Questions on measure of central tendency

1) ***Business Problem: A retail store wants to analyze the sales data of a particular product category to understand the typical sales performance and make strategic decisions.***

   Data:
   Let's consider the weekly sales data (in units) for the past month for a specific product category:

   Week 1: 50 units
   Week 2: 60 units
   Week 3: 55 units
   Week 4: 70 units

   Question:
   1. Mean: What is the average weekly sales of the product category?

   2. Median: What is the typical or central sales value for the product category?

   3. Mode: Are there any recurring or most frequently occurring sales figures for the product category?

   By answering these questions using the mean, median, and mode, the retail store can gain insights into the sales performance of the product category, identify any patterns or outliers, and make informed decisions regarding stock management, marketing strategies, and product placement.

2) ***Business Problem: A restaurant wants to analyze the waiting times of its customers to understand the typical waiting experience and improve service efficiency.***

   Data:
   Let's consider the waiting times (in minutes) for the past 20 customers:

   15, 10, 20, 25, 15, 10, 30, 20, 15, 10,
   10, 25, 15, 20, 20, 15, 10, 10, 20, 25

   Question:
   1. Mean: What is the average waiting time for customers at the restaurant?

   2. Median: What is the typical or central waiting time experienced by customers?

3. Mode: Are there any recurring or most frequently occurring waiting times for customers?

By answering these questions using the mean, median, and mode, the restaurant can gain insights into the average waiting time, identify any common or peak waiting periods, and make informed decisions to optimize the customer service process, such as adjusting staffing levels, streamlining operations, or implementing strategies to reduce waiting times.

3) ***Business Problem: A car rental company wants to analyze the rental durations of its customers to understand the typical rental period and optimize its pricing and fleet management strategies.***

Data:
Let's consider the rental durations (in days) for a sample of 50 customers:

3, 2, 5, 4, 7, 2, 3, 3, 1, 6,
4, 2, 3, 5, 2, 4, 2, 1, 3, 5,
6, 3, 2, 1, 4, 2, 4, 5, 3, 2,
7, 2, 3, 4, 5, 1, 6, 2, 4, 3,
5, 3, 2, 4, 2, 6, 3, 2, 4, 5

Question:
1. Mean: What is the average rental duration for customers at the car rental company?

2. Median: What is the typical or central rental duration experienced by customers?

3. Mode: Are there any recurring or most frequently occurring rental durations for customers?

By answering these questions using the mean, median, and mode, the car rental company can gain insights into the average rental duration, understand the most common rental periods, and make informed decisions regarding pricing, fleet size, and availability. Additionally, this analysis can help the company optimize resource allocation, plan for peak demand periods, and enhance customer satisfaction by aligning service offerings with customers' typical rental needs.

Questions on measure of dispersion

1) ***Problem: A manufacturing company wants to analyze the production output of a specific machine to understand the variability or spread in its performance.***

   Data:
   Let's consider the number of units produced per hour by the machine for a sample of 10 working days:

   Day 1: 120 units
   Day 2: 110 units
   Day 3: 130 units
   Day 4: 115 units
   Day 5: 125 units
   Day 6: 105 units
   Day 7: 135 units
   Day 8: 115 units
   Day 9: 125 units
   Day 10: 140 units

   Question:
   1. Range: What is the range of the production output for the machine?

   2. Variance: What is the variance of the production output for the machine?

   3. Standard Deviation: What is the standard deviation of the production output for the machine?

   By answering these questions using different measures of dispersion, the manufacturing company can gain insights into the variability in the machine's production output. This information can help identify any fluctuations, assess the consistency of performance, and make informed decisions regarding quality control, scheduling, and resource allocation.

2) ***Problem: A retail store wants to analyze the sales of a specific product to understand the variability in daily sales and assess its inventory management.***

   Data:
   Let's consider the daily sales (in dollars) for the past 30 days:

   $500, $700, $400, $600, $550, $750, $650, $500, $600, $550,

$800, $450, $700, $550, $600, $400, $650, $500, $750, $550,
$700, $600, $500, $800, $550, $650, $400, $600, $750, $550

Questions:
1. Range: What is the range of the daily sales?

2. Variance: What is the variance of the daily sales?

3. Standard Deviation: What is the standard deviation of the daily sales?

By answering these questions using different measures of dispersion, the retail store can gain insights into the variability in daily sales, assess the consistency of demand, and make informed decisions regarding inventory stocking levels, sales forecasting, and pricing strategies.

3) ***Problem: An e-commerce platform wants to analyze the delivery times of its shipments to understand the variability in order fulfillment and optimize its logistics operations.***

Data:
Let's consider the delivery times (in days) for a sample of 50 shipments:

3, 5, 2, 4, 6, 2, 3, 4, 2, 5,
7, 2, 3, 4, 2, 4, 2, 3, 5, 6,
3, 2, 1, 4, 2, 4, 5, 3, 2, 7,
2, 3, 4, 5, 1, 6, 2, 4, 3, 5,
3, 2, 4, 2, 6, 3, 2, 4, 5, 3

Questions:
1. Range: What is the range of the delivery times?

2. Variance: What is the variance of the delivery times?

3. Standard Deviation: What is the standard deviation of the delivery times?

By answering these questions using different measures of dispersion, the e-commerce platform can gain insights into the variability in delivery times, identify any bottlenecks in the logistics process, and make informed decisions regarding shipment tracking, customer expectations, and service level agreements.

4) ***Problem : A company wants to analyze the monthly revenue generated by one of its products to understand its performance and variability.***

Data:
Let's consider the monthly revenue (in thousands of dollars) for the past 12 months:

$120, $150, $110, $135, $125, $140, $130, $155, $115, $145, $135, $130

Questions:
1. Measure of Central Tendency: What is the average monthly revenue for the product?
2. Measure of Dispersion: What is the range of monthly revenue for the product?

By answering these questions, the company can gain insights into the average revenue generated by the product and understand the range or variability in its monthly revenue, which can help with financial planning, forecasting, and evaluating the product's performance.

5) ***Problem : A survey was conducted to gather feedback from customers regarding their satisfaction with a particular service on a scale of 1 to 10.***

Data:
Let's consider the satisfaction ratings from 50 customers:

8, 7, 9, 6, 7, 8, 9, 8, 7, 6,
8, 9, 7, 8, 7, 6, 8, 9, 6, 7,
8, 9, 7, 6, 7, 8, 9, 8, 7, 6,
9, 8, 7, 6, 8, 9, 7, 8, 7, 6,
9, 8, 7, 6, 7, 8, 9, 8, 7, 6

Questions:
1. Measure of Central Tendency: What is the average satisfaction rating?
2. Measure of Dispersion: What is the standard deviation of the satisfaction ratings?

By answering these questions, the company can gain insights into the average satisfaction rating of customers and understand the spread or variability in their ratings. This information can help identify areas for improvement, evaluate customer perception, and make informed decisions to enhance the service quality.

6) ***Problem :A company wants to analyze the customer wait times at its call center to assess the efficiency of its customer service operations.***

Data:
Let's consider the wait times (in minutes) for a sample of 100 randomly selected customer calls:

10, 15, 12, 18, 20, 25, 8, 14, 16, 22,
9, 17, 11, 13, 19, 23, 21, 16, 24, 27,
13, 10, 18, 16, 12, 14, 19, 21, 11, 17,
15, 20, 26, 13, 12, 14, 22, 19, 16, 11,
25, 18, 16, 13, 21, 20, 15, 12, 19, 17,
14, 16, 23, 18, 15, 11, 19, 22, 17, 12,
16, 14, 18, 20, 25, 13, 11, 22, 19, 17,
15, 16, 13, 14, 18, 20, 19, 21, 17, 12,
15, 13, 16, 14, 22, 21, 19, 18, 16, 11,
17, 14, 12, 20, 23, 19, 15, 16, 13, 18

Questions:
1. Measure of Central Tendency: What is the average wait time for customers at the call center?
2. Measure of Dispersion: What is the range of wait times for customers at the call center?
3. Measure of Dispersion: What is the standard deviation of the wait times for customers at the call center?

By answering these questions, the company can gain insights into the average wait time experienced by customers, assess the variability or spread in the wait times, and make informed decisions regarding staffing levels, call center efficiency, and customer satisfaction.

7) ***Problem : A transportation company wants to analyze the fuel efficiency of its vehicle fleet to identify any variations across different vehicle models.***

Data:
Let's consider the fuel efficiency (in miles per gallon, mpg) for a sample of 50 vehicles:

Model A: 30, 32, 33, 28, 31, 30, 29, 30, 32, 31,
Model B: 25, 27, 26, 23, 28, 24, 26, 25, 27, 28,
Model C: 22, 23, 20, 25, 21, 24, 23, 22, 25, 24,
Model D: 18, 17, 19, 20, 21, 18, 19, 17, 20, 19,
Model E: 35, 36, 34, 35, 33, 34, 32, 33, 36, 34

Questions:
1. Measure of Central Tendency: What is the average fuel efficiency for each vehicle model?
2. Measure of Dispersion: What is the range of fuel efficiency for each vehicle model?
3. Measure of Dispersion: What is the variance of the fuel efficiency for each vehicle model?

By answering these questions, the transportation company can

gain insights into the average fuel efficiency of different vehicle models, understand the variations or spread in the fuel efficiency, and make informed decisions regarding fleet management, vehicle selection, and fuel consumption optimization.

More Statistics Questions

8) ***Problem : A company wants to analyze the ages of its employees to understand the age distribution and demographics within the organization.***

Data:
Let's consider the ages of 100 employees:

28, 32, 35, 40, 42, 28, 33, 38, 30, 41,
37, 31, 34, 29, 36, 43, 39, 27, 35, 31,
39, 45, 29, 33, 37, 40, 36, 29, 31, 38,
35, 44, 32, 39, 36, 30, 33, 28, 41, 35,
31, 37, 42, 29, 34, 40, 31, 33, 38, 36,
39, 27, 35, 30, 43, 29, 32, 36, 31, 40,
38, 44, 37, 33, 35, 41, 30, 31, 39, 28,
45, 29, 33, 38, 34, 32, 35, 31, 40, 36,
39, 27, 35, 30, 43, 29, 32, 36, 31, 40,
38, 44, 37, 33, 35, 41, 30, 31, 39, 28

Questions:
1. Frequency Distribution: Create a frequency distribution table for the ages of the employees.
2. Mode: What is the mode (most common age) among the employees?
3. Median: What is the median age of the employees?
4. Range: What is the range of ages among the employees?

By answering these questions using frequency distribution and other measures, the company can gain insights into the age distribution of its workforce, identify the most common age group, understand the central tendency, and assess the spread of ages. This information can be useful for workforce planning, diversity initiatives, and understanding the generational dynamics within the organization.

9) ***Problem :A retail store wants to analyze the purchase amounts made by customers to understand their spending habits.***

Data:
Let's consider the purchase amounts (in dollars) for a sample of 50 customers:

56, 40, 28, 73, 52, 61, 35, 40, 47, 65,
52, 44, 38, 60, 56, 40, 36, 49, 68, 57,
52, 63, 41, 48, 55, 42, 39, 58, 62, 49,
59, 45, 47, 51, 65, 41, 48, 55, 42, 39,
58, 62, 49, 59, 45, 47, 51, 65, 43, 58

Questions:
1. Frequency Distribution: Create a frequency distribution table for the purchase amounts.
2. Mode: What is the mode (most common purchase amount) among the customers?
3. Median: What is the median purchase amount among the customers?
4. Interquartile Range: What is the interquartile range of the purchase amounts?

By answering these questions using frequency distribution and other measures, the retail store can gain insights into the spending habits of its customers, identify the most common purchase amount

10) Problem : A manufacturing company wants to analyze the defect rates of its production line to identify the frequency of different types of defects.

Data:
Let's consider the types of defects and their corresponding frequencies observed in a sample of 200 products:

Defect Type: A, B, C, D, E, F, G
Frequency: 30, 40, 20, 10, 45, 25, 30

Questions:
1. Bar Chart: Create a bar chart to visualize the frequency of different defect types.
2. Most Common Defect: Which defect type has the highest frequency?
3. Histogram: Create a histogram to represent the defect frequencies.

By answering these questions using a bar chart and histogram, the manufacturing company can visually understand the distribution of defect types, identify the most common defect, and prioritize quality control efforts to address the prevalent issues.

11) Problem : A survey was conducted to gather feedback from customers about their satisfaction levels with a specific service on a scale of 1 to 5.

Data:
Let's consider the satisfaction ratings from 100 customers:

Ratings: 4, 5, 3, 4, 4, 3, 2, 5, 4, 3,
5, 4, 2, 3, 4, 5, 3, 4, 5, 3,
4, 3, 2, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4,
3, 4, 5, 4, 2, 3, 4, 5, 3, 4,
5, 4, 3, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4,
3, 4, 5, 4, 2, 3, 4, 5, 3, 4,
5, 4, 3, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4

Questions:
1. Histogram: Create a histogram to visualize the distribution of satisfaction ratings.
2. Mode: Which satisfaction rating has the highest frequency?
3. Bar Chart: Create a bar chart to display the frequency of each satisfaction rating.

By answering these questions using a histogram and bar chart, the organization can gain insights into the distribution of satisfaction ratings, identify the most common satisfaction level, and assess overall customer satisfaction.

12) Problem : A company wants to analyze the monthly sales figures of its products to understand the sales distribution across different price ranges.

Data:
Let's consider the monthly sales figures (in thousands of dollars) for a sample of 50 products:

Sales: 35, 28, 32, 45, 38, 29, 42, 30, 36, 41,
47, 31, 39, 43, 37, 30, 34, 39, 28, 33,
36, 40, 42, 29, 31, 45, 38, 33, 41, 35,
37,

34, 46, 30, 39, 43, 28, 32, 36, 29,
31, 37, 40, 42, 33, 39, 28, 35, 38, 43

Questions:
1. Histogram: Create a histogram to visualize the sales distribution across different price ranges.
2. Measure of Central Tendency: What is the average monthly sales figure?
3. Bar Chart: Create a bar chart to display the frequency of sales in different price ranges.

By answering these questions using a histogram and bar chart, the company can understand the distribution of sales figures, determine the average sales performance, and identify the price ranges where sales are concentrated or lacking.

13) Problem : A study was conducted to analyze the response times of a website for different user locations.

Data:
Let's consider the response times (in milliseconds) for a sample of 200 user requests:

Response Times: 125, 148, 137, 120, 135, 132, 145, 122, 130, 141,
118, 125, 132, 136, 128, 123, 132, 138, 126, 129,
136, 127, 130, 122, 125, 133, 140, 126, 133, 135,
130, 134, 141, 119, 125, 131, 136, 128, 124, 132,
136, 127, 130, 122, 125, 133, 140, 126, 133, 135,
130, 134, 141, 119, 125, 131, 136, 128, 124, 132,
136, 127, 130, 122, 125, 133, 140, 126, 133, 135,
130, 134, 141, 119, 125, 131, 136, 128, 124, 132,
136, 127, 130, 122, 125, 133, 140, 126, 133, 135,
130, 134, 141, 119, 125, 131, 136, 128, 124, 132

Questions:
1. Histogram: Create a histogram to visualize the distribution of response times.
2. Measure of Central Tendency: What is the median response time?
3. Bar Chart: Create a bar chart to display the frequency of response times within different ranges.

By answering these questions using a histogram and bar chart, the study can gain insights into the distribution of response times, understand the typical response time experienced by users, and assess the performance of the website.

14) Problem : A company wants to analyze the sales performance of its products across different regions.

Data:
Let's consider the sales figures (in thousands of dollars) for a sample of 50 products in three regions:

Region 1: 45, 35, 40, 38, 42, 37, 39, 43, 44, 41,
Region 2: 32, 28, 30, 34, 33, 35, 31, 29, 36, 37,
Region 3: 40, 39, 42, 41, 38, 43, 45, 44, 41, 37

Questions:

1. Bar Chart: Create a bar chart to compare the sales figures across the three regions.
2. Measure of Central Tendency: What is the average sales figure for each region?
3. Measure of Dispersion

: What is the range of sales figures in each region?

By answering these questions using a bar chart and measures of central tendency and dispersion, the company can compare the sales performance across different regions, identify the average sales figures, and understand the variability in sales within each region. This information can be used for regional sales analysis, resource allocation, and decision-making processes.

Questions on Measure of Skewness and Kurtosis

1) ***Question : A company wants to analyze the monthly returns of its investment portfolio to understand the distribution and risk associated with the returns.***

Data:
Let's consider the monthly returns (%) for the portfolio over a one-year period:

Returns: -2.5, 1.3, -0.8, -1.9, 2.1, 0.5, -1.2, 1.8, -0.5, 2.3,
-0.7, 1.2, -1.5, -0.3, 2.6, 1.1, -1.7, 0.9, -1.4, 0.3,
1.9, -1.1, -0.4, 2.2, -0.9, 1.6, -0.6, -1.3, 2.4, 0.7,
-1.8, 1.5, -0.2, -2.1, 2.8, 0.8, -1.6, 1.4, -0.1, 2.5,
-1.0, 1.7, -0.9, -2.0, 2.7, 0.6, -1.4, 1.1, -0.3, 2.0

Questions:
1. Skewness: Calculate the skewness of the monthly returns.
2. Kurtosis: Calculate the kurtosis of the monthly returns.
3. Interpretation: Based on the skewness and kurtosis values, what can be said about the distribution of returns?

By answering these questions using measures of skewness and kurtosis, the company can understand the shape and symmetry of the return distribution, assess the level of risk and potential outliers, and make informed decisions regarding portfolio management and risk mitigation strategies.

