بتنسيق رقمي متسق، وتم توحيد عمود "الراتب" عن طريق إزالة الفواصل وتحويل القيم إلى قيمة عامة. ويساعد ذلك في معالجة حالات عدم الاتساق ويضمن أن البيانات أكثر ملاءمة للتحليل.

اعتماداً على طبيعة حالات عدم الاتساق في مجموعة البيانات الخاصة بك، قد تحتاج إلى تطبيق طرق مختلفة لتنظيف البيانات وتوحيدتها. غالباً ما تكون التعبيرات العادية ووظائف معالجة السلسلة وتحويلات أنواع البيانات أدوات مفيدة في معالجة حالات عدم التناسق.

تحويل البيانات Data Transformation

يعد تحويل البيانات جانبًا أساسياً للمعالجة المسبقة للبيانات التي تتضمن تحويل البيانات الأولية إلى تنسيق مناسب للتحليل أو النمذجة. وتهدف هذه العملية إلى تحسين جودة البيانات، وجعلها أكثر قابلية للفهم، وإعدادها لمختلف التقنيات التحليلية. يشمل تحويل البيانات مجموعة من التقنيات المصممة خصيصاً لأنواع محددة من البيانات ومتطلبات التحليل. وفيما يلي شرح مفصل لبعض تقنيات تحويل البيانات الشائعة:

1. التحجيم والتطبيع Scaling and Normalization
2. الترميز المتغير الفئوي Categorical Variable Encoding
3. التعامل مع البيانات المنحرفة Handling Skewed Data
4. التجزيئ أو التمييز Binning or Discretization
5. تحجيم الميزات للخوارزميات Feature Scaling for Algorithms
6. هندسة الميزات Feature Engineering
7. تحويل بيانات النص Text Data Transformation

Min-Max and Z-Score Normalization

Numerical Example

ما هو التطبيع؟

التطبيع هو أسلوب إحصائي يستخدم في **معالجة البيانات وتحليلها لضبط القيم** في مجموعة بيانات إلى مقياس مشترك دون تشويه الاختلافات في نطاقات القيم. هذه العملية ضرورية في مجالات مختلفة، بما في ذلك الإحصاء، تحليل البيانات، وعلوم البيانات، حيث تضمن إمكانية مقارنة البيانات وتفسيرها بدقة. من خلال تحويل البيانات إلى تنسيق طبيعي، يمكن للمحللين التخفيف من تأثير القيم المتطرفة والتأكد من أن كل ميزة تساهم بالتساوي في التحليل، وخاصة في خوارزميات التعلم الآلي الحساسة لمقياس بيانات الإدخال.

أهمية التطبيع في تحليل البيانات

في تحليل البيانات، يلعب التطبيع دوراً حاسماً في تعزيز أداء النماذج الإحصائية وخوارزميات التعلم الآلي. عندما تحتويمجموعات البيانات على ميزات بمقاييس مختلفة، قد تصبح النماذج متحيزة تجاه تلك الميزات ذات النطاقات الأكبر. على سبيل المثال، في مجموعة البيانات التي تحتوي على العمر (يتراوح من 0 إلى 100) والدخل (يتراوح من 0 إلى 100,000)، يمكن أن تؤثر ميزة الدخل بشكل غير مناسب على تنبؤات النموذج. تعالج التسوية هذه المشكلة عن طريق قياس كافة الميزات إلى نطاق مماثل، عادةً بين 0 و 1 أو -1 و 1، مما يسمح بإجراء تحليل أكثر توائناً وعدلاً.

التجييم والتقطيع Scaling and Normalization

1. التجييم والتقطيع Scaling and Normalization

- القياس Scaling: إعادة قياس نطاق الميزات إلى مقياس موحد دون تشويه الاختلافات في نطاقات القيم. تتضمن تقنيات القياس الشائعة قياس Min-Max والتوحيد القياسي (تطبيع درجة Z-score normalization).
- القياس بالاعتماد على الحد الأدنى والحد الأقصى Min-Max Scaling: يقيس المعالم إلى نطاق ثابت (على سبيل المثال، بين 0 و1) باستخدام الصيغة:

$$X_{scald} = \frac{X_{max} - X_{min}}{X - X_{min}}$$

التجييم والتطبيع Scaling and Normalization

- التوجيد القياسي: تحويل المعالم إلى متوسط 0 وانحراف معياري 1 باستخدام الصيغة:

$$X_{\text{standardized}} = \frac{X - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation.

يعد القياس والتطبيع من التقنيات المستخدمة لمعالجة البيانات الرقمية مسبقاً للتأكد من أن جميع الميزات على نطاق مماثل، وهو أمر بالغ الأهمية للعديد من خوارزميات التعلم الآلي. فيما يلي مثال لكيفية تنفيذها في لغة بايثون

باستخدام مكتبة Scikit-learn الشهيرة:

Min-Max and Z-Score Normalization - Numerical Example

- Use the two methods below to normalize the following group of data: 1000, 2000, 3000, 5000, 9000
- Min-Max Normalization by setting Min=0 and Max=1
- Z-Score Normalization

Min-Max and Z-Score Normalization - Numerical Example

Min-Max Normalization

min = 1000 and Max= 9000

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Min-Max Normalization

$$V = \frac{x - \min}{\max - \min}$$

min = 1000 and Max= 9000

$$V = \frac{1000 - 1000}{9000 - 1000} = 0$$

$$V = \frac{2000 - 1000}{9000 - 1000} = 0.125$$

$$V = \frac{3000 - 1000}{9000 - 1000} = 0.25$$

$$V = \frac{5000 - 1000}{9000 - 1000} = 0.5$$

$$V = \frac{9000 - 1000}{9000 - 1000} = 1$$

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Min-Max Normalization

$$V = \frac{x - \min}{\max - \min}$$

min = 1000 and Max= 9000

$$V = \frac{1000 - 1000}{9000 - 1000} = 0$$

$$V = \frac{2000 - 1000}{9000 - 1000} = 0.125$$

$$V = \frac{3000 - 1000}{9000 - 1000} = 0.25$$

$$V = \frac{5000 - 1000}{9000 - 1000} = 0.5$$

$$V = \frac{9000 - 1000}{9000 - 1000} = 1$$

Data(v)	Normalized Data(v)
1000	0
2000	0.125
3000	0.25
5000	0.5
9000	1

Min-Max and Z-Score Normalization - Numerical Example

Z-Score Normalization

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Z-Score Normalization

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

$$\text{Mean} = \frac{(1000 + 2000 + 3000 + 5000 + 9000)}{5} = 4000$$

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Z-Score Normalization

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

$$\text{Standard Deviation} = \sqrt{\frac{\sum(x_i - \mu)^2}{n-1}}$$

$$= \sqrt{\frac{(1000 - 4000)^2 + (2000 - 4000)^2 + (3000 - 4000)^2 + (5000 - 4000)^2 + (9000 - 4000)^2}{5 - 1}}$$

$$= 2489.97$$

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Z-Score Normalization

$$z = \frac{(x - \mu)}{\sigma}$$

$$V = \frac{1000 - 4000}{2489.97} = -1.204$$

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

Mean = 4000

Standard Deviation = 2489.97

Data(v)
1000
2000
3000
5000
9000

Min-Max and Z-Score Normalization - Numerical Example

Z-Score Normalization

$$z = \frac{(x - \mu)}{\sigma}$$

$$V = \frac{1000 - 4000}{2489.97} = -1.204$$

$$V = \frac{2000 - 4000}{2489.97} = -0.803$$

$$V = \frac{3000 - 4000}{2489.97} = -0.4016$$

$$V = \frac{5000 - 4000}{2489.97} = 0.4016$$

$$V = \frac{9000 - 4000}{2489.97} = 2.008$$

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

Mean = 4000

Standard Deviation = 2489.97

Data(v)	Normalized Data(v)
1000	-1.204
2000	-0.803
3000	-0.4016
5000	0.4016
9000	2.008

التجييم والتقطيع

Scaling and Normalization

```
جامعة الجزيرة - اب - اليمن#
برمجة متقدمة - لغة البايثون #
# Data transformation
from sklearn.preprocessing import MinMaxScaler
# Sample dataset
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

# Create scaler object
scaler = MinMaxScaler()

# Fit scaler to the data and transform the data
scaled_data = scaler.fit_transform(data)

print("Scaled Data:")
print(scaled_data)
```

```
Scaled Data:
[[0.  0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.  1.  ]]
```

يتضمن القياس تحويل الميزات بحيث تقع ضمن نطاق محدد. يكون هذا مفيداً بشكل خاص عندما تحتوي المعلم على وحدات أو مقاييس مختلفة، مما يضمن عدم سيطرة أي معلم على الميزات الأخرى.

مثال:

التجيم والتقطيع Scaling and Normalization

```
# جامعة الجزيرة - اب - اليمن
# برمجة متقدمة - لغة البايثون
# Data transformation (القياس والتقطيع)
from sklearn.preprocessing import Normalizer

# Sample dataset
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

# Create normalizer object
normalizer = Normalizer()

# Fit normalizer to the data and transform the data
normalized_data = normalizer.fit_transform(data)

print("Normalized Data:")
print(normalized_data)

Normalized Data:
[[-0.4472136  0.89442719]
 [-0.08304548  0.99654576]
 [ 0.          1.        ]
 [ 0.05547002  0.99846035]]
```

يتضمن التقطيع إعادة قياس كل عينة (صف) من مجموعة البيانات للحصول على معيار وحدة (أي حجم 1). يكون هذا مفيداً عندما يختلف حجم الميزات بشكل كبير. مثال:

الترميز المتغير الفئوي Categorical Variable Encoding

ترميز المتغيرات الفئوية هو عملية تحويل المتغيرات الفئوية (المتغيرات التي يمكن أن تأخذ

عددًا محدودًا وثابتاً من القيم) إلى تنسيق رقمي يمكن استخدامه لخوارزميات التعلم الآلي. يعد

ذلك ضروريًا لأن معظم نماذج التعلم الآلي تتطلب بيانات إدخال رقمية. هناك عدة طرق

لتشفيير المتغيرات الفئوية، مع وجود أسلوبين شائعين هما الترميز السريع وترميز الأسماء.

فيما يلي تعريف مختصر لكل منها مع مثال لكيفية تنفيذها في بايثون باستخدام مكتبة البايدا:

الترميز المتغير الفئوي Categorical Variable Encoding

ترميز واحد ساخن : One-Hot Encoding

يقوم التشفير الأحادي بتحويل المتغيرات الفئوية إلى متجهات ثنائية حيث يتم تمثيل كل فئة بميزة ثنائية (0 أو 1).

يؤدي هذا إلى إنشاء ميزات ثنائية جديدة، واحدة لكل فئة فريدة، ومناسبة للمتغيرات الفئوية الاسمية (فئات ليس لها ترتيب متصل).

```
import pandas as pd
# Sample dataset with a categorical variable
data = {'Category': ['A', 'B', 'C', 'A', 'B', 'A', 'C']}
df = pd.DataFrame(data)
# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(df['Category'], prefix='Category')
print("One-Hot Encoded Data:")
print(one_hot_encoded)
```

One-Hot Encoded Data:

	Category_A	Category_B	Category_C
0	True	False	False
1	False	True	False
2	False	False	True
3	True	False	False
4	False	True	False
5	True	False	False
6	False	False	True

الترميز المتغير الفئوي Categorical Variable Encoding

```
from sklearn.preprocessing import LabelEncoder
# Sample dataset with a categorical variable
data = {'Category': ['A', 'B', 'C', 'A', 'B']}
df = pd.DataFrame(data)
# Perform label encoding
label_encoder = LabelEncoder()
df['Category_LabelEncoded'] = label_encoder.fit_transform(df['Category'])
print("Label Encoded Data:")
print(df)
```

Label Encoded Data:

	Category	Category_LabelEncoded
0	A	0
1	B	1
2	C	2
3	A	0
4	B	1

ترميز التسمية : Label Encoding
يقوم ترميز التسمية بتعيين عدد صحيح فريد لكل فئة في متغير فئوي. وهذا مناسب للمتغيرات الفئوية الترتيبية (فئات ذات ترتيب ذي معنى). مثال:

في الأمثلة المذكورة أعلاه، نستخدم دالة `pandas.get_dummies` من `scikit-learn` لتشفيir التسمية. تساعد هذه التقنيات في تحويل المتغيرات الفئوية إلى تنسيق مناسب لخوارزميات التعلم الآلي مع الحفاظ على الخصائص الكامنة في البيانات الفئوية.