2) ***Question : A research study wants to analyze the income distribution of a population to understand the level of income inequality.***

Data:
Let's consider the monthly incomes (in thousands of dollars) of a sample of 100 individuals:

Incomes: 2.5, 4.8, 3.2, 2.1, 4.5, 2.9, 2.3, 3.1, 4.2, 3.9,
2.8, 4.1, 2.6, 2.4, 4.7, 3.3, 2.7, 3.0, 4.3, 3.7,
2.2, 3.6, 4.0, 2.7, 3.8, 3.5, 3.2, 4.4, 2.0, 3.4,
3.1, 2.9, 4.6, 3.3, 2.5, 4.9, 2.8, 3.0, 4.2, 3.9,
2.8, 4.1, 2.6, 2.4, 4.7, 3.3, 2.7, 3.0, 4.3, 3.7,
2.2, 3.6, 4.0, 2.7, 3.8, 3.5, 3.2, 4.4,

2.0, 3.4,
3.1, 2.9, 4.6, 3.3, 2.5, 4.9, 2.8, 3.0, 4.2, 3.9,
2.8, 4.1, 2.6, 2.4, 4.7, 3.3, 2.7, 3.0, 4.3, 3.7,
2.2, 3.6, 4.0, 2.7, 3.8, 3.5, 3.2, 4.4, 2.0, 3.4,
3.1, 2.9, 4.6, 3.3, 2.5, 4.9

Questions:
1. Skewness: Calculate the skewness of the income distribution.
2. Kurtosis: Calculate the kurtosis of the income distribution.
3. Interpretation: Based on the skewness and kurtosis values, what can be inferred about the income inequality?

By answering these questions using measures of skewness and kurtosis, the research study can assess the level of income inequality, determine the shape of the income distribution, and make informed policy recommendations to address income disparities.

3) ***Question : A survey was conducted to analyze the satisfaction ratings of customers on a scale of 1 to 5 for a specific product.***

Data:
Let's consider the satisfaction ratings from 200 customers:

Ratings: 4, 5, 3, 4, 4, 3, 2, 5, 4, 3,
5, 4, 2, 3, 4, 5, 3, 4, 5, 3,
4, 3, 2, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4,
3, 4, 5, 4, 2, 3, 4, 5, 3, 4,
5, 4, 3, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4,
3, 4, 5, 4, 2, 3, 4, 5, 3, 4,
5, 4, 3, 4, 5, 3, 4, 5, 4, 3,
3, 4, 5, 2, 3, 4, 4, 3, 5, 4

Questions:
1. Skewness: Calculate the skewness of the satisfaction ratings.
2. Kurtosis: Calculate the kurtosis of the satisfaction ratings.

3. Interpretation: Based on the skewness and kurtosis values, what can be inferred about the satisfaction ratings distribution?

By answering these questions using measures of skewness and kurtosis, the survey can assess the skewness and peakedness of the satisfaction ratings, determine if the ratings are skewed towards positive or negative evaluations, and understand the distribution characteristics of customer satisfaction.

4) ***Question : A study wants to analyze the distribution of house prices in a specific city to understand the market trends.***

Data:
Let's consider the house prices (in thousands of dollars) for

a sample of 150 houses:

House Prices: 280, 350, 310, 270, 390, 320, 290, 340, 310, 380,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290,
270, 350, 300, 330, 370, 310, 280, 320, 350, 290

Questions:
1. Skewness: Calculate the skewness of the house price distribution.
2. Kurtosis: Calculate the kurtosis of the house price distribution.
3. Interpretation: Based on the skewness and kurtosis values, what can be inferred about the distribution of house prices?

By answering these questions using measures of skewness and kurtosis, the study can assess the symmetry and peakedness of the house price distribution, identify any outliers or extreme values, and gain insights into the market trends and pricing dynamics.

5) ***Question : A company wants to analyze the waiting times of customers at a service center to improve operational efficiency.***

Data:
Let's consider the waiting times (in minutes) for a sample of 100 customers:

Waiting Times: 12, 18, 15, 22, 20, 14, 16, 21, 19, 17,
22, 19, 13, 16, 21, 22, 17, 19, 22, 18,
14, 20, 19, 17, 22, 18, 15, 21, 20, 16,
12, 18, 15, 22, 20, 14, 16, 21, 19, 17,
22, 19, 13, 16, 21, 22, 17, 19, 22, 18,
14, 20, 19, 17, 22, 18, 15, 21, 20, 16,
12, 18, 15, 22, 20, 14, 16, 21, 19, 17,
22, 19, 13, 16, 21, 22, 17, 19, 22, 18,
14, 20, 19, 17, 22, 18, 15, 21, 20, 16,
12, 18, 15, 22, 20, 14, 16, 21, 19, 17

Questions:
1. Skewness: Calculate the skewness of the waiting time distribution.
2. Kurtosis

: Calculate the kurtosis of the waiting time distribution.
3. Interpretation: Based on the skewness and kurtosis values, what can be inferred about the waiting time distribution?

By answering these questions using measures of skewness and kurtosis, the company can assess the symmetry and tail behavior of the waiting time distribution, identify any patterns or anomalies in customer waiting times, and make improvements to streamline the service process and enhance customer satisfaction.

Questions on Percentile and Quartiles

1) ***Question : A company wants to analyze the salary distribution of its employees to determine the income levels at different percentiles.***

Data:
Let's consider the monthly salaries (in thousands of dollars) of a sample of 200 employees:

Salaries: 40, 45, 50, 55, 60, 62, 65, 68, 70, 72,
75, 78, 80, 82, 85, 88, 90, 92, 95, 100,
105, 110, 115, 120, 125, 130, 135, 140, 145, 150,
155, 160, 165, 170, 175, 180, 185, 190, 195, 200,
205, 210, 215, 220, 225, 230, 235, 240, 245, 250,
255, 260, 265, 270, 275, 280, 285, 290, 295, 300,
305, 310, 315, 320, 325, 330, 335, 340, 345, 350,
355, 360, 365, 370, 375, 380, 385, 390, 395, 400,
405, 410, 415, 420, 425, 430, 435, 440, 445, 450,
455, 460, 465, 470, 475, 480, 485, 490, 495, 500

Questions:
1. Quartiles: Calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the salary distribution.
2. Percentiles: Calculate the 10th percentile, 25th percentile, 75th percentile, and 90th percentile of the salary distribution.
3. Interpretation: Based on the quartiles and percentiles, what can be inferred about the income distribution of the employees?

By answering these questions using quartiles and percentiles, the company can understand the income levels at different points in the distribution, identify the median salary and the spread of salaries, and make informed decisions related to compensation, employee benefits, and salary structures.

2) ***Question : A research study wants to analyze the weight distribution of a sample of individuals to assess their health and body composition.***

Data:
Let's consider the weights (in kilograms) of a sample of 100 individuals:

Weights: 55, 60, 62, 65, 68, 70, 72, 75, 78, 80,
82, 85, 88, 90, 92, 95, 100, 105, 110, 115,
120, 125, 130, 135, 140, 145, 150, 155, 160, 165,
170, 175, 180, 185, 190, 195, 200, 205, 210, 215,
220, 225, 230, 235, 240, 245, 250, 255, 260, 265,
270, 275, 280, 285, 290, 295, 300, 305, 310, 315,
320, 325, 330, 335, 340, 345, 350, 355, 360, 365,
370, 375,

380, 385, 390, 395, 400, 405, 410, 415,
420, 425, 430, 435, 440, 445, 450, 455, 460, 465,
470, 475, 480, 485, 490, 495, 500, 505, 510, 515

Questions:
1. Quartiles: Calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the weight distribution.
2. Percentiles: Calculate the 15th percentile, 50th percentile, and 85th percentile of the weight distribution.
3. Interpretation: Based on the quartiles and percentiles, what can be inferred about the weight distribution of the individuals?

By answering these questions using quartiles and percentiles, the research study can understand the weight distribution and identify the weight ranges at different percentiles, such as underweight, normal weight, overweight, and obese categories. This information

can be used for evaluating health risks, designing appropriate interventions, and providing personalized recommendations for weight management.

3) ***Question : A retail store wants to analyze the distribution of customer purchase amounts to identify their spending patterns.***

Data:
Let's consider the purchase amounts (in dollars) of a sample of 150 customers:

Purchase Amounts: 20, 25, 30, 35, 40, 45, 50, 55, 60, 65,
70, 75, 80, 85, 90, 95, 100, 105, 110, 115,
120, 125, 130, 135, 140, 145, 150, 155, 160, 165,
170, 175, 180, 185, 190, 195, 200, 205, 210, 215,
220, 225, 230, 235, 240, 245, 250, 255, 260, 265,
270, 275, 280, 285, 290, 295, 300, 305, 310, 315,
320, 325, 330, 335, 340, 345, 350, 355, 360, 365,
370, 375, 380, 385, 390, 395, 400, 405, 410, 415,
420, 425, 430, 435, 440, 445, 450, 455, 460, 465,
470, 475, 480, 485, 490, 495, 500, 505, 510, 515,
520, 525, 530, 535, 540, 545, 550, 555, 560, 565

Questions:
1. Quartiles: Calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the purchase amount distribution.
2. Percentiles: Calculate the 20th percentile, 40th percentile, and 80th percentile of the purchase amount distribution.
3. Interpretation: Based on the quartiles and percentiles, what can be inferred about the spending patterns of the customers?

By answering these questions using quartiles and percentiles, the retail store can understand the distribution of purchase amounts, identify the spending ranges at different percentiles, analyze customer segments based on their spending behavior, and tailor marketing strategies to target specific customer groups.

4) ***Question : A study wants to analyze the distribution of commute times of employees to determine the average time spent traveling to work.***

Data:
Let's consider the commute times (in minutes) of a sample of 250 employees:

Commute Times: 15, 20, 25, 30, 35, 40, 45, 50, 55, 60,
65, 70, 75, 80, 85, 90, 95, 100, 105, 110,
115, 120, 125, 130, 135, 140, 145, 150, 155, 160,
165, 170, 175, 180, 185, 190, 195, 200, 205, 210,

215, 220, 225, 230, 235, 240, 245, 250, 255, 260,
265, 270, 275, 280, 285, 290, 295, 300, 305, 310,
315, 320, 325, 330, 335, 340, 345, 350, 355, 360,
365, 370, 375, 380, 385, 390, 395, 400, 405, 410,
415, 420, 425, 430, 435, 440, 445, 450, 455, 460,
465, 470, 475, 480, 485, 490, 495, 500, 505, 510,
515, 520, 525, 530, 535, 540, 545, 550, 555, 560,
565, 570, 575, 580, 585, 590, 595, 600, 605, 610

Questions:
1. Quartiles: Calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the commute time distribution.
2. Percentiles: Calculate the 30th percentile, 50th percentile, and 70th percentile of the commute time distribution.
3. Interpretation: Based on the quartiles and percentiles, what can be inferred about the average commute time of the employees?

By answering these questions using quartiles and percentiles, the study can determine the typical commute times, understand the spread of commute times, identify any outliers or extreme values, and provide insights for transportation planning, scheduling, and employee well-being initiatives.

5) ***Question : A manufacturing company wants to analyze the defect rates in its production process to evaluate product quality.***

Data:
Let's consider the defect rates (in percentage) for a sample of 300 products:

Defect Rates: 0.5, 1.0, 0.2, 0.7, 0.3, 0.9, 1.2, 0.6, 0.4, 1.1,
0.8, 0.5, 0.3, 0.6, 1.0, 0.4, 0.5, 0.7, 0.9, 1.3,
0.8, 0.6, 0.4, 0.7, 0.9, 0.5, 0.2, 1.0, 0.8, 0.3,
0.6, 0.4, 0.7, 0.9, 1.2, 0.8, 0.3, 0.6, 0.5, 0.4,
0.7, 0.9, 1.1, 0.3, 1.4, 0,9, 0.6, 0.2, 1.5, 1.0
0.6, 0.4, 0.7, 1.0, 0.8, 0.3, 0.5, 0.8, 0.6, 0.3, 0.9
0.4, 0.7, 0.9, 1.0, 0.8, 0.3, 0.5, 0.6, 0.4, 0.7,
0.9, 1.1, 0.8, 0.3, 0.5, 0.6, 0.4, 0.7, 0.9, 1.0,
0.8, 0.3, 0.5, 0.6, 0.4, 0.7, 0.9, 1.1, 0.8, 0.3,
0.5, 0.6, 0.4, 0.7, 0.9, 1.0, 0.8, 0.3, 0.5, 0.6,
0.4, 0.7, 0.9, 1.1, 0.8, 0.3, 0.5, 0.6, 0.4, 0.7,
0.9, 1.0, 0.8, 0.3, 0.5, 0.6, 0.4, 0.7, 0.9, 1.1

Questions:
1. Quartiles: Calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the defect rate distribution.

2. Percentiles: Calculate the 25th percentile, 50th percentile, and 75th percentile of the defect rate distribution.
3. Interpretation: Based on the quartiles and percentiles, what can be inferred about the quality of the products?

By answering these questions using quartiles and percentiles, the manufacturing company can evaluate the defect rates, understand the spread of defects, identify any quality issues or deviations from standards, and take corrective actions to improve the production process and product quality.

Questions on Correlation and Covariance

1) ***Question : A marketing department wants to understand the relationship between advertising expenditure and sales revenue to assess the effectiveness of their advertising campaigns.***

   Data:
   Let's consider the monthly advertising expenditure (in thousands of dollars) and corresponding sales revenue (in thousands of dollars) for a sample of 12 months:

   Advertising Expenditure: 10, 12, 15, 18, 20, 22, 25, 28, 30, 32, 35, 38
   Sales Revenue: 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105

   Question:
   Calculate the correlation coefficient between advertising expenditure and sales revenue. Interpret the value of the correlation coefficient and explain the nature of the relationship between advertising expenditure and sales revenue.

   By analyzing the correlation coefficient, the marketing department can determine the strength and direction of the relationship between advertising expenditure and sales revenue. This information can help them make informed decisions about allocating their advertising budget and optimizing their marketing strategies.

2) ***Question : An investment analyst wants to assess the relationship between the stock prices of two companies to identify potential investment opportunities.***

   Data:
   Let's consider the daily closing prices (in dollars) of Company A and Company B for a sample of 20 trading days:

   Company A: 45, 47, 48, 50, 52, 53, 55, 56, 58, 60, 62, 64, 65, 67, 69, 70, 72, 74, 76, 77
   Company B: 52, 54, 55, 57, 59, 60, 61, 62, 64, 66, 67, 69, 71, 73, 74, 76, 78, 80, 82, 83

Question:
Calculate the covariance between the stock prices of Company A and Company B. Interpret the value of the covariance and explain the nature of the relationship between the two stocks.

By analyzing the covariance, the investment analyst can determine whether the stock prices of Company A and Company B move together (positive covariance) or in opposite directions (negative covariance). This information can assist in identifying potential investment opportunities and understanding the diversification benefits of combining these stocks in a portfolio.

3) ***Question : A researcher wants to examine the relationship between the hours spent studying and the exam scores of a group of students.***

Data:
Let's consider the number of hours spent studying and the corresponding exam scores for a sample of 30 students:

Hours Spent Studying: 10, 12, 15, 18, 20, 22, 25, 28, 30, 32, 35, 38, 40, 42, 45, 48, 50, 52, 55, 58, 60, 62, 65, 68, 70, 72, 75, 78, 80, 82
Exam Scores: 60, 65, 70, 75, 80, 82, 85, 88, 90, 92, 93, 95, 96, 97, 98, 99, 100, 102, 105, 106, 107, 108, 110, 112, 114, 115, 116, 118, 120, 122

Question:
Calculate the correlation coefficient between the hours spent studying and the exam scores. Interpret the value of the correlation coefficient and explain the nature of the relationship between studying hours and exam scores.

By analyzing the correlation coefficient, the researcher can determine the strength and direction of the relationship between studying hours and exam scores. This information can provide insights into the effectiveness of studying and help students and educators make informed decisions about study habits and academic performance.

Questions on discrete and continuous random variable

Discrete Random Variable:

1. Problem: A fair six-sided die is rolled 100 times. What is the probability of rolling exactly five 3's?
   Data: Number of rolls (n) = 100

2. Problem: In a deck of 52 playing cards, five cards are randomly drawn without replacement. What is the probability of getting two hearts?
   Data: Number of hearts in the deck (N) = 13, Number of cards drawn (n) = 5

3. Problem: A multiple-choice test consists of 10 questions, each with four possible answers. If a student randomly guesses on each question, what is the probability of getting at least 8 questions correct?
   Data: Number of questions (n) = 10, Number of possible answers per question (k) = 4

4. Problem: A bag contains 30 red balls, 20 blue balls, and 10 green balls. Three balls are drawn without replacement. What is the probability that all three balls are blue?
   Data: Number of blue balls in the bag (N) = 20, Number of balls drawn (n) = 3

5. Problem: In a football match, a player scores a goal with a 0.3 probability per shot. If the player takes 10 shots, what is the probability of scoring exactly three goals?
   Data: Number of shots (n) = 10, Probability of scoring per shot (p) = 0.3

Continuous Random Variable:

1. Problem: The heights of students in a class are normally distributed with a mean of 165 cm and a standard deviation of 10 cm. What is the probability that a randomly selected student is taller than 180 cm?
   Data: Mean height (μ) = 165 cm, Standard deviation (σ) = 10 cm, Height threshold (x) = 180 cm

2. Problem: The waiting times at a coffee shop are exponentially distributed with a mean of 5 minutes. What is the probability that a customer waits less than 3 minutes?
   Data: Mean waiting time (μ) = 5 minutes, Waiting time threshold (x) = 3 minutes

3. Problem: The lifetimes of a certain brand of light bulbs are normally distributed with a mean of 1000 hours and a standard deviation of 100 hours. What is the probability that a randomly selected light bulb lasts between 900 and 1100 hours?