معالجة البيانات المنحرفة Handling skewed data

تعد معالجة البيانات المنحرفة جانبًا مهمًا من المعالجة المسبقة للبيانات، خاصة بالنسبة لخوارزميات التعلم الآلي التي تفترض توزيع البيانات بشكل طبيعي. تشير البيانات المنحرفة إلى البيانات التي يكون فيها توزيع القيم غير متماثل وله ذيل طويل على جانب واحد.

تنتهك البيانات المنحرفة هذه الافتراضات، وقد تسبب العديد من المشاكل:

- انحياز النموذج: ستكون تنبؤات النموذج منحازة نحو القيم الأكثر تكرارًا (الذيل)، مما يضعف أدائها على القيم الأقل تكرارًا.
- ضعف الأداء: قد يؤدي إلى انخفاض الدقة في مجموعات التحقق والاختبار.
- الحساسية لقيم المتطرفة: تتأثر العديد من المقاييس الإحصائية (مثل المتوسط والتباين) بشدة بالقيم المتطرفة في التوزيع المنحرف.

هناك العديد من التقنيات للتعامل مع البيانات المنحرفة، بما في ذلك التحويل اللوغاريتمي وتحويل Box-Cox.

فيما يلي شرح موجز لكل منها مع مثال لكيفية تنفيذها في بايثون باستخدام مكتبة `scipy`

مثال

مجموعه بياناتنا النموذجية لنأخذ مجموعة بيانات صغيرة من القيم ذات ميل متعمد: [1, 1, 2, 5, 10, 20, 50]. لاحظ كيف تتجمع القيم في الطرف السفلي (1, 1, 2, 5)، مع بعض القيم الأكبر التي تكون ذيلاً طويلاً [50, 20, 10]. هذا الميل مائل لليمين. سنطبق كل تحويل على هذه المجموعة من القيم.

$\log(1 + x)$

Original (x)	Formula	Transformed Value (approx.)
--------------	---------	-----------------------------

1	$\log(1 + 1) = \log(2)$	0.693
---	-------------------------	-------

1	$\log(1 + 1) = \log(2)$	0.693
---	-------------------------	-------

2	$\log(1 + 2) = \log(3)$	1.099
---	-------------------------	-------

5	$\log(1 + 5) = \log(6)$	1.792
---	-------------------------	-------

10	$\log(1 + 10) = \log(11)$	2.398
----	---------------------------	-------

20	$\log(1 + 20) = \log(21)$	3.045
----	---------------------------	-------

50	$\log(1 + 50) = \log(51)$	3.932
----	---------------------------	-------

معالجة البيانات المنحرفة Handling skewed data

التحويل اللوغاريتمي :Logarithmic Transformation

يتضمن التحويل اللوغاريتمي تطبيق دالة اللوغاريتم الطبيعي على البيانات، مما يمكن أن يساعد في تقليل حجم القيم الكبيرة وجعل التوزيع أكثر تناسقاً. تكون هذه التقنية مفيدة عندما تكون البيانات منحرفة إلى اليمين. مثال:

```
# جامعة الجزيرة - اب - اليمن
# برمجة متقدمة - لغة البايثون
import numpy as np

# Sample dataset with skewed data
skewed_data = np.array([1, 10, 100, 1000])

# Apply logarithmic transformation
log_transformed_data = np.log(skewed_data)

print("Log-Transformed Data:")
print(log_transformed_data)
```

Log-Transformed Data:

```
[0.          2.30258509 4.60517019 6.90775528]
```

Example

دعونا نفكر في مثال واقعي يتضمن بيانات المبيعات لمتجر بيع بالتجزئة. لنفترض أن لدينا مجموعة بيانات تحتوي على عدد المنتجات المباعة كل يوم على مدار فترة زمنية. قد تبدو بيانات المبيعات كما يلي:

Day	Products Sold
1	10
2	20
3	30
...	...
30	300

في مجموعة البيانات هذه، يمثل عمود "المنتجات المباعة" عدد المنتجات المباعة كل يوم. وبما أن عدد المنتجات المباعة يمكن أن يختلف بشكل كبير من يوم لآخر، فقد تعرض البيانات نطاقاً واسعاً من القيم. لتحليل هذه البيانات، قد نرغب في قياس عمود "المنتجات المباعة" باستخدام تحويل لوغاريتمي. يمكن أن يكون التحويل اللوغاريتمي مفيداً عندما تمتد البيانات إلى عدة أوامر من حيث الحجم، ونريد ضغط المقياس لتصور البيانات أو تحليلها بشكل أفضل. دعونا نطبق تحويل اللوغاريتم الطبيعي (قاعدة السجل e) على عمود "المنتجات المباعة":

Example

Day	Products Sold	Log(Products Sold)
1	10	2.3026
2	20	2.9957
3	30	3.4012
...
30	300	5.7038

بعد تطبيق التحول اللوغاريتمي: يحتوي العمود "سجل (المنتجات المباعة)" على اللوغاريتم الطبيعي لقيم "المنتجات المباعة".

يؤدي التحويل اللوغاريتمي إلى ضغط حجم البيانات، خاصة عند التعامل مع أعداد كبيرة. يتم تكبير القيم الأصغر، بينما يتم ضغط القيم الأكبر، مما ينتج عنه نطاق أكثر قابلية للإدارة من القيم للتحليل. يمكن أن يكون هذا التحويل مفيداً بشكل خاص لتصور البيانات على مقياس لوغاريتمي، وتحديد الاتجاهات أو الأنماط، وتسهيل تفسير الاختلافات النسبية بين القيم، خاصة عند التعامل مع البيانات التي تظهر نمواً أسيّاً أو اضمحلالاً.

Square Root

2. Square Root Transformation

The Square Root Transform is $\text{sqrt}(x)$. It's less aggressive than the log transform.

Calculation:

Original (x)	Formula	Transformed Value (approx.)
1	$\text{sqrt}(1)$	1.000
1	$\text{sqrt}(1)$	1.000
2	$\text{sqrt}(2)$	1.414
5	$\text{sqrt}(5)$	2.236
10	$\text{sqrt}(10)$	3.162
20	$\text{sqrt}(20)$	4.472
50	$\text{sqrt}(50)$	7.071

Box-Cox Transformation

3. Box-Cox Transformation

The Box-Cox Transform is more complex. It finds a parameter, λ (lambda), that makes the data as normal as possible. The formula is:

For a value x and parameter λ :

$$(x^\lambda - 1) / \lambda \text{ if } \lambda \neq 0$$

$$\log(x) \text{ if } \lambda == 0$$

How to Interpret this:

The algorithm determined that the optimal λ is approximately **0.1235**. Since this is not 0, we use the first formula: $(x^\lambda - 1) / \lambda$.