Data: Mean lifetime (μ) = 1000 hours, Standard deviation (σ) = 100 hours, Lifetime range (lower limit x1, upper limit x2)

4. Problem: The weights of apples in a basket follow a uniform distribution between 100 grams and 200 grams. What is the probability that a randomly selected apple weighs between 150 and 170 grams?
Data: Weight range (lower limit x1, upper limit x2)

5. Problem: The time taken to complete a task is exponentially distributed with a mean of 20 minutes. What is the probability that the task is completed in less than 15 minutes?
Data: Mean time (μ) = 20 minutes, Time threshold (x) = 15 minutes

Questions on Discrete Distribution and Continuous Distribution

Discrete Distribution:

1. Problem: A company sells smartphones, and the number of defects per batch follows a Poisson distribution with a mean of 2 defects. What is the probability of having exactly 3 defects in a randomly selected batch?
Data: Mean number of defects (λ) = 2, Number of defects (x) = 3

Explanation: The problem involves a discrete distribution (Poisson) because we are dealing with the count of defects in a batch of smartphones. The Poisson distribution models the probability of a given number of events occurring within a fixed interval of time or space.

2. Problem: In a game, a player has a 0.3 probability of winning each round. If the player plays 10 rounds, what is the probability of winning exactly 3 rounds?
Data: Probability of winning (p) = 0.3, Number of rounds (n) = 10, Number of wins (x) = 3

Explanation: This problem also involves a discrete distribution (Binomial) because we are dealing with a fixed number of independent trials (rounds) with a probability of success (winning) in each trial. The Binomial distribution models the probability of achieving a certain number of successes in a fixed number of trials.

3. Problem: A six-sided fair die is rolled three times. What is the probability of obtaining at least one 6?
Data: Number of rolls (n) = 3

Explanation: Here, we have a discrete distribution (Geometric) since we are interested in the number of trials required to achieve the first success (rolling a 6) in a sequence of independent trials. The Geometric distribution models the probability of achieving the first success on a specific trial.

Continuous Distribution:

1. Problem: The weights of apples in a basket follow a normal distribution with a mean of 150 grams and a standard deviation of 10 grams. What is the probability that a randomly selected apple weighs between 140 and 160 grams?
Data: Mean weight (μ) = 150 grams, Standard deviation (σ) = 10 grams, Weight range (lower limit x1, upper limit x2)

Explanation: This problem involves a continuous distribution (Normal) since we are dealing with the weights of apples, which can take on any value within a range. The Normal distribution is commonly used to model continuous variables with a symmetric bell-shaped distribution.

2. Problem: The lifetimes of a certain brand of light bulbs are exponentially distributed with a mean of 1000 hours. What is the probability that a randomly selected light bulb lasts more than 900 hours?
Data: Mean lifetime (μ) = 1000 hours, Lifetime threshold (x) = 900 hours

Explanation: Here, we have a continuous distribution (Exponential) since we are interested in the time until an event (light bulb failure) occurs. The Exponential distribution models the probability of waiting a certain amount of time before the event happens.

Questions on Confidence Interval and Hypothesis Testings

Confidence Interval Problems:

1. Problem: A study is conducted to estimate the mean height of a population. A random sample of 100 individuals is selected, and their heights are measured. Calculate a 95% confidence interval for the population mean height, given that the sample mean height is 170 cm and the sample standard deviation is 8 cm.
   Data: Sample size (n) = 100, Sample mean ($\bar{x}$) = 170 cm, Sample standard deviation (s) = 8 cm, Confidence level = 95%

Explanation: In this problem, we use a sample to estimate the population mean height. By calculating a confidence interval, we provide a range of plausible values for the population mean. The 95% confidence level indicates that we are 95% confident that the true population mean height falls within the calculated interval.

2. Problem: A survey is conducted to estimate the proportion of people in a city who support a particular policy. A random sample of 500 individuals is surveyed, and 320 of them express support for the policy. Calculate a 90% confidence interval for the population proportion, given the sample proportion.
   Data: Sample size (n) = 500, Number of successes (x) = 320, Confidence level = 90%

Explanation: In this problem, we aim to estimate the population proportion based on the sample proportion. By constructing a confidence interval, we provide a range of plausible values for the population proportion. The 90% confidence level indicates that we are 90% confident that the true population proportion falls within the calculated interval.

Hypothesis Testing Problems:

3. Problem: A researcher wants to test whether a new teaching method improves student performance. A random sample of 50 students is divided into two groups: one group taught using the new method and the other using the traditional method. The average test scores of the two groups are compared. State the null and alternative hypotheses for this study.
   Data: Sample size (n) = 50, Test scores of the two groups

Explanation: In this problem, we are interested in comparing the means of two groups (new method vs. traditional method). The null hypothesis (H0) states that there is no

significant difference between the means, while the alternative hypothesis (Ha) suggests that there is a significant difference.

4. Problem: A manufacturing company claims that the average weight of its product is 500 grams. To test this claim, a random sample of 25 products is selected, and their weights are measured. The sample mean weight is found to be 510 grams with a sample standard deviation of 20 grams. Perform a hypothesis test to determine if there is evidence to support the company's claim.
Data: Sample size (n) = 25, Sample mean ($\bar{x}$) = 510 grams, Sample standard deviation (s) = 20 grams, Population mean (μ) = 500 grams

Explanation: In this problem, we are conducting a hypothesis test to assess whether the sample mean weight provides evidence to support the company's claim about the population mean weight. The null hypothesis (H0) assumes that the population mean weight is equal to the claimed value, while the alternative hypothesis (Ha) suggests otherwise.

Module (2 and 3)

1) Use the average function and calculate the average of all the three category of weight. (for this question use excel file named *average 1*).
2) The excel file named ***Average 3*** , the table below contains precipitation measurement as measured in the Rochester NY area last year and we sampled 3 days in each of the first three months of 2018. Complete all the question in the file given .
3) In excel file named ***Count 1***, The table below shows survey responses; the respondents could use any value for their answers. Answer all the questions using COUNT and COUNTA function.

4) In excel file named ***COUNT 2*** , The following table represents a bank statement of ExcelMaster company. Column E shows the total dollar value amount of each of the accounts. Answer all the questions using COUNT and COUNTA function.
5) In excel file named ***COUNT 3*** , Solve all the question by using formulas COUNT, COUNTA and COUNTBLANK:
6) In excel file named ***HLOOKUP***, Solve all the question using HLOOKUP only.
7) In excel file named ***IF 1*** , Table A contains names and their respective grades for Excel 101 Course. Complete column C using only IF formula.
8) In excel file named ***IF 2***, The following table is an extract from an accounting system that contains four journal entries. Check if column A's cells match column B's cell. if they match - return "match", otherwise return "no match".
9) In excel file named ***IF 3***, The table below contains details of high school students names and ages, use IF formula to complete columns D and E.If the student's age is 16 or above, he/she is eligible for a driver's license. Check if they are eligible or not. Answer in column D. If the student is younger than 18 years old he/she is a minor. Check whether the student is a minor or not. for Minor return "Minor" and non minor = "Adult" anwswer in column E.
10) In excel file named ***IF 4***, An A+ student gets 100% scholarship and non A+ gets 50% scholarship , The following table contains the names of students from 2024 class. Use IF function to calculate the scholarships' amounts each of them will get.
11) In excel file named ***Math 1***, Use the following guidelines to calculate the statements given the file.
12) In excel file named ***MAX MIN 1***, Use max, min and average formulas to answer all the following questions given in the file.
13) In the file named ***MAX MIN 2***, The following table contains details about the scores of 4 students in a driving theory test. If a student fails at least one test - she or he needs to retake the course. Use IF and MAX/MIN to check if a student passed the test.
14) In the file named ***MAX MIN 3,*** IF at least one student got 99 points or more in a test - the test considered easy, Use MAX and IF to create a logic that checks if the test was "Easy" or not.
15) In the file named ***Nested IF 1***, The school decided to use the following grade system:

    A. Grade higher or equal to 80 - Excellent
    B. Grade higher or equal to 60 but lower than 80 – Good
    C. Grade lower than 60 - Failed

    Complete all the task given in the file.

16) In the file named ***SUM 1***, The following table includes ABC company's revenue by month. The company's CFO asked you to use SUM formula to calculate the total revenue for the year.
17) In the file named ***SUM 2***, The following table represents daily costs by day for the first quarter of 2015. Calculate the total costs at the bottom of the table. Hint: to save time, use sum shortcuts.

18) In the file named ***SUM 3***, Find the number of residents for each of the following groups from the table below, complete all the question in the file.
19) In the file named ***SUMIF 1***, answer all the question given in the file.
20) In the file named ***SUMIF 2***, answer all the question given in the file based on table.
21) In the file named ***VLOOKUP APPROXIMATE MATCH***, Retrieve the GBP:USD exchange rate for the following dates using VLOOKUP function, from the table in columns G-H. In case there is no exchange rate for a certain date entry, return the the last known rate for that day.
22) In the file named ***VLOOKUP 1***, Below is a list of the employees who work in your company: Answer all the question given in the file using vlookup function.
23) In the file named ***VLOOKUP 2a***, According to the table , answer all the question given in the file using vlookup.
24) In Excel file ***first_exercise***, For a table of populations, change data types and make other changes in Power Query. Do the following things to make this table easier to read:
    a. Tell Power Query to use the first row as column headings.
    b. Delete the Source column (we don't need it).
    c. Change the data type of the Date column to Date.
    d. Change the data type of the Population column to Whole Number.
    e. Shorten the name of the Country column.
    f. And make other changes if needed to read data properly.
25) In Excel file ***first_exercise***, Import a population table using Power Query, then tidy up the data:
    a. In the **"question_3_power_query_file"** , all the steps are mentiond to do the above exercise, please look and follow the steps.
26) In Excel file ***first_exercise*** , Divide exchange rate and investment symbols into more parts by splitting and removing columns:
    a. In the **"question_4_power_query_file"** , all the steps are mentiond to do the above exercise, please look and follow the steps.
27) In Excel file ***first_exercise*** , How to unpivot data in Power Query, then use this as a basis for a pivot table:
    a. In the **"question_5_power_query_file"** , all the steps are mentiond to do the above exercise, please look and follow the steps.
28) In Excel file ***first_exercise***, Tidy up exchange rate and investment data in Power Query, splitting columns and replacing values:
    a. In the **"question_2_power_query_file"** , all the steps are mentiond to do the above exercise, please look and follow the steps.
29) In Excel file ***first_exercise*** , Create a query to import a table of tall buildings, create new columns and then pivot the data:
    a. In the **"question_6_power_query_file"** , all the steps are mentiond to do the above exercise, please look and follow the steps.
30) In Excel file ***first_exercise*** ,For a table of skyscrapers, perform lots of query transforms and then pivot the results:

a. In the ***"question_7_power_query_file"*** , all the steps are mentiond to do the above exercise, please look and follow the steps.

Module 4 (SQL)

Database Name: HR

1. Display all information in the tables EMP and DEPT.

2. Display only the hire date and employee name for each employee.

3. Display the ename concatenated with the job ID, separated by a comma and space, and name the column Employee and Title

4. Display the hire date, name and department number for all clerks.

5. Create a query to display all the data from the EMP table. Separate each column by a comma. Name the column THE OUTPUT

6. Display the names and salaries of all employees with a salary greater than 2000.

7. Display the names and dates of employees with the column headers "Name" and "Start Date"

8. Display the names and hire dates of all employees in the order they were hired.

9. Display the names and salaries of all employees in reverse salary order.

10. Display 'ename" and "deptno" who are all earned commission and display salary in reverse order.

11. Display the last name and job title of all employees who do not have a manager

12. Display the last name, job, and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to $2,500, $3,500, or $5,000

Database Name: HR

1) Display the maximum, minimum and average salary and commission earned.

2) Display the department number, total salary payout and total commission payout for each department.

3) Display the department number and number of employees in each department.

4) Display the department number and total salary of employees in each department.

5) Display the employee's name who doesn't earn a commission. Order the result set without using the column name

6) Display the employees name, department id and commission. If an Employee doesn't earn the commission, then display as 'No commission'. Name the columns appropriately

7) Display the employee's name, salary and commission multiplied by 2. If an Employee doesn't earn the commission, then display as 'No commission. Name the columns appropriately

8) Display the employee's name, department id who have the first name same as another employee in the same department

9) Display the sum of salaries of the employees working under each Manager.

10) Select the Managers name, the count of employees working under and the department ID of the manager.

11) Select the employee name, department id, and the salary. Group the result with the manager name and the employee last name should have second letter 'a!

12) Display the average of sum of the salaries and group the result with the department id. Order the result with the department id.

13) Select the maximum salary of each department along with the department id

14) Display the commission, if not null display 10% of salary, if null display a default value 1

Database Name: HR

1. Write a query that displays the employee's last names only from the string's 2-5th position with the first letter capitalized and all other letters lowercase, Give each column an appropriate label.

2. Write a query that displays the employee's first name and last name along with a " in between for e.g.: first name : Ram; last name : Kumar then Ram-Kumar. Also displays the month on which the employee has joined.

3. Write a query to display the employee's last name and if half of the salary is greater than ten thousand then increase the salary by 10% else by 11.5% along with the bonus amount of 1500 each. Provide each column an appropriate label.

4. Display the employee ID by Appending two zeros after 2nd digit and 'E' in the end, department id, salary and the manager name all in Upper case, if the Manager name consists of 'z' replace it with '$!

5. Write a query that displays the employee's last names with the first letter capitalized and all other letters lowercase, and the length of the names, for all employees whose name starts with J, A, or M. Give each column an appropriate label. Sort the results by the employees' last names

6. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with $. Label the column SALARY

7. Display the employee's name if it is a palindrome

8. Display First names of all employees with initcaps.

9. From LOCATIONS table, extract the word between first and second space from the STREET ADDRESS column.

10. Extract first letter from First Name column and append it with the Last Name. Also add "@systechusa.com" at the end. Name the column as e-mail address. All characters should be in lower case. Display this along with their First Name.

11. Display the names and job titles of all employees with the same job as Trenna.

12. Display the names and department name of all employees working in the same city as Trenna.

13. Display the name of the employee whose salary is the lowest.

14. Display the names of all employees except the lowest paid.

Database Name: HR

1. Write a query to display the last name, department number, department name for all employees.
2. Create a unique list of all jobs that are in department 4. Include the location of the department in the output.

3. Write a query to display the employee last name,department name,location id and city of all employees who earn commission.

4. Display the employee last name and department name of all employees who have an 'a' in their last name

5. Write a query to display the last name,job,department number and department name for all employees who work in ATLANTA.

6. Display the employee last name and employee number along with their manager's last name and manager number.

7. Display the employee last name and employee number along with their manager's last name and manager number (including the employees who have no manager).

8. Create a query that displays employees last name,department number,and all the employees who work in the same department as a given employee.

9. Create a query that displays the name,job,department name,salary,grade for all employees. Derive grade based on salary(>=50000=A, >=30000=B,<30000=C)

10. Display the names and hire date for all employees who were hired before their managers along withe their manager names and hire date. Label the columns as Employee name, emp_hire_date,manager name,man_hire_date

Database Name: AdventureWorks

1. Write a query to display employee numbers and employee name (first name, last name) of all the sales employees who received an amount of 2000 in bonus.

2. Fetch address details of employees belonging to the state CA. If address is null, provide default value N/A.

3. Write a query that displays all the products along with the Sales OrderID even if an order has never been placed for that product.

4. Find the subcategories that have at least two different prices less than $15.

5. A. Write a query to display employees and their manager details. Fetch employee id, employee first name, and manager id, manager name.
B. Display the employee id and employee name of employees who do not have manager.

6. A. Display the names of all products of a particular subcategory 15 and the names of their vendors.
B. Find the products that have more than one vendor.

7. Find all the customers who do not belong to any store.

8. Find sales prices of product 718 that are less than the list price recommended for that product.

9. Display product number, description and sales of each product in the year 2001.

10. Build the logic on the above question to extract sales for each category by year. Fetch Product Name, Sales_2001, Sales_2002, Sales_2003.

Hint: For questions 9 & 10 (From Sales.SalesOrderHeader, sales. SalesOrderDetail, Production. Product. Use ShipDate of SalesOrderHeader to extract shipped year. Calculate sales using QTY and unitprice from Sales OrderDetail.)

Database Name: HR

1. Write a query to display the last name and hire date of any employee in the same department as SALES.

2. Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.

3. Write a query that displays the employee numbers and last names of all employees who work in a department with any employee whose last name contains a' u

4. Display the last name, department number, and job ID of all employees whose department location is ATLANTA.

5. Display the last name and salary of every employee who reports to FILLMORE.

6. Display the department number, last name, and job ID for every employee in the OPERATIONS department.

7. Modify the above query to display the employee numbers, last names, and salaries of all employees who earn more than the average salary and who work in a department with any employee with a 'u'in their name.

8. Display the names of all employees whose job title is the same as anyone in the sales dept.

9. Write a compound query to produce a list of employees showing raise percentages, employee IDs, and salaries. Employees in department 1 and 3 are given a 5% raise, employees in department 2 are given a 10% raise, employees in departments 4 and 5 are given a 15% raise, and employees in department 6 are not given a raise.