Let's calculate the value for $x=50$ manually to verify:

$$\lambda = 0.1235$$

$$(50^{0.1235} - 1) / 0.1235$$

$$50^{0.1235} \approx 1.176 \text{ (Calculate this as np.power(50, 0.1235))}$$

$$1.176 - 1 = 0.176$$

$$0.176 / 0.1235 \approx 9.528$$

Example

Optimal lambda (λ) value: 0.1235

	Original	Log_Transformed	Sqrt_Transformed	BoxCox_Transformed
0	1	0.693	1.000	0.000
1	1	0.693	1.000	0.000
2	2	1.099	1.414	0.822
3	5	1.792	2.236	2.207
4	10	2.398	3.162	3.783
5	20	3.045	4.472	5.893
6	50	3.932	7.071	9.528

معالجة البيانات المنحرفة Handling skewed data

```
from scipy import stats  
  
# Sample dataset with skewed data  
skewed_data = np.array([1, 10, 100, 1000])  
  
# Apply Box-Cox transformation  
boxcox_transformed_data, lambda_value = stats.boxcox(skewed_data)  
  
print("Box-Cox Transformed Data:")  
print(boxcox_transformed_data)  
print("Lambda value:", lambda_value)
```

Box-Cox Transformed Data:

[0. 2.3025851 4.60517022 6.90775535]

Lambda value: 3.034325234395205e-09

تحويل بوكس-كوكس :Box-Cox Transformation

هو عبارة عن مجموعة من تحويلات الطاقة التي يمكنها تثبيت تباين البيانات وجعلها موزعة بشكل أكثر طبيعية. ويطلب تقدير معلمة (لامدا) لتحديد التحول المطبق على البيانات.

في الأمثلة أعلاه، نستخدم الدالة `np.log` لتحويل اللوغاريتمي والدالة `stats.boxcox` تساعد هذه التقنيات في معالجة الانحراف في البيانات وإعدادها لخوارزميات التعلم الآلي التي تفترض وجود بيانات موزعة بشكل طبيعي.

Example

في مجموعة البيانات هذه، يمثل عمود "السعر (\$)" سعر كل منزل بالدولار. يمكن أن تختلف أسعار المنازل بشكل كبير، وقد لا يتم توزيع الأسعار بشكل طبيعي. لتحسين توزيع بيانات الأسعار وجعلها أكثر ملاءمة لبعض التحليلات الإحصائية أو خوارزميات التعلم الآلي، يمكننا تطبيق تحويل Box-Cox. تحويل بوكس-كوكس هو عبارة عن عائلة من تحويلات القوة التي تتضمن التحويل اللوغاريتمي كحالة خاصة.

House ID	Size (sq ft)	Bedrooms	Bathrooms	Price (\$)	Transformed Price
1	1500	3	2	200,000	6.216
2	2000	4	3	300,000	6.906
3	1200	2	2	180,000	6.073

لتطبيق تحويل Box-Cox على عمود "السعر (\$)":

- حساب λ : يتطلب تحويل Box-Cox تقدير المعلمة، λ ، التي تحدد التحويل المطبق. يتم تقدير هذه المعلمة لتعظيم احتمالية توزيع البيانات بشكل طبيعي. ويمكن تقديرها باستخدام الأساليب الإحصائية.
- تطبيق التحويل: بمجرد تقدير قيمة λ ، نطبق تحويل Box-Cox على عمود "السعر (\$)" باستخدام الصيغة:

Example

حيث يمثل السعر الأصلي للمنزل. بعد تطبيق تحويل Box-Cox، سيتم تحويل عمود "السعر (\\$)" وفقاً لقيمة المقدرة لـ λ يمكن أن يساعد هذا التحول في جعل توزيع الأسعار أكثر تناسقاً وأقرب إلى التوزيع الطبيعي، مما يجعله أكثر ملاءمة لبعض التحليلات الإحصائية أو تقنيات

$$y_i^{(\lambda)} = \begin{cases} \frac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y_i) & \text{if } \lambda = 0 \end{cases}$$

يوجد في قلب تحويل Box Cox، λ ، والذي يتراوح من -5 إلى 5. يتمأخذ جميع قيم λ في الاعتبار ويتم تحديد القيمة المثلثى لبياناتك؛ "القيمة المثلثى" هي التي تؤدي إلى أفضل تقرير لمنحنى التوزيع الطبيعي.

معالجة البيانات التقديرية والمترددة Binning or discretization

أو التقديرية : Binning

هي تقنية معالجة مسبقة للبيانات تستخدم لتحويل البيانات الرقمية المستمرة إلى فترات أو صناديق منفصلة.

يمكن أن يكون ذلك مفيداً لتقليل تعقيد البيانات، والتعامل مع القيم المتطرفة، والتقاط العلاقات غير الخطية.

يتضمن Binning تقسيم نطاق القيم المستمرة إلى فترات ثم تعيين كل قيمة للفاصل الزمني أو السلة المقابلة لها.

فيما يلي شرح مختصر لعملية binning بالإضافة إلى مثال لكيفية تنفيذها في لغة Python باستخدام مكتبة pandas

How to Handle Noisy Data?

- **Binning**

- First sort data and partition into (equal-frequency) bins
- Then one can smooth by:
 1. Bin means,
 2. Smooth by bin median,
 3. Smooth by bin boundaries.

بینینج

أولاً، قم بفرز البيانات وتقسيمها إلى صناديق (متساوية التردد).