10. Write a query to display the top three earners in the EMPLOYEES table. Display their last names and salaries.

11. Display the names of all employees with their salary and commission earned. Employees with a null commission should have O in the commission column

12. Display the Managers (name) with top three salaries along with their salaries and department information.

Date Function

1) Find the date difference between the hire date and resignation_date for all the employees. Display in no. of days, months and year(1 year 3 months 5 days).

Emp_ID	Hire Date	Resignation_Date
1	1/1/2000	7/10/2013
2	4/12/2003	3/8/2017
3	22/9/2012	21/6/2015
4	13/4/2015	NULL
5	03/06/2016	NULL
6	08/08/2017	NULL
7	13/11/2016	NULL

2) Format the hire date as mm/dd/yyyy(09/22/2003) and resignation_date as mon dd, yyyy(Aug 12th, 2004). Display the null as (DEC, 01th 1900)

3) Calcuate experience of the employee till date in Years and months(example 1 year and 3 months)

Use Getdate() as input date for the below three questions.

4) Display the count of days in the previous quarter

5) Fetch the previous Quarter's second week's first day's date

6) Fetch the financial year's 15th week's dates (Format: Mon DD YYYY)

7) Find out the date that corresponds to the last Saturday of January, 2015 using with clause.

Use Airport database for the below two question:

8) Find the number of days elapsed between first and last flights of a passenger.

9) Find the total duration in minutes and in seconds of the flight from Rostov to Moscow.

Module 6(Power Pivot & Power BI)

Power Pivot Questions

1) Create a pivot table of average prices for transactions. See and follow the steps in file ***"power_pivot_question_01"***.
2) From the MAM database, import 6 tables and use them to show the quantity sold by town. See and follow the steps in file ***"power_pivot_question_02"***.
3) Import tables from the Make-a-Mammal database, then hide tables and columns to create a clean data model. See and follow the steps in file "***power_pivot_question_03***".
4) Import a table then amend it and import others to create data model. See and follow the steps in file ***"power_pivot_question_04"***.
5) Import tables into PowerPivot, hide tables and columns and create a pivot table and slicer. See and follow the steps in file ***"power_pivot_question_05"***.
6) Create a pivot table with slicer based on ten different tables. See and follow the steps in file ***"power_pivot_question_06"***.
7) Create two pivot tables, and two timelines which control both of the pivot tables. See and follow the steps in file ***"power_pivot_question_07"***.
8) Use slicers to control 2 pivot tables, and Quick Explore to drill down. See and follow the steps in file ***"power_pivot_question_08"***.
9) Use a timeline to restrict a pivot table to 4 specific quarters. See and follow the steps in file ***"power_pivot_question_09"***.
10) Create a linked Excel workbook in PowerPivot and use it in relationships. See and follow the steps in file ***"power_pivot_question_10"***.
11) Import data from Access, Word and Excel, and link an Excel table, to create a PowerPivot data model. See and follow the steps in file ***"power_pivot_question_11"***.
12) Link to Excel, Access and the clipboard (via Word) to import and link 4 tables. See and follow the steps in file ***"power_pivot_question_12"***.
13) Link to two Excel workbooks and one SQL Server table in PowerPivot. See and follow the steps in file ***"power_pivot_question_13"***.
14) Create an aggregator column to sum transaction values by weekday in a pivot table. See and follow the steps in file ***"power_pivot_question_14"***.
15) Create two new calculated columns in a table, using RELATED and CONCATENATE. See and follow the steps in file ***"power_pivot_question_15"***.
16) Total sales by weekday, using simple two calculated columns. See and follow the steps in file ***"power_pivot_question_16"***.
17) Calculate age bands for different dates using the SWITCH function. See and follow the steps in file ***"power_pivot_question_17"***.
18) Divide shopping centres into the circles of hell, using the IF and the SWITCH functions. See and follow the steps in file ***"power_pivot_question_18"***.
19) Divide years into bands using SWITCH and calculated columns. See and follow the steps in file ***"power_pivot_question_19"***.

20) Summarise sales by status of animal, using calculated columns. See and follow the steps in file ***"power_pivot_question_20"***.
21) Calculate total and average transaction values using measures. See and follow the steps in file ***"power_pivot_question_21"***.
22) Calculate ratio of area to units for shopping centres using AVERAGEX. See and follow the steps in file ***"power_pivot_question_22"***.
23) Divide sales by 4 legs and other for a measure, using the FILTER function. See and follow the steps in file ***"power_pivot_question_23"***.
24) Use ALL and CALCULATE to get proportions of totals for regions. See and follow the steps in file ***"power_pivot_question_24"***.
25) Use the CALCULATE function to pick out only transactions whose price is a given amount. See and follow the steps in file ***"power_pivot_question_25"***.
26) Use the CALCULATE function to show percentages of row and column totals in a pivot table. See and follow the steps in file ***"power_pivot_question_26"***.
27) Create a ratio of sales between two different habitats, using the CALCULATE and SUMX functions. See and follow the steps in file ***"power_pivot_question_27"***.
28) Create measure using CALCULATE and ALL to get ratios against total. See and follow the steps in file ***"power_pivot_question_28"***.
29) Exclude a month from totals using the VALUES function to retain context. See and follow the steps in file ***"power_pivot_question_29"***.
30) Use CALCULATE to work out the ratio of total sales to sales for a specific type of animal. See and follow the steps in file ***"power_pivot_question_30"***.
31) Use FILTER and ALL to show sales as a proportion of one region's total. See and follow the steps in file ***"power_pivot_question_31"***.
32) Use the CALCULATE function to show total sales for Northern powerhouse shopping centres. See and follow the steps in file ***"power_pivot_question_32"***.
33) Use the RANKX function to order total sales over species. See and follow the steps in file ***"power_pivot_question_33"***.
34) Filter calculated sums to compare two stores' sales figures. See and follow the steps in file ***"power_pivot_question_34"***.
35) Use AVERAGEX to find average ratios, then CALCULATE to avoid divide-by-zero errors. See and follow the steps in file ***"power_pivot_question_35"***.
36) Create two measures, showing average price for the South and all other regions. See and follow the steps in file ***"power_pivot_question_36"***.
37) Exclude a single animal from a pivot table, using CALCULATE combined with the VALUES function. See and follow the steps in file ***"power_pivot_question_37"***.
38) Sorting total sales into ascending order, using the RANKX function. See and follow the steps in file ***"power_pivot_question_38"***.
39) Group purchases into shopping centre size bands, using the EARLIER and FILTER functions. See and follow the steps in file ***"power_pivot_question_39"***.
40) Group sales into size bands using the EARLIER function. See and follow the steps in file ***"power_pivot_question_40"***.
41) Create a basic report to show a simple table of Abba songs. See and follow the steps in file ***"power_pivot_question_41"***.

42) Create a matrix and return some appropriate images above. See and follow the steps in file ***"power_pivot_question_42"***.
43) Create a report listing Game of Thrones episodes, importing two tables. See and follow the steps in file ***"power_pivot_question_43"***.
44) Load a webpage of the best films, and use this to create a table. See and follow the steps in file ***"power_pivot_question_44"***.
45) Load FTSE data, and create a report with a table, shape and image. See and follow the steps in file ***"power_pivot_question_45"***.
46) Use a matrix to compare the number of websites by country and type. See and follow the steps in file ***"power_pivot_question_46"***.
47) Compare Oscars won by genre and certificate for films using a matrix. See and follow the steps in file ***"power_pivot_question_47"***.
48) Count the number of world events for each country and year. See and follow the steps in file ***"power_pivot_question_48"***.
49) Load example tables from a SQL Server database, and use them to create a matrix. See and follow the steps in file ***"power_pivot_question_49"***.
50) Create relationships between tables using two methods. See and follow the steps in file ***"power_pivot_question_50"***.
51) Load an Excel workbook of Disney princesses, and create a table from this. See and follow the steps in file ***"power_pivot_question_51"***.
52) Importing, tidying up and filtering skyscraper data to create a column chart. See and follow the steps in file ***"power_pivot_question_52"***.
53) Use Query Editor to import and tidy up a list of the richest people. See and follow the steps in file ***"power_pivot_question_53"***.
54) Use Query Editor to load and tidy up a list of FTSE share prices. See and follow the steps in file ***"power_pivot_question_54"***.
55) Use the query editor to transform a rubbish data file into something useful. See and follow the steps in file ***"power_pivot_question_55"***.
56) Use Query Editor to cleanse a list of imported top websites. See and follow the steps in file ***"power_pivot_question_56"***.
57) Use Query Editor to rename and split columns in a Game of Thrones worksheet. See and follow the steps in file ***"power_pivot_question_57"***.
58) Load some pivoted forecast data, unpivot it and much more!. See and follow the steps in file ***"power_pivot_question_58"***.
59) Use Query Editor to remove, transform and add columns to a tall buildings list. See and follow the steps in file ***"power_pivot_question_59"***.
60) Apply a filter and a slicer by continent to a list of most-visited websites. See and follow the steps in file ***"power_pivot_question_60"***.
61) Apply a page filter to a list of films, then create a slicer by category. See and follow the steps in file ***"power_pivot_question_61"***.
62) Allow a user to choose pizzas by calorie count and type using slicers. See and follow the steps in file ***"power_pivot_question_62"***.
63) Create a date, numeric, dropdown and horizontal slicer on a report. See and follow the steps in file ***"power_pivot_question_63"***.

64) Create a slicer and chart to choose which whale sightings dataset you want to see. See and follow the steps in file ***"power_pivot_question_64"***.
65) Import skyscraper data, creating a new column and showing this in a chart controlled by a slicer. See and follow the steps in file ***"power_pivot_question_65"***.
66) Use hidden synced slicers to filter all pages with a single slicer. See and follow the steps in file ***"power_pivot_question_66"***.
67) Create date and normal slicers on one page to affect visuals on other pages. See and follow the steps in file ***"power_pivot_question_67"***.
68) Create linked slicers to show a chart of crime statistics. See and follow the steps in file ***"power_pivot_question_68"***.
69) Enable drill-through for a report to show a breakdown of tests taken. See and follow the steps in file ***"power_pivot_question_69"***.
70) Compare Pizza Express pizza calories using pie and doughnut charts. See and follow the steps in file ***"power_pivot_question_70"***.
71) Compare the number of Abba songs released by year using a column chart. See and follow the steps in file ***"power_pivot_question_71"***.
72) Create a donut chart of population data, and morph this into a tree chart. See and follow the steps in file ***"power_pivot_question_72"***.
73) Analyse 2018 crime figures for the Manchester area using various visuals. See and follow the steps in file ***"power_pivot_question_73"***.
74) Compare the heights of skyscrapers by country and city, and create a KPI. See and follow the steps in file ***"power_pivot_question_74"***.
75) Create a column chart of record sales, and drill-down to a pie chart. See and follow the steps in file ***"power_pivot_question_75"***.
76) Use grouping in charts to show viewing figures by genre for BBC1. See and follow the steps in file ***"power_pivot_question_76"***.
77) Compare skills test results using a waterfall chart with breakdown. See and follow the steps in file ***"power_pivot_question_77"***.
78) Create a bubble chart comparing two sets of numbers, and play it over time to show changes. See and follow the steps in file ***"power_pivot_question_78"***.
79) Show a chart comparing films when you click on each genre in a tree map. See and follow the steps in file ***"power_pivot_question_79"***.
80) Compare sales of goods across the UK for large shopping centres. See and follow the steps in file ***"power_pivot_question_80"***.
81) Create a map showing passenger numbers for UK stations, with drill-down. See and follow the steps in file ***"power_pivot_question_81"***.
82) Use ArcGIS to generate a heat map showing train passengers in the UK. See and follow the steps in file ***"power_pivot_question_82"***.
83) Create a map comparing house price sales for expensive houses across the UK. See and follow the steps in file ***"power_pivot_question_83"***.
84) Create a map to show sales by town for selected regions. See and follow the steps in file ***"power_pivot_question_84"***.
85) Analyse Brexit voting patterns for the countries of the UK, using Electoral Commission data. See and follow the steps in file ***"power_pivot_question_85"***.

Module 7 (Numpy)

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

Questions on NumPy Array

Write answer of each question and also give explanation with example

- How to create an empty and a full NumPy array?
- Create a Numpy array filled with all zeros
- Create a Numpy array filled with all ones
- Check whether a Numpy array contains a specified row
- How to Remove rows in Numpy array that contains non-numeric values?
- Remove single-dimensional entries from the shape of an array
- Find the number of occurrences of a sequence in a NumPy array
- Find the most frequent value in a NumPy array
- Combining a one and a two-dimensional NumPy Array
- How to build an array of all combinations of two NumPy arrays?

Numpy Array Operations

- In this assignment you will:

  > - Pick 5 interesting Numpy array functions by going through the documentation: https://numpy.org/doc/stable/reference/routines.html
  > - Run and modify this Jupyter notebook to illustrate their usage (some explanation and 3 examples for each function). Use your imagination to come up with interesting and unique examples.

- Try to give your notebook a catchy title & subtitle e.g. `All about Numpy array operations`, `5 Numpy functions you didn't know you needed`, `A beginner's guide to broadcasting in Numpy`, `Interesting ways to create Numpy arrays`, `Trigonometic functions in Numpy`, `How to use Python for Linear Algebra` etc.

Title Here

Subtitle Here

Write a short introduction about Numpy and list the chosen functions.

- function 1
- function 2
- function 3
- function 4
- function 5

In [ ]:
```
# List of functions explained
function1 = np.concatenate  # (change this)
function2 = ???
function3 = ???
function4 = ???
function5 = ???
```

Function 1 - np.concatenate (change this)

Add some explanation about the function in your own words

In [ ]:
```
# Example 1 - working (change this)
arr1 = [[1, 2],
        [3, 4.]]

arr2 = [[5, 6, 7],
        [8, 9, 10]]

np.concatenate((arr1, arr2), axis=1)
```

Explanation about example

In [ ]:
```
# Example 2 - working
???
```

Explanation about example

In [ ]:
```
# Example 3 - breaking (to illustrate when it breaks)
arr1 = [[1, 2],
        [3, 4.]]

arr2 = [[5, 6, 7],
        [8, 9, 10]]

np.concatenate((arr1, arr2), axis=0)
```

Explanation about example (why it breaks and how to fix it)

Some closing comments about when to use this function.

Function 2 - ???

Add some explanations

In [3]:
```
# # Example 1 - working
???
```

Explanation about example

In [ ]:
```
# Example 2 - working
???
```

Explanation about Example

In [ ]:
```
# Example 3 - breaking (to illustrate when it breaks)
???
```

Explanation about example (why it breaks and how to fix it)

Some closing comments about when to use this function.

Function 3 - ???

Add some explanations

In [ ]:
```
# Example 1 - working

???
```

Explanation about example

In [ ]:
```
# Example 2 - working

???
```

Explanation about example

In [ ]:
```
# Example 3 - breaking (to illustrate when it breaks)
???
```

Explanation about example (why it breaks and how to fix it)

Some closing comments about when to use this function.

Function 4 - ???

Add some explanations

In [ ]:
```
# Example 1 - working

???
```

Explanation about example

In [ ]:
```
# Example 2 - working
???
```

Explanation about example

In [ ]:
```
# Example 3 - breaking (to illustrate when it breaks)
???
```

Explanation about example (why it breaks and how to fix it)
Some closing comments about when to use this function.

Function 5 - ???

Add some explanations

In [ ]:
```
# Example 1 - working

???
```

Explanation about example

In [5]:
```
# Example 2- working

???
```

Explanation about example

In [ ]:
```
# Example 3 - breaking (to illustrate when it breaks)
???
```

Explanation about example (why it breaks and how to fix it)
Some closing comments about when to use this function.

Reference Links

Provide links to your references and other interesting articles about Numpy arrays:

- Numpy official tutorial : https://numpy.org/doc/stable/user/quickstart.html

Ex2 - Getting and Knowing your Data

This time we are going to pull data directly from the internet. Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

Step 4. See the first 10 entries

Step 5. What is the number of observations in the dataset?

Step 6. What is the number of columns in the dataset?

Step 7. Print the name of all the columns.

Step 8. How is the dataset indexed?

Step 9. Which was the most-ordered item?

Step 10. For the most-ordered item, how many items were ordered?

Step 11. What was the most ordered item in the choice_description column?

Step 12. How many items were orderd in total?

Step 13. Turn the item price into a float

Step 13.a. Check the item price type

Step 13.b. Create a lambda function and change the type of item price

Step 13.c. Check the item price type

Step 14. How much was the revenue for the period in the dataset?

Step 15. How many orders were made in the period?

Step 16. What is the average revenue amount per order?

Step 17. How many different items are sold?

Ex3 - Getting and Knowing your Data

This time we are going to pull data directly from the internet. Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called users and use the 'user_id' as index

Step 4. See the first 25 entries

Step 5. See the last 10 entries

Step 6. What is the number of observations in the dataset?

Step 7. What is the number of columns in the dataset?

Step 8. Print the name of all the columns.

Step 9. How is the dataset indexed?

Step 10. What is the data type of each column?

Step 11. Print only the occupation column

Step 12. How many different occupations are in this dataset?

Step 13. What is the most frequent occupation?

Step 14. Summarize the DataFrame.

Step 15. Summarize all the columns

Step 16. Summarize only the occupation column

Step 17. What is the mean age of users?

Step 18. What is the age with least occurrence?

Step 1. Go to https://www.kaggle.com/openfoodfacts/world-food-facts/data

Step 2. Download the dataset to your computer and unzip it.

Step 3. Use the tsv file and assign it to a dataframe called food

Step 4. See the first 5 entries

Step 5. What is the number of observations in the dataset?

Step 6. What is the number of columns in the dataset?

Step 7. Print the name of all the columns.

Step 8. What is the name of 105th column?

Step 9. What is the type of the observations of the 105th column?

Step 10. How is the dataset indexed?

Step 11. What is the product name of the 19th observation?

Ex1 - Filtering and Sorting Data

This time we are going to pull data directly from the internet. Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

Step 4. How many products cost more than $10.00?

Step 5. What is the price of each item?

print a data frame with only two columns item_name and item_price

Step 6. Sort by the name of the item

Step 7. What was the quantity of the most expensive item ordered?

Step 8. How many times was a Veggie Salad Bowl ordered?

Step 9. How many times did someone order more than one Canned Soda?

Ex2 - Filtering and Sorting Data

This time we are going to pull data directly from the internet.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called euro12.

Step 4. Select only the Goal column.

Step 5. How many team participated in the Euro2012?

Step 6. What is the number of columns in the dataset?

Step 7. View only the columns Team, Yellow Cards and Red Cards and assign them to a dataframe called discipline

Step 8. Sort the teams by Red Cards, then to Yellow Cards

Step 9. Calculate the mean Yellow Cards given per Team

Step 10. Filter teams that scored more than 6 goals

Step 11. Select the teams that start with G

Step 12. Select the first 7 columns

Step 13. Select all columns except the last 3.

Step 14. Present only the Shooting Accuracy from England, Italy and Russia

Fictional Army - Filtering and Sorting

Introduction:

This exercise was inspired by this page

Special thanks to: https://github.com/chrisalbon for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. This is the data given as a dictionary

In [4]:
```
# Create an example dataframe about a fictional army
raw_data = {'regiment': ['Nighthawks', 'Nighthawks', 'Nighthawks',
                         'Nighthawks',
                         'Dragoons', 'Dragoons', 'Dragoons', 'Dragoons', 'Scouts',
                         'Scouts', 'Scouts', 'Scouts'],
            'company': ['1st', '1st', '2nd', '2nd', '1st', '1st',
                        '2nd', '2nd','1st', '1st', '2nd', '2nd'],
            'deaths': [523, 52, 25, 616, 43, 234, 523, 62, 62, 73, 37, 35],
            'battles': [5, 42, 2, 2, 4, 7, 8, 3, 4, 7, 8, 9],
            'size': [1045, 957, 1099, 1400, 1592, 1006, 987,
                     849,
                     973, 1005, 1099, 1523],
            'veterans': [1, 5, 62, 26, 73, 37, 949, 48, 48,
                         435, 63, 345],
            'readiness': [1, 2, 3, 3, 2, 1, 2, 3, 2, 1, 2, 3],
            'armored': [1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1],
            'deserters': [4, 24, 31, 2, 3, 4, 24, 31, 2, 3, 2, 3],
            'origin': ['Arizona', 'California', 'Texas', 'Florida', 'Maine',
                       'Iowa', 'Alaska', 'Washington', 'Oregon', 'Wyoming',
                       'Louisana', 'Georgia']}
```

Step 3. Create a dataframe and assign it to a variable called army.

Don't forget to include the columns names in the order presented in the dictionary ('regiment', 'company', 'deaths'...) so that the column index order is consistent with the solutions. If omitted, pandas will order the columns alphabetically.

Step 4. Set the 'origin' colum as the index of the dataframe

Step 5. Print only the column veterans

Step 6. Print the columns 'veterans' and 'deaths'

Step 7. Print the name of all the columns.

Step 8. Select the 'deaths', 'size' and 'deserters' columns from Maine and Alaska

Step 9. Select the rows 3 to 7 and the columns 3 to 6

Step 10. Select every row after the fourth row and all columns

Step 11. Select every row up to the 4th row and all columns

Step 12. Select the 3rd column up to the 7th column

Step 13. Select rows where df.deaths is greater than 50

Step 14. Select rows where df.deaths is greater than 500 or less than 50

Step 15. Select all the regiments not named "Dragoons"

Step 16. Select the rows called Texas and Arizona

Step 17. Select the third cell in the row named Arizona

Step 18. Select the third cell down in the column named deaths

Ex - GroupBy

Introduction:

GroupBy can be summarized as Split-Apply-Combine.

Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Check out this Diagram

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called drinks.

Step 4. Which continent drinks more beer on average?

Step 5. For each continent print the statistics for wine consumption.

Step 6. Print the mean alcohol consumption per continent for every column

Step 7. Print the median alcohol consumption per continent for every column

Step 8. Print the mean, min and max values for spirit consumption.

This time output a DataFrame

Occupation

Introduction:

Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called users.

Step 4. Discover what is the mean age per occupation

Step 5. Discover the Male ratio per occupation and sort it from the most to the least

Step 6. For each occupation, calculate the minimum and maximum ages

Step 7. For each combination of occupation and gender, calculate the mean age

Step 8. For each occupation present the percentage of women and men

Regiment

Introduction:

Special thanks to: http://chrisalbon.com/ for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Create the DataFrame with the following values:

In [51]:
```
raw_data = {'regiment': ['Nighthawks', 'Nighthawks', 'Nighthawks',
                         'Nighthawks', 'Dragoons', 'Dragoons', 'Dragoons',
                         'Dragoons', 'Scouts', 'Scouts', 'Scouts', 'Scouts'],
        'company': ['1st', '1st', '2nd', '2nd', '1st', '1st',
                    '2nd', '2nd','1st', '1st', '2nd', '2nd'],
        'name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze',
                 'Jacon', 'Ryaner', 'Sone', 'Sloan', 'Piger',
                 'Riani', 'Ali'],
        'preTestScore': [4, 24, 31, 2, 3, 4, 24, 31, 2, 3,
                         2, 3],
        'postTestScore': [25, 94, 57, 62, 70, 25, 94, 57,
                          62, 70, 62, 70]}
```

Step 3. Assign it to a variable called regiment.

Don't forget to name each column

Step 4. What is the mean preTestScore from the regiment Nighthawks?

Step 5. Present general statistics by company

Step 6. What is the mean of each company's preTestScore?

Step 7. Present the mean preTestScores grouped by regiment and company

Step 8. Present the mean preTestScores grouped by regiment and company without heirarchical indexing

Step 9. Group the entire dataframe by regiment and company

Step 10. What is the number of observations in each regiment and company

Step 11. Iterate over a group and print the name and the whole data from the regiment

Student Alcohol Consumption

Introduction:

This time you will download a dataset from the UCI.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called df.

Step 4. For the purpose of this exercise slice the dataframe from 'school' until the 'guardian' column

Step 5. Create a lambda function that will capitalize strings.

Step 6. Capitalize both Mjob and Fjob

Step 7. Print the last elements of the data set.

Step 8. Did you notice the original dataframe is still lowercase? Why is that? Fix it and capitalize Mjob and Fjob.

Step 9. Create a function called majority that returns a boolean value to a new column called legal_drinker (Consider majority as older than 17 years old)

Step 10. Multiply every number of the dataset by 10.

I know this makes no sense, don't forget it is just an exercise

United States - Crime Rates - 1960 - 2014

Introduction:

This time you will create a data

Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called crime.

Step 4. What is the type of the columns?

Have you noticed that the type of Year is int64. But pandas has a different type to work with Time Series. Let's see it now.

Step 5. Convert the type of the column Year to datetime64

Step 6. Set the Year column as the index of the dataframe

Step 7. Delete the Total column

Step 8. Group the year by decades and sum the values

Pay attention to the Population column number, summing this column is a mistake

Step 9. What is the most dangerous decade to live in the US?

MPG Cars

Introduction:

The following exercise utilizes data from UC Irvine Machine Learning Repository

Step 1. Import the necessary libraries

Step 2. Import the first dataset cars1 and cars2.

Step 3. Assign each to a variable called cars1 and cars2

Step 4. Oops, it seems our first dataset has some unnamed blank columns, fix cars1

Step 5. What is the number of observations in each dataset?

Step 6. Join cars1 and cars2 into a single DataFrame called cars

Step 7. Oops, there is a column missing, called owners. Create a random number Series from 15,000 to 73,000.

Step 8. Add the column owners to cars

Fictitious Names

Introduction:

This time you will create a data again

Special thanks to Chris Albon for sharing the dataset and materials. All the credits to this exercise belongs to him.

In order to understand about it go here.

Step 1. Import the necessary libraries

Step 2. Create the 3 DataFrames based on the following raw data

In [1]:
```
raw_data_1 = {
        'subject_id': ['1', '2', '3', '4', '5'],
        'first_name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
        'last_name': ['Anderson', 'Ackerman', 'Ali', 'Aoni', 'Atiches']}

raw_data_2 = {
        'subject_id': ['4', '5', '6', '7', '8'],
        'first_name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
        'last_name': ['Bonder', 'Black', 'Balwner', 'Brice', 'Btisan']}

raw_data_3 = {
        'subject_id': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'],
        'test_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]}
```

Step 3. Assign each to a variable called data1, data2, data3

Step 4. Join the two dataframes along rows and assign all_data

Step 5. Join the two dataframes along columns and assing to all_data_col

Step 6. Print data3

Step 7. Merge all_data and data3 along the subject_id value

Step 8. Merge only the data that has the same 'subject_id' on both data1 and data2

Step 9. Merge all values in data1 and data2, with matching records from both sides where available.

Housing Market

Introduction:

This time we will create our own dataset with fictional numbers to describe a house market.
As we are going to create random data don't try to reason of the numbers.

Step 1. Import the necessary libraries

Step 2. Create 3 differents Series, each of length 100, as follows:

1. The first a random number from 1 to 4
2. The second a random number from 1 to 3
3. The third a random number from 10,000 to 30,000

Step 3. Let's create a DataFrame by joinning the Series by column

Step 4. Change the name of the columns to bedrs, bathrs, price_sqr_meter

Step 5. Create a one column DataFrame with the values of the 3 Series and assign it to 'bigcolumn'

Step 6. Oops, it seems it is going only until index 99. Is it true?

Step 7. Reindex the DataFrame so it goes from 0 to 299

US - Baby Names

Introduction:

We are going to use a subset of US Baby Names from Kaggle.
In the file it will be names from 2004 until 2014

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called baby_names.

Step 4. See the first 10 entries

Step 5. Delete the column 'Unnamed: 0' and 'Id'

Step 6. Is there more male or female names in the dataset?

Step 7. Group the dataset by name and assign to names

Step 8. How many different names exist in the dataset?

Step 9. What is the name with most occurrences?

Step 10. How many different names have the least occurrences?

Step 11. What is the median name occurrence?

Step 12. What is the standard deviation of names?

Step 13. Get a summary with the mean, min, max, std and quartiles.

Wind Statistics

Introduction:

The data have been modified to contain some missing values, identified by NaN.
Using pandas should make this exercise easier, in particular for the bonus question.

You should be able to perform all of these operations without using a for loop or other looping construct.

1. The data in 'wind.data' has the following format:

In [434…
```
"""
Yr Mo Dy   RPT   VAL   ROS   KIL   SHA   BIR   DUB   CLA   MUL   CLO   BEL   MAL
61  1  1 15.04 14.96 13.17  9.29   NaN  9.87 13.67 10.25 10.83 12.58 18.50 15.04
61  1  2 14.71   NaN 10.83  6.50 12.62  7.67 11.50 10.04  9.79  9.67 17.54 13.83
61  1  3 18.50 16.88 12.33 10.13 11.17  6.17 11.25   NaN  8.50  7.67 12.75 12.71
"""
```

Out[434]:
```
'\nYr Mo Dy   RPT   VAL   ROS   KIL   SHA   BIR   DUB   CLA   MUL   CLO   BEL   MA
L\n61  1  1 15.04 14.96 13.17  9.29   NaN  9.87 13.67 10.25 10.83 12.58 18.50 15.0
4\n61  1  2 14.71   NaN 10.83  6.50 12.62  7.67 11.50 10.04  9.79  9.67 17.54 13.8
3\n61  1  3 18.50 16.88 12.33 10.13 11.17  6.17 11.25   NaN  8.50  7.67 12.75 12.7
1\n'
```

The first three columns are year, month and day. The remaining 12 columns are average windspeeds in knots at 12 locations in Ireland on that day.

More information about the dataset go here.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address

Step 3. Assign it to a variable called data and replace the first 3 columns by a proper datetime index.

Step 4. Year 2061? Do we really have data from this year? Create a function to fix it and apply it.

Step 5. Set the right dates as the index. Pay attention at the data type, it should be datetime64[ns].

Step 6. Compute how many values are missing for each location over the entire record.

They should be ignored in all calculations below.

Step 7. Compute how many non-missing values there are in total.

Step 8. Calculate the mean windspeeds of the windspeeds over all the locations and all the times.

A single number for the entire dataset.

Step 9. Create a DataFrame called loc_stats and calculate the min, max and mean windspeeds and standard deviations of the windspeeds at each location over all the days

A different set of numbers for each location.

Step 10. Create a DataFrame called day_stats and calculate the min, max and mean windspeed and standard deviations of the windspeeds across all the locations at each day.

A different set of numbers for each day.

Step 11. Find the average windspeed in January for each location.

Treat January 1961 and January 1962 both as January.

Step 12. Downsample the record to a yearly frequency for each location.

Step 13. Downsample the record to a monthly frequency for each location.

Step 14. Downsample the record to a weekly frequency for each location.

Step 15. Calculate the min, max and mean windspeeds and standard deviations of the windspeeds across all locations for each week (assume that the first week starts on January 2 1961) for the first 52 weeks.

Visualizing Chipotle's Data

This time we are going to pull data directly from the internet. Special thanks to: https://github.com/justmarkham for sharing the dataset and materials.

Step 1. Import the necessary libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter

# set this so the graphs open internally
%matplotlib inline
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

Step 4. See the first 10 entries

Step 5. Create a histogram of the top 5 items bought

Step 6. Create a scatterplot with the number of items orderered per order price

Hint: Price should be in the X-axis and Items ordered in the Y-axis

Step 7. BONUS: Create a question and a graph to answer your own question.

Online Retails Purchase

Introduction:

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called online_rt

Note: if you receive a utf-8 decode error, set `encoding = 'latin1'` in `pd.read_csv()`.

Step 4. Create a histogram with the 10 countries that have the most 'Quantity' ordered except UK

Step 5. Exclude negative Quantity entries

Step 6. Create a scatterplot with the Quantity per UnitPrice by CustomerID for the top 3 Countries (except UK)

Step 7. Investigate why the previous results look so uninformative.

This section might seem a bit tedious to go through. But I've thought of it as some kind of a simulation of problems one might encounter when dealing with data and other people. Besides there is a prize at the end (i.e. Section 8).

(But feel free to jump right ahead into Section 8 if you want; it doesn't require that you finish this section.)

Step 7.1 Look at the first line of code in Step 6. And try to figure out if it leads to any kind of problem.

Step 7.1.1 Display the first few rows of that DataFrame.

Step 7.1.2 Think about what that piece of code does and display the dtype of `UnitPrice`

Step 7.1.3 Pull data from `online_rt` for `CustomerID`s 12346.0 and 12347.0.

Step 7.2 Reinterpreting the initial problem.

To reiterate the question that we were dealing with:
"Create a scatterplot with the Quantity per UnitPrice by CustomerID for the top 3 Countries"

The question is open to a set of different interpretations. We need to disambiguate.

We could do a single plot by looking at all the data from the top 3 countries. Or we could do one plot per country. To keep things consistent with the rest of the exercise, let's stick to the latter oprion. So that's settled.

But "top 3 countries" with respect to what? Two answers suggest themselves: Total sales volume (i.e. total quantity sold) or total sales (i.e. revenue). This exercise goes for sales volume, so let's stick to that.

Step 7.2.1 Find out the top 3 countries in terms of sales volume.

Step 7.2.2

Now that we have the top 3 countries, we can focus on the rest of the problem:
"Quantity per UnitPrice by CustomerID".
We need to unpack that.

"by CustomerID" part is easy. That means we're going to be plotting one dot per CustomerID's on our plot. In other words, we're going to be grouping by CustomerID.

"Quantity per UnitPrice" is trickier. Here's what we know:
One axis will represent a Quantity assigned to a given customer. This is easy; we can just plot the total Quantity for each customer. The other axis will represent a UnitPrice assigned to a given customer. Remember a single customer can have any number of orders with different prices, so summing up prices isn't quite helpful. Besides it's not quite clear what we mean when we say "unit price per customer"; it sounds like price of the customer! A reasonable alternative is that we assign each customer the average amount each has paid per item. So let's settle that question in that manner.

Step 7.3 Modify, select and plot data

Step 7.3.1 Add a column to online_rt called `Revenue` calculate the revenue (Quantity * UnitPrice) from each sale.

We will use this later to figure out an average price per customer.

Step 7.3.2 Group by `CustomerID` and `Country` and find out the average price (`AvgPrice`) each customer spends per unit.

Step 7.3.3 Plot

Step 7.4 What to do now?

We aren't much better-off than what we started with. The data are still extremely scattered around and don't seem quite informative.

But we shouldn't despair! There are two things to realize: 1) The data seem to be skewed towaards the axes (e.g. we don't have any values where Quantity = 50000 and AvgPrice = 5). So that might suggest a trend. 2) We have more data! We've only been looking at the data from 3 different countries and they are plotted on different graphs.

So: we should plot the data regardless of `Country` and hopefully see a less scattered graph.

Step 7.4.1 Plot the data for each `CustomerID` on a single graph

Step 7.4.2 Zoom in so we can see that curve more clearly

8. Plot a line chart showing revenue (y) per UnitPrice (x).

Did Step 7 give us any insights about the data? Sure! As average price increases, the quantity ordered decreses. But that's hardly surprising. It would be surprising if that wasn't the case!