ثم يمكن للمرء أن يجانس بواسطة المعنى،

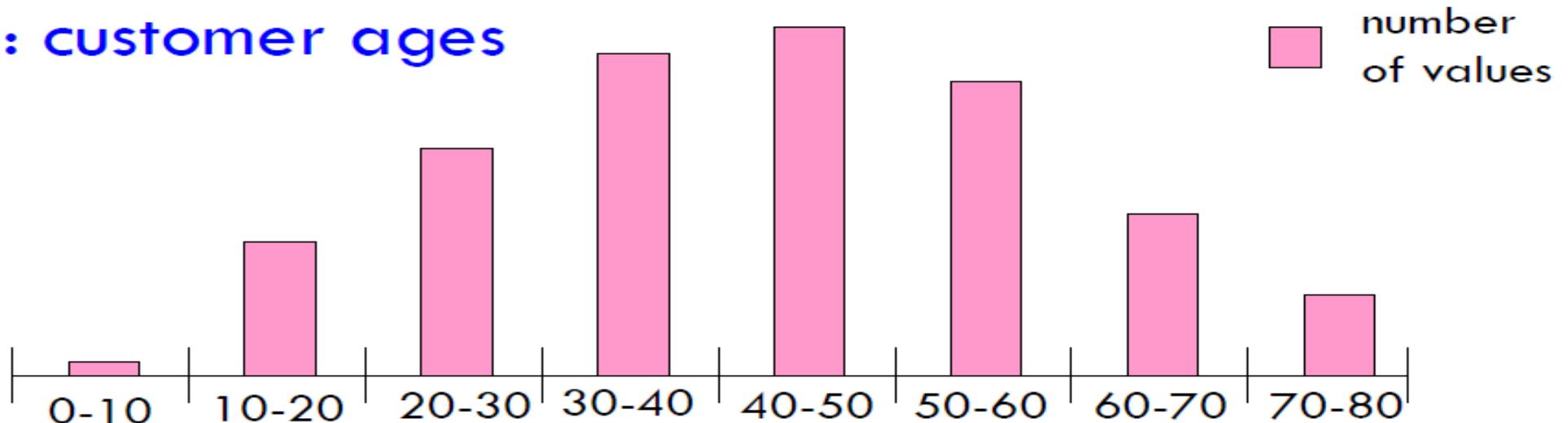
تجانس البيانات بواسطة الوسيط،

تجانس باستخدام الحدود، الخ.

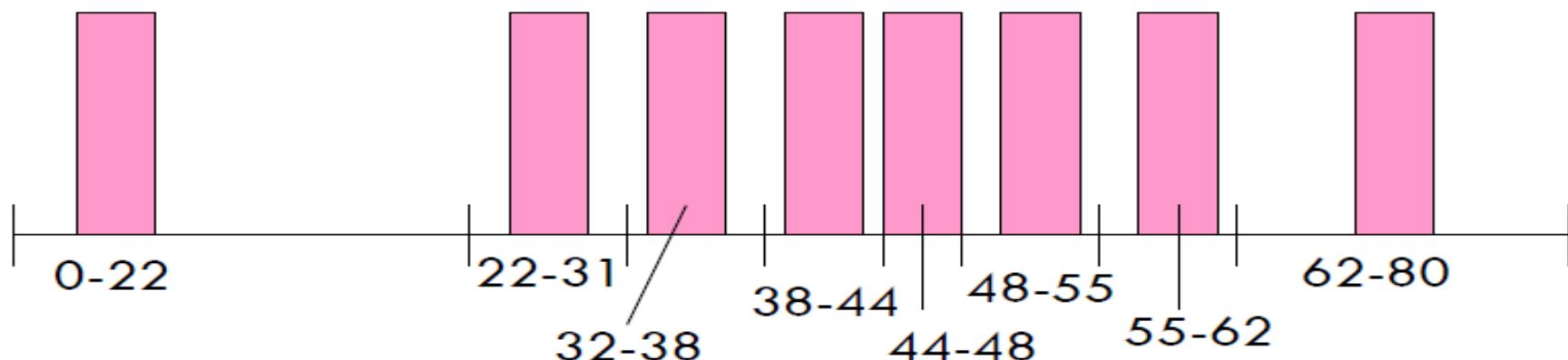
Simple Methods: Binning

Example: customer ages

Equi-width
binning:



Equi-width
binning:



Data Quality: Handle Noise(Binning)

Example: Sorted price values 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

- * Partition into three (equi-depth) bins
 - Bin 1: 4, 8, 9, 15
 - Bin 2: 21, 21, 24, 25
 - Bin 3: 26, 28, 29, 34
- * Smoothing by bin means
 - Bin 1: 9, 9, 9, 9
 - Bin 2: 23, 23, 23, 23
 - Bin 3: 29, 29, 29, 29
- * Smoothing by bin boundaries
 - Bin 1: 4, 4, 4, 15
 - Bin 2: 21, 21, 25, 25
 - Bin 3: 26, 26, 26, 34

معالجة البيانات المنحرفة Handling skewed data

```
import pandas as pd
# Sample dataset with continuous numerical data
data = {'Age': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70]}
# Create a DataFrame
df = pd.DataFrame(data)
# Define bin edges (boundaries) to divide the range of ages into intervals
bin_edges = [20, 40, 60, 80]
# Define bin labels to assign to each interval
bin_labels = ['Young', 'Middle-aged', 'Senior']
# Use the `cut` function to perform binning
df['Age_Group'] = pd.cut(df['Age'], bins=bin_edges, labels=bin_labels)
# Display the DataFrame with binning results
print(df)
```

	Age	Age_Group
0	25	Young
1	30	Young
2	35	Young
3	40	Young
4	45	Middle-aged
5	50	Middle-aged
6	55	Middle-aged
7	60	Middle-aged
8	65	Senior
9	70	Senior

يمكن أن يساعد التجميع أو التمييز في تبسيط تمثيل البيانات، وتقليل تأثير القيم المتطرفة، والتقط الأنماط التي قد لا تكون واضحة مع البيانات الرقمية المستمرة.

لنفترض أن لدينا مجموعة بيانات تحتوي على معلومات حول عمر الأفراد. نريد دمج هذا المتغير العدي المستمر في فئات عمرية منفصلة. في هذا المثال، قمنا بتقسيم نطاق الأعمار إلى ثلات فترات:

(40-20)، [60-60)، و[80-80). تقوم الدالة pd.cut بذلك بتعيين كل قيمة عمرية للفاصل الزمني المقابل لها بناءً على حواجز الصندوق المحددة. أخيراً، أضفنا عموداً جديداً DataFrame إلى Age_Group التجميع.