Nevertheless the rate of drop in quantity is so drastic, it makes me wonder how our revenue changes with respect to item price. It would not be that surprising if it didn't change that much. But it would be interesting to know whether most of our revenue comes from expensive or inexpensive items, and how that relation looks like.

That is what we are going to do now.

8.1 Group `UnitPrice` by intervals of 1 for prices [0,50), and sum `Quantity` and `Revenue`.

8.3 Plot.

8.4 Make it look nicer.

x-axis needs values.
y-axis isn't that easy to read; show in terms of millions.

BONUS: Create your own question and answer it.

Scores

Introduction:

This time you will create the data.

Exercise based on Chris Albon work, the credits belong to him.

Step 1. Import the necessary libraries

Step 2. Create the DataFrame that should look like the one below.

In [2]:

Out[2]:

	first_name	last_name	age	female	preTestScore	postTestScore
0	Jason	Miller	42	0	4	25
1	Molly	Jacobson	52	1	24	94
2	Tina	Ali	36	1	31	57
3	Jake	Milner	24	0	2	62
4	Amy	Cooze	73	1	3	70

Step 3. Create a Scatterplot of preTestScore and postTestScore, with the size of each point determined by age

Hint: Don't forget to place the labels

Step 4. Create a Scatterplot of preTestScore and postTestScore.

This time the size should be 4.5 times the postTestScore and the color determined by sex

BONUS: Create your own question and answer it.

Tips

Introduction:

This exercise was created based on the tutorial and documentation from Seaborn
The dataset being used is tips from Seaborn.

Step 1. Import the necessary libraries:

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called tips

Step 4. Delete the Unnamed 0 column

Step 5. Plot the total_bill column histogram

Step 6. Create a scatter plot presenting the relationship between total_bill and tip

Step 7. Create one image with the relationship of total_bill, tip and size.

Hint: It is just one function.

Step 8. Present the relationship between days and total_bill value

Step 9. Create a scatter plot with the day as the y-axis and tip as the x-axis, differ the dots by sex

Step 10. Create a box plot presenting the total_bill per day differetiation the time (Dinner or Lunch)

Step 11. Create two histograms of the tip value based for Dinner and Lunch. They must be side by side.

Step 12. Create two scatterplots graphs, one for Male and another for Female, presenting the total_bill value and tip relationship, differing by smoker or no smoker

They must be side by side.

BONUS: Create your own question and answer it using a graph.

Visualizing the Titanic Disaster

Introduction:

This exercise is based on the titanic Disaster dataset avaiable at Kaggle.
To know more about the variables check here

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address

Step 3. Assign it to a variable titanic

Step 4. Set PassengerId as the index

Step 5. Create a pie chart presenting the male/female proportion

Step 6. Create a scatterplot with the Fare payed and the Age, differ the plot color by gender

Step 7. How many people survived?

Step 8. Create a histogram with the Fare payed

BONUS: Create your own question and answer it.

Pokemon

Introduction:

This time you will create the data.

Step 1. Import the necessary libraries

Step 2. Create a data dictionary that looks like the DataFrame below

Step 3. Assign it to a variable called pokemon

In [5]:

Out[5]:

	evolution	hp	name	pokedex	type
0	Ivysaur	45	Bulbasaur	yes	grass
1	Charmeleon	39	Charmander	no	fire
2	Wartortle	44	Squirtle	yes	water
3	Metapod	45	Caterpie	no	bug

Step 4. Ops...it seems the DataFrame columns are in alphabetical order. Place the order of the columns as name, type, hp, evolution, pokedex

Step 5. Add another column called place, and insert what you have in mind.

Step 6. Present the type of each column

BONUS: Create your own question and answer it.

Apple Stock

Introduction:

We are going to use Apple's stock price.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address

Step 3. Assign it to a variable apple

Step 4. Check out the type of the columns

Step 5. Transform the Date column as a datetime type

Step 6. Set the date as the index

Step 7. Is there any duplicate dates?

Step 8. Ops...it seems the index is from the most recent date. Make the first entry the oldest date.

Step 9. Get the last business day of each month

Step 10. What is the difference in days between the first day and the oldest

Step 11. How many months in the data we have?

Step 12. Plot the 'Adj Close' value. Set the size of the figure to 13.5 x 9 inches

BONUS: Create your own question and answer it.

Getting Financial Data - Pandas Datareader

Introduction:

This time you will get data from a website.

Step 1. Import the necessary libraries

Step 2. Create your time range (start and end variables). The start date should be 01/01/2015 and the end should today (whatever your today is).

Step 3. Get an API key for one of the APIs that are supported by Pandas Datareader, preferably for AlphaVantage.

If you do not have an API key for any of the supported APIs, it is easiest to get one for AlphaVantage. (Note that the API key is shown directly after the signup. You do *not* receive it via e-mail.)

(For a full list of the APIs that are supported by Pandas Datareader, see here. As the APIs are provided by third parties, this list may change.)

Step 4. Use Pandas Datarader to read the daily time series for the Apple stock (ticker symbol AAPL) between 01/01/2015 and today, assign it to df_apple and print it.

Step 5. Add a new column "stock" to the dataframe and add the ticker symbol

Step 6. Repeat the two previous steps for a few other stocks, always creating a new dataframe: Tesla, IBM and Microsoft. (Ticker symbols TSLA, IBM and MSFT.)

Step 7. Combine the four separate dataFrames into one combined dataFrame df that holds the information for all four stocks

Step 8. Shift the stock column into the index (making it a multi-level index consisting of the ticker symbol and the date).

Step 7. Create a dataFrame called vol, with the volume values.

Step 8. Aggregate the data of volume to weekly.

Hint: Be careful to not sum data from the same week of 2015 and other years.

Step 9. Find all the volume traded in the year of 2015

Investor - Flow of Funds - US

Introduction:

Special thanks to: https://github.com/rgrp for sharing the dataset.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called

Step 4. What is the frequency of the dataset?

Step 5. Set the column Date as the index.

Step 6. What is the type of the index?

Step 7. Set the index to a DatetimeIndex type

Step 8. Change the frequency to monthly, sum the values and assign it to monthly.

Step 9. You will notice that it filled the dataFrame with months that don't have any data with NaN. Let's drop these rows.

Step 10. Good, now we have the monthly data. Now change the frequency to year.

BONUS: Create your own question and answer it.

Iris

Introduction:

This exercise may seem a little bit strange, but keep doing it.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called iris

Step 4. Create columns for the dataset

In [57]:
```
# 1. sepal_length (in cm)
# 2. sepal_width (in cm)
# 3. petal_length (in cm)
# 4. petal_width (in cm)
# 5. class
```

Step 5. Is there any missing value in the dataframe?

Step 6. Lets set the values of the rows 10 to 29 of the column 'petal_length' to NaN

Step 7. Good, now lets substitute the NaN values to 1.0

Step 8. Now let's delete the column class

Step 9. Set the first 3 rows as NaN

Step 10. Delete the rows that have NaN

Step 11. Reset the index so it begins with 0 again

BONUS: Create your own question and answer it.

Wine

Introduction:

This exercise is a adaptation from the UCI Wine dataset. The only pupose is to practice deleting data with pandas.

Step 1. Import the necessary libraries

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called wine

Step 4. Delete the first, fourth, seventh, nineth, eleventh, thirteenth and fourteenth columns

Step 5. Assign the columns as below:

The attributes are (donated by Riccardo Leardi, riclea '@' anchem.unige.it):
1) alcohol
2) malic_acid
3) alcalinity_of_ash
4) magnesium
5) flavanoids
6) proanthocyanins
7) hue

Step 6. Set the values of the first 3 rows from alcohol as NaN

Step 7. Now set the value of the rows 3 and 4 of magnesium as NaN

Step 8. Fill the value of NaN with the number 10 in alcohol and 100 in magnesium

Step 9. Count the number of missing values

Step 10. Create an array of 10 random numbers up until 10

Step 11. Use random numbers you generated as an index and assign NaN value to each of cell.

Step 12. How many missing values do we have?

Step 13. Delete the rows that contain missing values

Step 14. Print only the non-null values in alcohol

Step 15. Reset the index, so it starts with 0 again

BONUS: Create your own question and answer it.

Module (9, 10, 11) [Machine Learning & Deep Learning]

Applied Machine Learning

Homework 1: Programming with Python

About this assignment:

The main purpose of this assignment is to check whether your programming knowledge is adequate to take this class. This assignment covers two python packages, `numpy` and `pandas`, which we'll be using throughout the course. For some of you, Python/numpy/pandas will be familiar; for others, it will be new. Either way, if you find this assignment very difficult then that could be a sign that you will struggle later on in the course.

Also, as part of this assignment you will likely need to consult the documentation for various Python packages we're using. This is, of course, totally OK and in fact strongly encouraged. Reading and interpreting documentation is an important skill, and in fact is one of the skills this assignment is meant to assess.

Imports

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Points

Each question or sub-question will have a number of points allocated to it, which is indicated right below the question name.

Exercise 1: Loading files with Pandas

rubric={points:12}

When working with tabular data, you will typically be creating Pandas dataframes by reading data from .csv files using `pd.read_csv()`. The documentation for this function is available here.

In the "data" folder in this homework repository there are 6 different .csv files named `wine_#.csv/.txt`. Look at each of these files and use `pd.read_csv()` to load these data so that they resemble the following:

Bottle	Grape	Origin	Alcohol	pH	Colour	Aroma
1	Chardonnay	Australia	14.23	3.51	White	Floral
2	Pinot Grigio	Italy	13.20	3.30	White	Fruity
3	Pinot Blanc	France	13.16	3.16	White	Citrus
4	Shiraz	Chile	14.91	3.39	Red	Berry
5	Malbec	Argentina	13.83	3.28	Red	Fruity

You are provided with tests that use `df.equals()` to check that all the dataframes are identical. If you're in a situation where the two dataframes look identical but `df.equals()` is returning `False`, it may be an issue of types - try checking `df.index`, `df.columns`, or `df.info()`.

```
df1 = pd.read_csv("./data/wine_1.csv")
df2 = None
df3 = None
df4 = None
df5 = None
df6 = None
```

```
df1
```

	Bottle	Grape	Origin	Alcohol	pH	Colour	Aroma
0	1	Chardonnay	Australia	14.23	3.51	White	Floral
1	2	Pinot Grigio	Italy	13.20	3.30	White	Fruity
2	3	Pinot Blanc	France	13.16	3.16	White	Citrus
3	4	Shiraz	Chile	14.91	3.39	Red	Berry
4	5	Malbec	Argentina	13.83	3.28	Red	Fruity

```
for i, df in enumerate([df2, df3, df4, df5, df6]):
    assert df1.equals(df), f"df1 not equal to df{i + 2}"
print("All tests passed.")
```

```
df
```

Exercise 2: The Titanic dataset

The file *titanic.csv* contains data of 1309 passengers who were on the Titanic's unfortunate voyage. For each passenger, the following data are recorded:

- survival - Survival (0 = No; 1 = Yes)
- class - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- name - Name
- sex - Sex
- age - Age
- sibsp - Number of Siblings/Spouses Aboard
- parch - Number of Parents/Children Aboard
- ticket - Ticket Number
- fare - Passenger Fare
- cabin - Cabin
- embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
- boat - Lifeboat (if survived)
- body - Body number (if did not survive and body was recovered)

In this exercise you will perform a number of wrangling operations to manipulate and extract subsets of the data.

Note: many popular datasets have sex as a feature where the possible values are male and female. This representation reflects how the data were collected and is not meant to imply that, for example, gender is binary.

2(a)

rubric={points:1}

Load the `titanic.csv` dataset into a pandas dataframe named `titanic_df`.

```
titanic_df = None
```

```
assert set(titanic_df.columns) == set(
    [
        "pclass",
        "survived",
        "name",
        "sex",
        "age",
        "sibsp",
        "parch",
```

```
        "ticket",
        "fare",
        "cabin",
        "embarked",
        "boat",
        "body",
        "home.dest",
    ]
), "All required columns are not present"
assert len(titanic_df.index) == 1309, "Wrong number of rows in dataframe"
print("Success")
```

2(b)

rubric={points:2}

The column names `sibsp` and `parch` are not very descriptive. Use `df.rename()` to rename these columns to `siblings_spouses` and `parents_children` respectively.

```

```

```
assert set(["siblings_spouses", "parents_children"]).issubset(
    titanic_df.columns
), "Column names were not changed properly"
print("Success")
```

2(c)

rubric={points:2}

We will practice indexing different subsets of the dataframe in the following questions.

Select the column `age` using single bracket notation `[]`. What type of object is returned?

```

```

2(d)

rubric={points:2}

Now select the `age` using double bracket notation `[[]]`. What type of object is returned?

```

```

2(e)

rubric={points:1}

Select the columns `pclass`, `survived`, and `age` using a single line of code.

```

```

2(f)

rubric={points:2}

Use the `iloc` method to obtain the first 5 rows of the columns `name`, `sex` and `age` using a single line of code.

```

```

```

```

2(g)

rubric={points:2}

Now use the `loc` method to obtain the first 5 rows of the columns `name`, `sex` and `age` using a single line of code.

```

```

2(h)

rubric={points:2}

How many passengers survived (`survived = 1`) the disaster? Hint: try using `df.query()` or `[]` notation to subset the dataframe and then `df.shape` to check its size.

```

```

2(i)

rubric={points:1}

How many passengers that survived the disaster (`survived = 1`) were over 60 years of age?

```

```

2(j)

rubric={points:2}

What was the lowest and highest fare paid to board the titanic? Store your answers as floats in the variables `lowest` and `highest`.

```
lowest = None
highest = None
```

2(k)

rubric={points:1}

Sort the dataframe by fare paid (most to least).

```

```

2(l)

rubric={points:1}

Save the sorted dataframe to a .csv file called 'titanic_fares.csv' using `to_csv()`.

```

```

2(m)

rubric={points:3}

Create a scatter plot of fare (y-axis) vs. age (x-axis). Make sure to follow the guidelines on figures. You are welcome to use pandas built-in plotting or `matplotlib`.

```

```

2(n)

rubric={points:3}

Create a bar plot of `embarked` values.

> Make sure to name the axes and give a title to your plot.

```
```

Exercise 3: Treasure Hunt

In this exercise, we will generate various collections of objects either as a list, a tuple, or a dictionary. Your task is to inspect the objects and look for treasure, which in our case is a particular object: **the character "T"**.

Your tasks:

For each of the following cases, index into the Python object to obtain the "T" (for Treasure).

> Please do not modify the original line of code that generates `x` (though you are welcome to copy it). You are welcome to answer this question "manually" or by writing code - whatever works for you. However, your submission should always end with a line of code that prints out `'T'` at the end (because you've found it).

```
import string

letters = string.ascii_uppercase
```

The first one is done for you as an example.

Example question

```
x = ("nothing", {-i: l for i, l in enumerate(letters)})
x
```

Example answer:

```
x[1][-19]
```

> Note: In these questions, the goal is not to understand the code itself, which may be confusing. Instead, try to probe the types of the various objects. For example `type(x)` reveals that `x` is a tuple, and `len(x)` reveals that it has two elements. Element 0 just contains "nothing", but element 1 contains more stuff, hence `x[1]`. Then we can again probe `type(x[1])` and see that it's a dictionary. If you `print(x[1])` you'll see that the letter "T" corresponds to the key -19, hence `x[1][-19]`.

3(a)

rubric={points:2}

```
# Do not modify this cell
x = [
    [letters[i] for i in range(26) if i % 2 == 0],
    [letters[i] for i in range(26) if i % 2 == 1],
]
```

```
```

3(b)

rubric={points:2}

```
# Do not modify this cell
np.random.seed(1)
x = np.random.choice(list(set(letters) - set("T")), size=(100, 26), replace=True)
x[np.random.randint(100), np.random.randint(26)] = "T"
```

```

```

3(c)

rubric={points:3}

```
# Do not modify this cell
n = 26
x = dict()
for i in range(n):
    x[string.ascii_lowercase[i]] = {
        string.ascii_lowercase[(j + 1) % n]: [[letters[j]] if j - 2 == i else None]
        for j in range(n)
    }
```

```

```

```

```

Applied Machine Learning

Homework 2: Decision trees and machine learning fundamentals

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.rcParams["font.size"] = 16

from sklearn.model_selection import cross_val_score, cross_validate, train_test_split
from sklearn.tree import DecisionTreeClassifier
```

Introducing the data set

For this assignment you'll be looking at Kaggle's Spotify Song Attributes dataset. The dataset contains a number of features of songs from 2017 and a binary variable `target` that represents whether the user liked the song (encoded as 1) or not (encoded as 0). See the documentation of all the features here.

This dataset is publicly available on Kaggle, and you will have to download it yourself. Follow the steps below to get the data CSV.

1. If you do not have an account with Kaggle, you will first need to create one (it's free).
2. Login to your account and download the dataset.
3. Unzip the data file if needed, then rename it to `spotify.csv`, and move it to the same directory as this notebook.

Exercise 1: Exploratory data analysis

1(a)

rubric={points:2}

Read in the data CSV and store it as a pandas dataframe named `spotify_df`. The first column of the .csv file should be set as the index.

```

```

1(b)

rubric={points:2}

Run the following line of code to split the data. How many training and test examples do we have?

> Note: we are setting the `random_state` so that everyone has the same split on their assignments. This will make it easier for the TAs to grade.