تجحيم الميزة Feature Scaling

تجحيم الميزة Feature Scaling

يعد قياس الميزة أحد أساليب المعالجة المسقبة المستخدمة لتوحيد أو تطبيع نطاق المتغيرات أو الميزة المستقلة في مجموعة البيانات. يعد هذا أمراً مهماً للعديد من خوارزميات التعلم الآلي الحساسة لمقياس ميزة الإدخال، مثل الخوارزميات support vector gradient descent-based algorithms، وأجهزة المتجهات الداعمة على النسب المترادج .k-nearest neighbors k machines.

فيما يلي شرح موجز لتجحيم الميزة بالإضافة إلى مثال لكيفية تنفيذها في Python باستخدام مكتبة scikit-Learn: مثال: لنفترض أن لدينا مجموعة بيانات تحتوي على خصائصتين: العمر (بالسنوات) والدخل (بالدولار). نريد توسيع نطاق هذه الميزة إلى نطاق مماثل باستخدام تقنيات توسيع الميزة.

تجحيم الميزة Feature Scaling

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
# Sample dataset with features
data = {'Age': [25, 30, 35, 40, 45],
        'Income': [50000, 60000, 70000, 80000, 90000]}
# Create a DataFrame
df = pd.DataFrame(data)
# Initialize scalers
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()
# Perform feature scaling
scaled_data_standard = standard_scaler.fit_transform(df)
scaled_data_minmax = minmax_scaler.fit_transform(df)
# Convert scaled data back to DataFrame for visualization
df_scaled_standard = pd.DataFrame(scaled_data_standard, columns=df.columns)
df_scaled_minmax = pd.DataFrame(scaled_data_minmax, columns=df.columns)
# Display scaled data
print("Standard Scaled Data:")
print(df_scaled_standard)
print("\nMinMax Scaled Data:")
print(df_scaled_minmax)
```

Standard Scaled Data:

	Age	Income
0	-1.414214	-1.414214
1	-0.707107	-0.707107
2	0.000000	0.000000
3	0.707107	0.707107
4	1.414214	1.414214

MinMax Scaled Data:

	Age	Income
0	0.00	0.00
1	0.25	0.25
2	0.50	0.50
3	0.75	0.75
4	1.00	1.00

تجحيم الميزة Feature Scaling

In this example, we use two common feature scaling techniques:

- **Standard Scaling (Z-score normalization):**

$$z = \frac{x - \mu}{\sigma}$$

This technique scales the features to have a mean of 0 and a standard deviation of 1. It transforms the data such that it has a standard normal distribution.

- **Min-Max Scaling:**

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This technique scales the features to a specified range, usually between 0 and 1. It linearly transforms the data to fit within this range.

Both techniques help ensure that the features are on a similar scale, which can improve the performance and convergence of machine learning algorithms.

معالجة البيانات التقديرية والمترددة Data reduction

يشير تقليل البيانات في مرحلة ما قبل المعالجة إلى عملية تقليل حجم أو أبعاد مجموعة البيانات مع الاحتفاظ بخصائصها المهمة وتقليل فقدان المعلومات. ويتم ذلك عادةً لتحسين كفاءة تحليل البيانات، وتقليل التعقيد الحسابي، وتحسين أداء خوارزميات التعلم الآلي. هناك عدة تقنيات لتقليل البيانات في المعالجة المسبقه:

• اختيار الميزة : Feature Selection

يتضمن اختيار الميزة تحديد مجموعة فرعية من الميزات (المتغيرات) الأكثر صلة من مجموعة البيانات الأصلية مع التخلص من الميزات غير ذات الصلة أو الزائدة عن الحاجة. ويساعد ذلك على تقليل أبعاد البيانات والتركيز على أهم العوامل المؤثرة على المتغير المستهدف.

• استخراج الميزات :Feature Extraction

يعمل استخراج الميزات على تحويل الميزات الأصلية إلى مساحة ذات أبعاد أقل عن طريق إنشاء ميزات جديدة تلقط المعلومات الأساسية في البيانات. تُستخدم تقنيات مثل تحليل المكون الرئيسي (PCA) والتحليل التمييزي الخطي (LDA) بشكل شائع لاستخراج الميزات.

معالجة البيانات التقديرية والمترددة Data reduction

٠ أخذ العينات :Sampling

تتضمن تقييات أخذ العينات اختيار مجموعة فرعية تمثيلية من مجموعة البيانات الأصلية لتحليلها. يمكن أن يشمل ذلك تقييات مثل أخذ العينات العشوائية، أو أخذ العينات الطبقية، أو أخذ العينات على أساس المجموعات. يساعد أخذ العينات على تقليل حجم البيانات مع الحفاظ على خصائصها الإحصائية.

٠ تحويل البيانات :Data Transformation

تعمل تقييات تحويل البيانات على تغيير حجم البيانات أو توزيعها لتقليل تعقيدها. يمكن أن يشمل ذلك تقييات مثل التطبيع أو التوحيد القياسي أو التحويل اللوغاريتمي أو تحويل Box-Cox.

٠ تخفيض الأبعاد :Dimensionality Reduction

تهدف تقييات تقليل الأبعاد إلى تقليل عدد الأبعاد (المعالم) في مجموعة البيانات مع الحفاظ على أكبر قدر ممكن من المعلومات. يتضمن ذلك تقييات مثل PCA، وتضمين الجوار العشوائي الموزع ((t-SNE)، وأجهزة التشفير التلقائي.

معالجة البيانات غير المتوازنة

Handling Imbalanced Data

تشير معالجة البيانات غير المتوازنة في مرحلة ما قبل المعالجة إلى التقنيات المستخدمة لمعالجة المواقف التي لا يتم فيها تمثيل الفئات في مشكلة التصنيف بشكل متساوٍ في مجموعة البيانات. تحدث البيانات غير المتوازنة عندما تكون فئة واحدة (فئة الأقلية) ممثلة تمثيلاً ناقصاً بشكل ملحوظ مقارنة بالفئات الأخرى (فئة أو فئات الأغلبية). يمكن أن يؤدي هذا الاختلال في التوازن إلى نماذج متحيزه تفضل فئة الأغلبية ويكون أداؤها سيئاً في التنبؤ بطبقة الأقلية. هناك عدة تقنيات للتعامل مع البيانات غير المتوازنة في المعالجة المسبقه:

تقنيات إعادة أخذ العينات :Resampling Techniques

➤ الإفراط في أخذ العينات :Over-sampling

تضمن هذه التقنية زيادة عدد الحالات في فئة الأقلية عن طريق تكرار الحالات الموجودة بشكل عشوائي أو إنشاء عينات تركيبية.

➤ نقص أخذ العينات :Under-sampling

يؤدي أخذ العينات الناقص إلى تقليل عدد المثلثات في فئة الأغلبية عن طريق إزالة المثلثات بشكل عشوائي حتى يتم تحقيق توزيع أكثر توازناً.

➤ تقنية الإفراط في أخذ العينات للأقليات الاصطناعية :Synthetic Minority Over-sampling Technique (SMOTE)

يقوم SMOTE بإنشاء مثيلات تركيبية لفئة الأقلية عن طريق الاستيفاء بين المثلثات الموجودة.

معالجة البيانات غير المتوازنة

Handling Imbalanced Data

أساليب المجموعة :Ensemble Methods

✓ وزن الفئة :Class Weighting

تعديل أوزان الفئة في النموذج لإعطاء أهمية أكبر لفئة الأقلية أثناء التدريب. يمكن تحقيق ذلك عن طريق تعين أوزان أعلى لمثيلات فئة الأقلية أو أوزان أقل لمثيلات فئة الأغلبية.

► التعبئة :Bagging

استخدام تقنيات مثل الغابات العشوائية التي تجمع التنبؤات من نماذج متعددة تم تدريبيها على مجموعات فرعية متوازنة من البيانات.

► التعزيز :Boosting

تعمل تقنيات مثل AdaBoost و Gradient Boosting على تدريب النماذج بشكل تسلسلي، مما يعطي وزناً أكبر لمثيلات التي تم تصنيفها بشكل خاطئ من فئة الأقلية.

✓ التعلم الحساس من حيث التكلفة :Cost-sensitive Learning

تعديل دالة التكلفة للنموذج لمعاقبة التصنيف الخاطئ لفئة الأقلية بشكل أكبر من التصنيف الخاطئ لفئة الأغلبية.

✓ تعزيز البيانات :Data Augmentation

إنشاء مثيلات إضافية لفئة الأقلية من خلال تقنيات مثل التدوير أو الترجمة أو إضافة ضوضاء إلى المثيلات الموجودة.

معالجة البيانات غير المتوازنة

Handling Imbalanced Data

✓ الكشف عن الشذوذ :Anomaly Detection

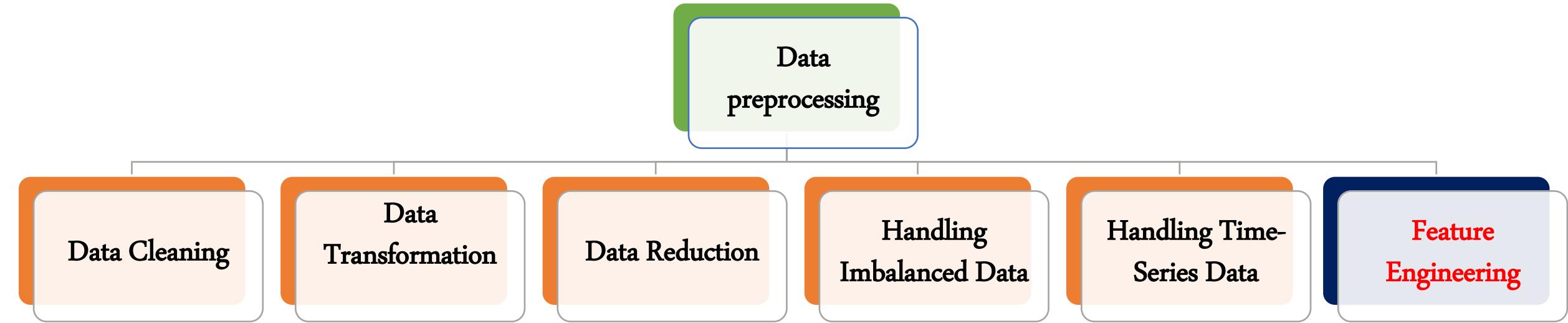
تعامل مع فئة الأقلية على أنها حالات شاذة واستخدم تقنيات الكشف عن الحالات الشاذة لتحديد هذه الحالات والتركيز عليها.

▷ مجموعة من تقنيات أخذ العينات المختلفة :Ensemble of Different Sampling Techniques

الجمع بين تقنيات أخذ العينات المتعددة أو استراتيجيات إعادة أخذ العينات لمعالجة البيانات غير المتوازنة بشكل أكثر فعالية.

يعتمد اختيار التقنية على عوامل مثل مدى خطورة عدم توازن الفئة، وحجم مجموعة البيانات، والموارد الحسابية المتوفرة، والأداء المطلوب للنموذج. من الشائع تجربة تقنيات متعددة واختيار الأسلوب الذي يحقق أفضل النتائج لمشكلة التصنيف المحددة المطروحة.

Data preprocessing



هندسة الميزات (Feature Engineering)

تعد هندسة الميزات عملية مهمة في تحليل البيانات وتنقيتها. وتهدف هندسة الميزات إلى استخلاص المعلومات الهامة والقيمة من البيانات وتحويلها إلى مجموعة من الميزات (المتغيرات) المناسبة التي يمكن استخدامها في تطبيق تقنيات التعلم الآلي والتنبؤ.

هندسة الميزات تحصل بسبب انه يوجد لدينا:

ميزات كثيرات الحدود :Polynomial Features

ينشئ مجموعات متعددة الحدود من الميزات لالتقاط العلاقات غير الخطية بين المتغيرات.

شروط التعامل :Interaction Terms

ينشئ ميزات جديدة عن طريق التفاعلات (المنتجات) بين الميزات الموجودة.

عملية الميزات تشمل عدة خطوات، وفيما يلي نظرة عامة على هذه الخطوات:

استكشاف البيانات : Exploratory Data

يتم تحليل واستكشاف البيانات المتاحة لفهمها بشكل أفضل. يتضمن ذلك فحص توزيع البيانات، القيم المفقودة، القيم الزائفة، الانحرافات الكبيرة، وأي تحليل إحصائي آخر يساعد في فهم البيانات.

تحويل المتغيرات : Transformation Variable

قد يتطلب تحليل البيانات تحويل المتغيرات إلى توزيعات أخرى تكون أكثر ملائمة لاستخدامها في نماذج التعلم الآلي. يشمل ذلك تحويل المتغيرات الرقمية باستخدام عمليات مثل التحويل اللوغاريتمي.

إنشاء المتغيرات الجديدة : Feature Creation

يمكن إنشاء متغيرات جديدة من البيانات المتاحة بناءً على الخبرة المجالية أو المفاهيم الإحصائية. قد تشمل هذه المتغيرات المشتقة مثل: معادلات، نسب، مجاميع، فروق، تصنيفات، أو أي معلومة تساعد في تحسين قدرة النموذج على التنبؤ

عملية هندسة الميزات تشمل عدة خطوات، وفيما يلي نظرة عامة على هذه الخطوات:

تحديد المتغيرات الهامة : Feature Selection

في حالة وجود عدد كبير من المتغيرات، يمكن أن يكون من المفيد تحديد المتغيرات الأكثر أهمية لتحسين أداء النموذج وتقليل الزمن والموارد المستهلكة. يمكن استخدام تقنيات التحليل الإحصائي أو تقنيات تقليل الأبعاد (مثل تحليل المكونات الرئيسية) لتحديد المتغيرات الهامة.

تجهيز المتغيرات : Feature Scaling

تطلب بعض تقنيات التعلم الآلي أن تكون المتغيرات على نفس المقياس أو نطاق محدد. يتم استخدام تقنيات مثل تحويل المدى أو تحويل القياس المعياري لتجهيز المتغيرات وجعلها جاهزة للاستخدام. يجب أن يتم تنفيذ هذه الخطوات بعناية وفقاً للبيانات المتوفرة وطبيعة المشكلة المحددة. يمكن أيضاً تكرار هذه الخطوات وتجربة مجموعة متنوعة من التحويرات والإبداع في إنشاء الميزات لتحسين أداء النموذج.

هناك معلومات إضافية لشرح هندسة الميزات أكثر توضيحاً:

1-استخلاص السمات من البيانات النصية:

عند التعامل مع البيانات النصية، يمكن استخلاص المعلومات الهامة من النصوص وتحويلها إلى ميزات قابلة للتحاليل. يمكن استخدام تقنيات مثل تحويل النص إلى متجهات الكلمات (Word2Vec) أو استخراج الميزات اللغوية (مثل عدد الكلمات، التركيب الجملي) لتمثيل البيانات النصية بشكل رقمي.

2-تحويل الوقت والتاريخ:

في حالة وجود متغيرات تاريخية أو زمنية، يمكن تحويلها إلى ميزات أكثر فائدة. يمكن استخلاص مكونات الوقت مثل السنة والشهر واليوم والفصل الزمني، أو يمكن حساب المدة بين تواريخ مختلفة كميزة. يمكن أيضاً تحويل الوقت إلى دورة يومية بواسطة حساب الساعة والدقائق، مما يسمح بالتقاط الأنماط اليومية في البيانات

الهدف من هندسة الميزات

الهدف من هندسة الميزات هو اشتقاق ميزات جديدة من ميزات موجودة وذلك لتوفير معلومات إضافية للنموذج .

ملاحظة ...

هندسة الميزات عملية استكشافية وإبداعية، تعتمد على المعرفة المجالية والتحليل الإحصائي للاستفادة القصوى من البيانات المتاحة. من خلال تحسين الميزات وتحويلها ل تكون مناسبة لتطبيقات التعلم الآلي، يمكن تحسين أداء النماذج وزيادة دقتها وقدرتها على التنبؤ بشكل أفضل.

الكود البرمجي بلغة البايثون