```
df_train, df_test = train_test_split(spotify_df, test_size=0.2, random_state=321)
```

Double-click (or enter) to edit

1(c)

rubric={points:3}

- Print out the output of `describe()` **on the training split**. This will compute some summary statistics of the numeric columns.
- Which feature has the smallest range?

> Hint: You can subtract the min value from the max value of the column to get the range.

Note that `describe` returns another DataFrame.

Double-click (or enter) to edit

1(d)

rubric={points:5}

Let's focus on the following features:

- danceability
- tempo
- energy
- valence

For each of these features (in order), produce a histogram that shows the distribution of the feature values in the training set, **separated for positive and negative examples**. By "positive examples" we mean target = 1 (user liked the song, positive sentiment) and by "negative examples" we mean target = 0 (used disliked the song, negative sentiment). As an example, here is what the histogram would look like for a different feature, loudness:

(You don't have to match all the details exactly, such as colour, but your histograms should look something like this, with a reasonable number of bins to see the shape of the distribution.) As shown above, there are two different histograms, one for target = 0 and one for target = 1, and they are overlaid on top of each other. The histogram above shows that extremely quiet songs tend to be disliked (more blue bars than orange on the left) and very loud songs also tend to be disliked (more blue than orange on the far right).

To adhere to the DRY (Don't Repeat Yourself) principle, make sure you use a `for` loop for your plotting, rather than repeating the plotting code 4 times. For this to work, I used `plt.show()` at the end of your loop, which draws the figure and resets the canvas for your next plot.

Here is some code that separates out the dataset into positive and negative examples, to help you get started:

```
negative_examples = df_train.query("target == 0")
positive_examples = df_train.query("target == 1")
```

Double-click (or enter) to edit

1(e)

rubric={points:4}

Let's say you had to make a decision stump (decision tree with depth 1), *by hand*, to predict the target class. Just from looking at the plots above, describe a reasonable split (feature name and threshold) and what class you would predict in the two cases. For example, in the loudness histogram provided earlier on, it seems that very large values of loudness are generally disliked (more blue on the right side of the histogram), so you might answer something like this: "A reasonable split would be to predict 0 if loudness > -5 (and predict 1 otherwise)."

Double-click (or enter) to edit

1(f)

rubric={points:2}

Let's say that, for a particular feature, the histograms of that feature are identical for the two target classes. Does that mean the feature is not useful for predicting the target class?

1(g)

rubric={points:2}

Note that the dataset includes two free text features labeled `song_title` and `artist`:

```
df_train[["song_title", "artist"]].head()
```

- Do you think these features could be useful in predicting whether the user liked the song or not?
- Would there be any difficulty in using them in your model?

Double-click (or enter) to edit

Exercise 2: Using sklearn to build a decision tree classifier

2(a)

rubric={points:2}

- Create `X_train` and `y_train` and `X_test` and `y_test` from `df_train` and `df_test` above. Skip the `song_title` and `artist` features for now.
- Fit a `DecisionTreeClassifier` on the train set.

```

```

2(b)

rubric={points:2}

Use the `predict` method to predict the class of the first example in your `X_train`. Is the prediction correct? That is, does it match with the corresponding class in `y_train`?

> Hint: you can grab the first example with `X_train.iloc[[0]]`.

Double-click (or enter) to edit

2(c)

rubric={points:2}

Use the `cross_val_score` function on your training set to compute the 10-fold cross-validation accuracy of your tree.

2(d)

rubric={points:2}

The above is useful, but we would like to see the training accuracy as well.

- Compute the 10-fold cross-validation again but this time using the `cross_validate` function with `return_train_score=True`.
- Print out both the cross-validation score and the training score.
- Is your cross-validation score exactly the same as what you got in the previous part? Very briefly discuss.

Double-click (or enter) to edit

2(e)

rubric={points:1}

Do you see a significant difference between the training score and the cross-validation score? Briefly discuss.

Double-click (or enter) to edit

2(f)

rubric={points:1}

Inspect the 10 sub-scores from the 10 folds of cross-validation. How does this inform the trustworthiness of your cross validation score?

Double-click (or enter) to edit

Exercise 3: Hyperparameters

rubric={points:10}

In this exercise, you'll experiment with the `max_depth` hyperparameter of the decision tree classifier. See the `DecisionTreeClassifier` documentation for more details.

- Explore the `max_depth` hyperparameter. Run 10-fold cross-validation for trees with different values of `max_depth` (at least 10 different values in the range 1 to 25).
- For each `max_depth`, get both the train accuracy and the cross-validation accuracy.

- Make a plot with `max_depth` on the *x*-axis and the train and cross-validation scores on the *y*-axis. That is, your plot should have two curves, one for train and one for cross-validation. Include a legend to specify which is which.
- Discuss how changing the `max_depth` hyperparameter affects the training and cross-validation accuracy. From these results, what depth would you pick as the optimal depth?
- Do you think that the depth you chose would generalize to other "spotify" datasets (i.e., data on other spotify users)?

> Note: generally speaking (for all assignments) you are welcome to copy/paste code directly from the lecture notes, though I ask that you add a small citation (e.g. "Adapted from lecture 2") if you do so.

Double-click (or enter) to edit

Exercise 4: Test set

rubric={points:4}

Remember the test set you created way back at the beginning of this assignment? Let's use it now to see if our cross-validation score from the previous exercise is trustworthy.

- Select your favorite `max_depth` from the previous part.
- Train a decision tree classifier using that `max_depth` on the *entire training set*.
- Compute and display the test score.
- How does it compare to the cross-validation score from the previous exercise? Briefly discuss.

Double-click (or enter) to edit

Exercise 5: Conceptual questions

rubric={points:3}

Consider the dataset below, which has 6 examples and 2 features:

$$X = \begin{bmatrix} 5 & 2 \\ 4 & 3 \\ 2 & 2 \\ 10 & 10 \\ 9 & -1 \\ 9 & 9 \end{bmatrix}, \quad y = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}.$$

1. Say we fit a decision stump (depth 1 decision tree) and the first split is on the first feature (left column) being less than 5.5. What would we predict in the "true" and "false" cases

here?

2. What training accuracy would the above stump get on this data set?
3. Can we obtain 100% accuracy with a single decision stump in this particular example?

Double-click (or enter) to edit

Applied Machine Learning

Homework 3: Preprocessing

Table of Contents

Imports

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.dummy import DummyClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import cross_val_score, cross_validate, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

Introduction

A crucial step when using machine learning algorithms on real-world datasets is preprocessing. This homework will give you some practice of data preprocessing and building a supervised machine learning pipeline on a real-world dataset.

Exercise 1: Introducing the dataset

In this lab, you will be working on the adult census dataset. Download the CSV and save it as `adult.csv` locally in this homework folder.

This is a classification dataset and the classification task is to predict whether income exceeds 50K per year or not based on the census data. You can find more information on the dataset and features here.

The starter code below loads the data CSV (assuming that it is saved as `adult.csv` in this folder).

Note that many popular datasets have sex as a feature where the possible values are male and female. This representation reflects how the data were collected and is not meant to imply that, for example, gender is binary.

```
train = pd.read_csv('adult.csv')
```

1.1 Data splitting

rubric={points:4}

In order to avoid violation of the golden rule, the first step before we do anything is splitting the data.

Your tasks:

1. Split the data into `train_df` (60%) and `test_df` (40%) with `random_state = 42`. Keep the target column (`income`) in the splits so that we can use it in the exploratory data analysis.

Usually having more data for training is a good idea. But here I'm using 60%/40% split because this is kind of a big dataset for a modest laptop. A smaller training data means it won't take too long to train the model on your laptop. A side advantage of this would be that with a bigger test split, we'll have a more reliable estimate of the deployment performance!

Exercise 2: Exploratory data analysis (EDA)

Let's examine our `train_df`.

```
train_df.sort_index()
```

We see some missing values represented with a "?". Probably these were the questions not answered by some people during the census. Usually `.describe()` or `.info()` methods would give you information on missing values. But here, they won't pick "?" as missing values as they are encoded as strings instead of an actual NaN in Python. So let's replace them with `np.nan` before we carry out EDA. If you do not do it, you'll encounter an error later on when you try to pass this data to a classifier.

The "?" symbols are now replaced with NaN values.

2.1 Visualizing features

rubric={points:10}

Your tasks:

1. Examine the information given by `train_df_nan.info()` and `train_df_nan.describe()` methods. In case of `.describe()`, use the `include="all"` argument to show summary statistics of all features.
2. Visualize the histograms of numeric features.
3. From the visualizations, which features seem relevant for the given prediction task?

> Note: (Optional) If you're feeling excited about this you are welcome to use `pandas_profiling` for more elaborate visualization and EDA.

2.2 Identify transformations to apply

rubric={points:18}

Your tasks:

1. Identify the sequence of transformations that you would apply on each column in the dataset and fill in the table below accordingly. An example of the sequence of transformations to be applied on the `occupation` feature is shown in the table below. You may decide not to apply any transformations on a certain column or entirely drop a column from your model. That's totally fine.

2. Are there common transformations you would like to apply on certain types of features? Identify different feature types for applying different transformations. In particular, fill in the lists below.
3. Is including the `race` feature for predicting income ethically a good idea? Briefly discuss.

> Note: This question is a bit open-ended and there is no single correct solution.

Feature	Transformation
occupation	imputation, OHE
age	
workclass	
fnlwgt	
education	
education.num	
marital.status	
relationship	
race	
sex	
capital.gain	
capital.loss	
hours.per.week	
native.country	

```
# Fill in the lists below.
# It's OK to keep some of the lists empty or add new lists.
numeric_features = []
categorical_features = []
ordinal_features = []
binary_features = []
drop_features = []
passthrough_features = []
target = "income"
```

Double-click (or enter) to edit

2.3 Separating feature vectors and targets

rubric={points:4}

Your tasks:

1. Create `X_train`, `y_train`, `X_test`, `y_test` from `train_df_nan` and `test_df_nan`.

2. At this point, if you train `sklearn`'s `SVC` model on `X_train` and `y_train` would it work? Why or why not?

Exercise 3: Preprocessing

3.1 Preprocessing using `sklearn`'s `ColumnTransformer` and `Pipeline`

rubric={points:18}

Let's carry out preprocessing using `sklearn`'s `ColumnTransformer` and `Pipeline`. Note that you can define pipelines in two ways:

- by using `Pipeline` and explicitly providing named steps
- by using `make_pipeline`, which automatically names the steps in the pipeline with their class names.

Similarly you can create a column transformer in two ways:

- by using `ColumnTransformer`
- by using `make_column_transformer`

You may use the method of your choice but `make_pipeline` and `make_column_transformer` are highly recommended.

Your tasks:

1. Create a column transformer `preprocessor` based on transformations you want to apply on the data from 2.2.
2. Transform the data by calling `fit_transform` on the training set. What's the shape of the transformed data?
3. Why do we need to use a column transformer in this case? Briefly explain.

Exercise 4: Building models

Now that we have preprocessed features, we are ready to build models. Below, I'm providing the function we used in class which returns mean cross-validation score along with standard

deviation for a given model. Feel free to use it to keep track of your results if you like.

```
results_dict = {} # dictionary to store all the results
```

```
def mean_std_cross_val_scores(model, X_train, y_train, **kwargs):
    """
    Returns mean and std of cross validation

    Parameters
    ----------
    model :
        scikit-learn model
    X_train : numpy array or pandas DataFrame
        X in the training data
    y_train :
        y in the training data

    Returns
    ----------
        pandas Series with mean scores from cross_validation
    """

    scores = cross_validate(model, X_train, y_train, **kwargs)

    mean_scores = pd.DataFrame(scores).mean()
    std_scores = pd.DataFrame(scores).std()
    out_col = []

    for i in range(len(mean_scores)):
        out_col.append((f"%0.3f (+/- %0.3f)" % (mean_scores[i], std_scores[i])))

    return pd.Series(data=out_col, index=mean_scores.index)
```

4.1 Baseline model

rubric={points:6}

Your tasks:

1. Define a pipeline with two steps: `preprocessor` from 3.1 and `scikit-learn`'s `DummyClassifier` with `strategy="prior"` as your classifier.
2. Carry out 5-fold cross-validation with the pipeline. Store the results in `results_dict` above. Display the results as a pandas DataFrame.

> You may use the function `mean_std_cross_val_scores` above to carry out cross-validation and storing results. Refer to the class notes if you are unsure about how

> to use it.

4.2 Trying different classifiers

rubric={points:14}

Your tasks:

1. For each of the models in the starter code below:
    - Define a pipeline with two steps: `preprocessor` from 3.1 and the model as your classifier.
    - Carry out 5-fold cross-validation with the pipeline.
    - Store the results in `results_dict`.
2. Display all the results so far as a pandas dataframe.
3. Compare the train and validation accuracies and `fit` and `score` times in each case. How do the the validation accuracies compare to the baseline model from 4.1? Which model has the best validation accuracy? Which model is the fastest one?

> Note that this might take a while to run.

> You may use the function above `mean_std_cross_val_scores` to carry out cross-validation and storing results. Refer to the class notes if you are unsure about how to use it.

```
models = {
    "decision tree": DecisionTreeClassifier(),
    "kNN": KNeighborsClassifier(),
    "RBF SVM": SVC(),
}
```

```
print("The best model with highest test accuracy is {} of model {}".format(results_datafra
```

(optional) 4.3 Exploring importance of scaling

rubric={points:1}

In this exercise you'll examine whether scaling helps in case of KNNs and SVM RBFs.

Your tasks:

1. Create a column transformer without the `StandardScaler` step for `numeric_features`.
2. Repeat the steps in 4.2 with this new column transformer.
3. Compare the results of scaled numeric features with unscaled numeric features. Is scaling necessary for decision trees? Why or why not?

Double-click (or enter) to edit

4.4 Hyperparameter optimization

rubric={points:10}

In this exercise, you'll carry out hyperparameter optimization for the hyperparameter `C` of SVC RBF classifier. In practice you'll carry out hyperparameter optimization for all different hyperparameters for the most promising classifiers. For the purpose of this assignment, we'll only do it for the `SVC` classifier with one hyperparameter: `C`.

Your tasks:

1. For each `C` value in the `param_grid` in the starter code below:
    - Create a pipeline object with two steps: preprocessor from 3.1 and `SVC` classifier with the value of `C`.
    - Carry out 5-fold cross validation with the pipeline.
    - Store the results in `results_dict` and display results as a pandas DataFrame.
2. Which hyperparameter value seems to be performing the best? Is it different than the default value for the hyperparameter used by `scikit-learn`?

> Note: Running this might take a while.

```
param_grid = {"C": np.logspace(-2, 2, 4)}
```

Exercise 5: Evaluating on the test set

Now that we have a best performing model, it's time to assess our model on the set aside test set. In this exercise you'll examine whether the results you obtained using cross-validation on the train set are consistent with the results on the test set.

5.1 Scoring on the unseen test set

rubric={points:10}

Your tasks:

1. Train the best performing model on the entire training set.
2. Report the results of this model on `X_test`.
3. Are the cross-validation results and test results consistent?

Congratulations on finishing the homework! This was a tricky one but I hope you are feeling good after working on it. You are now ready to build a simple supervised machine learning pipeline on real-world datasets! Well done :clap:!

Applied Machine Learning

Homework 4: Logistic regression, hyperparameter optimization

Imports

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.rcParams["font.size"] = 16

from sklearn.dummy import DummyClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import (
    GridSearchCV,
    cross_val_score,
    cross_validate,
    train_test_split,
)
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.tree import DecisionTreeClassifier
```

Exercise 1: Implementing `DummyClassifier`

---

rubric={points:25}

In this course you will generally **not** be asked to implement machine learning algorihtms (like logistic regression) from scratch. However, this exercise is an exception: you will implement the simplest possible classifier, `DummyClassifier`.

As a reminder, `DummyClassifier` is meant as a baseline and is generally the worst possible "model" you could "fit" to a dataset. All it does is predict the most popular class in the training set. So if there are more 0s than 1s it predicts 0 every time, and if there are more 1s than 0s it predicts 1 every time. For `predict_proba` it looks at the frequencies in the training set, so if you have 30% 0's 70% 1's it predicts `[0.3 0.7]` every time. Thus, `fit` only looks at `y` (not `X`).

Below you will find starter code for a class called `MyDummyClassifier`, which has methods `fit()`, `predict()`, `predict_proba()` and `score()`. Your task is to fill in those four functions. To get your started, I have given you a `return` statement in each case that returns the correct data type: `fit` can return nothing, `predict` returns an array whose size is the number of examples, `predict_proba` returns an array whose size is the number of examples x 2, and `score` returns a number.

The next code block has some tests you can use to assess whether your code is working.

I suggest starting with `fit` and `predict`, and making sure those are working before moving on to `predict_proba`. For `predict_proba`, you should return the frequency of each class in the training data, which is the behaviour of `DummyClassifier(strategy='prior')`. Your `score` function should call your `predict` function. Again, you can compare with `DummyClassifier` using the code below.