```
import pandas as pd
```

تم استدعاء مكتبة ال **pandas** للتعامل مع القيم والمصفوفات

```
from sklearn.preprocessing import StandardScaler
```

تم الاستدعاء من مكتبة **sklearn** أداة لتحويل الميزات **StandardScaler** (تحديد المقاييس القياسي للميزات) وذلك لتحقيق التوزيع القياسي للبيانات

```
from sklearn.feature_selection import SelectKBest
```

```
from sklearn.feature_selection import chi2
```

تم استخدام المكتبة **SelectKBest** لتحديد افضل الميزات بواسطة اختبار الكاي المربع **chi2** لاختيار افضل 10 ميزات

```
from sklearn.decomposition import PCA
```

قمنا باستدعاء خوارزمية ال **PCA** لتقليل الابعاد

تابع الكود البرمجي

```
Data=pd.read_csv('dataset.csv')
```

قراءة ملف ال csv وإسناده للمتغير Data

```
X = data.drop('target', axis=1)
```

```
y = data['target']
```

اختيار المميزات المستهدفة

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

تحويل المميزات الرقمية باستخدام المقياس القياسي

```
k_best = SelectKBest(score_func=chi2, k=10)
```

```
X_best = k_best.fit_transform(X_scaled, y)
```

```
pca = PCA(n_components=3)
```

```
X_pca = pca.fit_transform(X_best)
```

```
print(X_pca)
```

تحويل البيانات النصية Text Data Transformation

يتضمن تحويل البيانات النصية تحويل بيانات النص الخام إلى تنسيق مناسب لمهام معالجة اللغة الطبيعية NLP أو نماذج التعلم الآلي التي تعمل على البيانات النصية.

غالبًا ما يتضمن هذا التحويل خطوات مثل تنظيف النص، والترميز، والتوجيه، وهندسة الميزات. فيما يلي شرح مختصر لكل خطوة بالإضافة إلى مثال لكيفية تفيذها في بايثون باستخدام مكتبات nltk و scikit-learn:

مثال :

لنفترض أن لدينا مجموعة بيانات تتضمن مراجعات نصية للمنتجات. نريد تحويل بيانات النص الخام هذه إلى تنسيق مناسب لتحليل المشاعر.

تحويل البيانات النصية Text Data Transformation

يتضمن تحويل البيانات النصية تحويل بيانات النص الخام إلى تنسيق مناسب لمهام معالجة اللغة الطبيعية NLP أو

نماذج التعلم الآلي التي تعمل على البيانات النصية.

غالبًا ما يتضمن هذا التحويل خطوات مثل تنظيف النص، والترميز، والتوجيه، وهندسة الميزات. فيما يلي شرح

مختصر لكل خطوة بالإضافة إلى مثال لكيفية تفيذها في بايثون باستخدام مكتبات scikit-learn و nltk.