To simplify this question, you can assume **binary classification**, and furthermore that these classes are **encoded as 0 and 1**. In other words, you can assume that `y` contains only 0s and 1s. The real `DummyClassifier` works when you have more than two classes, and also works if the target values are encoded differently, for example as "cat", "dog", "mouse", etc.

```
class MyDummyClassifier:
    """
    A baseline classifier that predicts the most common class.
    The predicted probabilities come from the relative frequencies
```

```
    of the classes in the training data.

    This implementation only works when y only contains 0s and 1s.
    """

    def fit(self, X, y):

        return None  # Replace with your code

    def predict(self, X):

        return np.zeros(X.shape[0])  # Replace with your code

    def predict_proba(self, X):

        return np.zeros((X.shape[0], 2))  # Replace with your code

    def score(self, X, y):

        return 0.0  # Replace with your code
```

Below are some tests for `predict` using randomly generated data. You may want to run the cell a few times to make sure you explore the different cases (or automate this with a loop or random seeds).

```
# For testing, generate random data
n_train = 101
n_valid = 21
d = 5
X_train_dummy = np.random.randn(n_train, d)
X_valid_dummy = np.random.randn(n_valid, d)
y_train_dummy = np.random.randint(2, size=n_train)
y_valid_dummy = np.random.randint(2, size=n_valid)

my_dc = MyDummyClassifier()
sk_dc = DummyClassifier(strategy="prior")

my_dc.fit(X_train_dummy, y_train_dummy)
sk_dc.fit(X_train_dummy, y_train_dummy)

assert np.array_equal(my_dc.predict(X_train_dummy), sk_dc.predict(X_train_dummy))
assert np.array_equal(my_dc.predict(X_valid_dummy), sk_dc.predict(X_valid_dummy))
```

Below are some tests for `predict_proba`.

```
assert np.allclose(
    my_dc.predict_proba(X_train_dummy), sk_dc.predict_proba(X_train_dummy)
)
assert np.allclose(
    my_dc.predict_proba(X_valid_dummy), sk_dc.predict_proba(X_valid_dummy)
)
```

Below are some tests for `score`.

```
assert np.isclose(
    my_dc.score(X_train_dummy, y_train_dummy), sk_dc.score(X_train_dummy, y_train_dummy)
)
assert np.isclose(
    my_dc.score(X_valid_dummy, y_valid_dummy), sk_dc.score(X_valid_dummy, y_valid_dummy)
)
```

Exercise 2: Trump Tweets

For the rest of this assignment we'll be looking at a dataset of Donald Trump's tweets as of June 2020. You should start by downloading the dataset. Unzip it and move the file `realdonaldtrump.csv` into this directory. As usual, please do not submit the dataset when you submit the assignment.

```
tweets_df = pd.read_csv("realdonaldtrump.csv", index_col=0)
tweets_df.head()
```

```
tweets_df.shape
```

We will be trying to predict whether a tweet will go "viral", defined as having more than 10,000 retweets:

```
y = tweets_df["retweets"] > 10_000
```

To make predictions, we'll be using only the content (text) of the tweet.

```
X = tweets_df["content"]
```

For the purpose of this assignment, you can ignore all the other columns in the original dataset.

2(a) ordering the steps

rubric={points:8}

Let's start by building a model using `CountVectorizer` and `LogisticRegression`. The code required to do this has been provided below, but in the wrong order.

- Rearrange the lines of code to correctly fit the model and compute the cross-validation score.
- Add a short comment to each block to describe what the code is doing.

```
# YOUR COMMENT HERE
countvec = CountVectorizer(stop_words="english")

# YOUR COMMENT HERE
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=321)

# YOUR COMMENT HERE
cross_val_results = pd.DataFrame(
    cross_validate(pipe, X_train, y_train, return_train_score=True)
)

# YOUR COMMENT HERE
pipe = make_pipeline(countvec, lr)

# YOUR COMMENT HERE
cross_val_results.mean()

# YOUR COMMENT HERE
lr = LogisticRegression(max_iter=1000)
```

```

```

2(b) Cross-validation fold sub-scores

rubric={points:5}

Above we averaged the scores from the 5 folds of cross-validation.

- Print out the 5 individual scores. Reminder: `sklearn` calls them `"test_score"` but they are really (cross-)validation scores.
- Are the 5 scores close to each other or spread far apart? (This is a bit subjective, answer to the best of your ability.)
- How does the size of this dataset (number of rows) compare to the cities dataset we have been using in class? How does this relate to the different sub-scores from the 5 folds?

2(c) baseline

rubric={points:3}

By the way, are these scores any good?

- Run `DummyClassifier` (or `MyDummyClassifier`!) on this dataset.
- Compare the `DummyClassifier` score to what you got from logistic regression above. Does logistic regression seem to be doing anything useful?
- Is it necessary to use `CountVectorizer` here? Briefly explain.

Double-click (or enter) to edit

2(d) probability scores

rubric={points:5}

Here we train a logistic regression classifier on the entire training set:

(Note: this is relying on the `pipe` variable from 2(a) - you'll need to redefine it if you overwrote that variable in between.)

```
pipe.fit(X_train, y_train);
```

Using this model, find the tweet in the **test set** with the highest predicted probability of being viral. Print out the tweet and the associated probability score.

Reminder: you are free to reuse/adapt code from lecture. Please add in a small attribution, e.g. "From Lecture 7".

Double-click (or enter) to edit

2(e) coefficients

rubric={points:4}

We can extract the `CountVectorizer` and `LogisticRegression` objects from the `make_pipeline` object as follows:

```
vec_from_pipe = pipe.named_steps["countvectorizer"]
lr_from_pipe = pipe.named_steps["logisticregression"]
```

Using these extracted components above, display the 5 words with the highest coefficients and the 5 words with the smallest coefficients.

Double-click (or enter) to edit

2(f)

rubric={points:10}

scikit-learn provides a lot of useful tools like `make_pipeline` and `cross_validate`, which are awesome. But with these fancy tools it's also easy to lose track of what is actually happening under the hood. Here, your task is to "manually" (without `Pipeline` and without `cross_validate` or `cross_val_score`) compute logistic regression's validation score on one fold (that is, train on 80% and validate on 20%) of the training data.

You should start with the following `CountVectorizer` and `LogisticRegression` objects, as well as `X_train` and `y_train` (which you should further split):

```
countvec = CountVectorizer(stop_words="english")
lr = LogisticRegression(max_iter=1000)
```

Meta-comment: you might be wondering why we're going into "implementation" here if this course is about *applied* ML. In CPSC 340, we would go all the way down into `LogisticRegression` and understand how `fit` works, line by line. Here we're not going into that at all, but I still think this type of question (and Exercise 1) is a useful middle ground. I do want you to know what is going on in `Pipeline` and in `cross_validate` even if we don't cover the details of `fit`. To get into logistic regression's `fit` requires a bunch of math; here, we're keeping it more conceptual and avoiding all those prerequisites.

Exercise 3: hyperparameter optimization

3(a)

rubric={points:4}

The following code varies the `max_features` hyperparameter of `CountVectorizer` and makes a plot (with the x-axis on a log scale) that shows train/cross-validation scores vs. `max_features`. It also prints the results. Based on the plot/output, what value of `max_features` seems best? Briefly explain.

Note: the code may take a minute or two to run. You can uncomment the `print` statement if you want to see it show the progress.

```
train_scores = []
cv_scores = []

max_features = [10, 100, 1000, 10_000, 100_000]

for mf in max_features:
    #     print(mf)
    pipe = make_pipeline(
        CountVectorizer(stop_words="english", max_features=mf),
        LogisticRegression(max_iter=1000),
    )
    cv_results = cross_validate(pipe, X_train, y_train, return_train_score=True)
    train_scores.append(cv_results["train_score"].mean())
    cv_scores.append(cv_results["test_score"].mean())

plt.semilogx(max_features, train_scores, label="train")
plt.semilogx(max_features, cv_scores, label="valid")
plt.legend()
plt.xlabel("max_features")
plt.ylabel("accuracy");
```

3(b)

rubric={points:4}

The following code varies the `C` hyperparameter of `LogisticRegression` and makes a plot (with the x-axis on a log scale) that shows train/cross-validation scores vs. `C`. Based on the plot, what value of `C` seems best?

Note: the code may take a minute or two to run. You can uncomment the `print` statement if you want to see it show the progress.

3(c)

rubric={points:12}

- Using `GridSearchCV`, jointly optimize `max_features` and `C` across all the combinations of values we tried above.
  - Note: the code might be a bit slow here.
  - Setting `n_jobs=-1` should speed it up if you have a multi-core processor.
  - You can reduce the number of folds (e.g. `cv=2`) to speed it up if necessary.
- What are the best values of `max_features` and `C` according to your grid search?
- Do these best values agree with what you found in parts (a) and (b)?
- Generally speaking, *should* these values agree with what you found in parts (a) and (b)? Explain.

3(d)

rubric={points:5}

- Evaluate your final model on the test set.
- How does your test accuracy compare to your validation accuracy?
- If they are different: do you think this is because you "overfitted on the validation set", or simply random luck?

Exercise 4: Very short answer questions

rubric={points:10}

Each question is worth 2 points. Max 2 sentences per answer.

1. What is the problem with calling `fit_transform` on your test data with `CountVectorizer`?
2. Why is it important to follow the Golden Rule? If you violate it, will that give you a worse classifier?
3. If you could only access one of `predict` or `predict_proba`, which one would you choose? Briefly explain.
4. What are two advantages of using sklearn `Pipeline`s?
5. What are two advantages of `RandomizedSearchCV` over `GridSearchCV`?

Applied Machine Learning

Homework 5: Evaluation metrics

Imports

```
import os
import re
import sys
from hashlib import sha1

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tests_hw5
from sklearn import datasets
from sklearn.compose import make_column_transformer
from sklearn.dummy import DummyClassifier, DummyRegressor
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.linear_model import LogisticRegression, Ridge
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    f1_score,
    make_scorer,
    precision_score,
    recall_score,
)
from sklearn.model_selection import (
    GridSearchCV,
    RandomizedSearchCV,
    cross_val_score,
    cross_validate,
    train_test_split,
)
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, StandardScaler
```

Exercise 1: Precision, recall, and f1 score by hand

Consider the problem of predicting whether a patient has a disease or not. Below are confusion matrices of two machine learning models: Model A and Model B.

- Model A

Actual/Predicted	Predicted disease	Predicted no disease
Actual disease	2	8
Actual no disease	0	100

- Model B

Actual/Predicted	Predicted disease	Predicted no disease
Actual disease	6	4
Actual no disease	10	90

1.1 Positive vs. negative class

rubric={points:2}

Your tasks:

Precision, recall, and f1 score depend upon which class is considered "positive", that is the thing you wish to find. In the example above, which class is likely to be the "positive" class? Why?

1.2 Accuracy

rubric={points:2}

Your tasks:

Calculate accuracies for Model A and Model B.

We'll store all metrics associated with Model A and Model B in the `results_dict` below.

```
results_dict = {"A": {}, "B": {}}
```

```
results_dict["A"]["accuracy"] = None
results_dict["B"]["accuracy"] = None
```

```
assert tests_hw5.ex1_2_1(
    results_dict["A"]["accuracy"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_2_2(
    results_dict["B"]["accuracy"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
pd.DataFrame(results_dict)
```

1.3 Which model would you pick?

rubric={points:1}

Your tasks:

Which model would you pick simply based on the accuracy metric?

1.4 Precision, recall, f1-score

rubric={points:6}

Your tasks:

1. Calculate precision, recall, f1-score for Model A and Model B manually, without using `scikit-learn` tools.

```
results_dict["A"]["precision"] = None
results_dict["B"]["precision"] = None
results_dict["A"]["recall"] = None
results_dict["B"]["recall"] = None
results_dict["A"]["f1"] = None
```

```
results_dict["B"]["f1"] = None
```

```
assert tests_hw5.ex1_4_1(
    results_dict["A"]["precision"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_4_2(
    results_dict["B"]["precision"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_4_3(
    results_dict["A"]["recall"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_4_4(
    results_dict["B"]["recall"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_4_5(
    results_dict["A"]["f1"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

```
assert tests_hw5.ex1_4_6(
    results_dict["B"]["f1"]
), "Your answer is incorrect, see traceback above."
print("Success")
```

Show the dataframe with all results.

```
pd.DataFrame(results_dict)
```

1.5 Discussion

rubric={points:4}

Your tasks:

1. Which metric is more informative in this problem? Why?
2. Which model would you pick based on this information?

(Optional) 1.6

rubric={points:1}

Your tasks:

Provide 4 to 5 example classification datasets (with links) where accuracy metric would be misleading. Discuss which evaluation metric would be more appropriate for each dataset. You may consider datasets we have used in this course so far. You could also look up datasets on Kaggle.

Exercise 2: Classification evaluation metrics using `sklearn`

In general, when a dataset is imbalanced, accuracy does not provide the whole story. In class, we looked at credit card fraud dataset which is a classic example of an imbalanced dataset.

Another example is customer churn datasets. Customer churn refers to the notion of customers leaving a subscription service like Netflix. In this exercise, we will try to predict customer churn in a dataset where most of the customers stay with the service and a small minority cancel their subscription. To start, please download the Kaggle telecom customer churn dataset. Once you have the data, you should be able to run the following code:

The starter code below reads the data CSV as a pandas dataframe and splits it into 70% train and 30% test.

Note that `churn` column in the dataset is the target. "True" means the customer left the subscription (churned) and "False" means they stayed.

> Note that for this kind of problem a more appropriate technique is something called survival analysis and we'll be talking about it later in the course. For now, we'll just treat it as a binary classification problem.

```
df = pd.read_csv("bigml_59c28831336c6604c800002a.csv", encoding="utf-8")
train_df, test_df = train_test_split(df, test_size=0.3, random_state=123)
train_df
```

2.1 Distribution of target values

rubric={points:4}

Your tasks:

Examine the distribution of target values in the train split. Do you see class imbalance? If yes, do we need to deal with it? Why or why not?

(Optional) 2.2 EDA

rubric={points:1}

Your tasks:

Come up with **two** exploratory questions you would like to answer and explore those. Briefly discuss your results in 1-3 sentences.

You are welcome to use `pandas_profiling` (see Lecture 10) but you don't have to.

2.3 Column transformer

rubric={points:10}

The code below creates `X_train`, `y_train`, `X_test`, `y_test` for you. In preparation for building a classifier, set up a `ColumnTransformer` that performs whatever feature transformations you deem sensible. This can include dropping features if you think they are not helpful. Remember that by default `ColumnTransformer` will drop any columns that aren't accounted for when it's created.

In each case, briefly explain your rationale with 1-2 sentences. You do not need an explanation for every feature, but for every group of features that are being transformed the same way. For example, "I am doing transformation X to the following categorical features: `a`, `b`, `c` because of reason Y," etc.

```
X_train = train_df.drop(columns=["churn"])
X_test = test_df.drop(columns=["churn"])

y_train = train_df["churn"]
y_test = test_df["churn"]
```

2.4 Visualizing the transformed data

rubric={points:4}

Fit and transform your `ColumnTransformer` on your training set. Print the first 5 rows of the transformed data as a dataframe (not numpy array). See lecture 10 for code that can get you the new column names after transforming.

2.5 area code feature

rubric={points:4}

The original dataset had a feature called `area code`. Let's assume we encoded this feature with one-hot encoding.

1. The area codes were numbers to begin with. Why do we want to use one-hot encoding on this feature?
2. What were the possible values of `area code`?
3. What new feature(s) were created to replace `area code`?

2.6 Dummy classifier

rubric={points:4}

Your tasks:

Create a `DummyClassifier`. Report the following scoring metrics via cross-validation: accuracy, precision, recall, f1-score. Briefly comment on your results, including any warnings the code produces (2 sentences max).

2.7 Logistic regression

rubric={points:8}

Your tasks:

1. Train and score a logistic regression classifier on the dataset.
2. Report the same metrics as in the previous part.
3. Are you satisfied with the results? Use your `DummyClassifier` results as a reference point. Discuss in a few sentences.

2.8 Logistic regression with `class_weight`

rubric={points:6}

Your tasks:

1. Set the `class_weight` parameter of your logistic regression model to `'balanced'` and report the same metrics as in the previous part.
2. Do you prefer this model to the one in the previous part? Discuss your results in a few sentences.

2.9 Hyperparameter optimization

rubric={points:10}

Now let's tune the hyperparameters of our `LogisticRegression` using `GridSearchCV` to maximize cross-validation f1 score.

Your tasks:

1. Jointly optimize `C` (choose some reasonable values) and `class_weight` (`None` vs. `'balanced'`) with `GridSearchCV` and `scoring="f1"`.
2. What values of `C` and `class_weight` are chosen and what is the best cross-validation f1 score?

2.10 Test results

rubric={points:10}

Your tasks

1. Evaluate the best model on the test set. In particular show each of the following on the test set:
    - Confusion matrix.
    - Classification report.
    - Precision-recall curve with average precision score.
    - ROC curve with AUC.
2. Comment on the results.

> Note that we are not doing it here but in real life, you would also plot confusion matrix, precision-recall curve, and ROC curve on validation data to examine errors and to choose a threshold which works for your operating point.

Exercise 3: Regression metrics

For this exercise, we'll use California housing dataset from `sklearn datasets`. The code below loads the dataset.

```
from sklearn.datasets import fetch_california_housing

housing_df = fetch_california_housing(as_frame=True).frame
```

3.1: Data spitting and exploration

rubric={points:4}

Your tasks:

1. Split the data into train (80%) and test (20%) splits.
2. Explore the train split. Do you need to apply any transformations on the data? If yes, create a preprocessor with the appropriate transformations.
3. Separate `X` and `y` in train and test splits.

3.2 Baseline: DummyRegressor

rubric={points:2}

Your tasks:

1. Carry out cross-validation using `DummyRegressor` with default scoring.
2. What metric is used for scoring by default?

3.3 Different regressors

rubric={points:8}

In this exercise, we are going to use `RandomForestRegressor` model which we haven't looked into yet. At this point you should feel comfortable using models with our usual ML workflow even if you don't know the details. We'll talk about `RandomForestRegressor` later in the course.

The code below defines a custom scorer called `mape_scorer` and creates dictionaries for different regressors (`models`) and different scoring metrics (`score_types_reg`).

Your tasks:

1. Using the `models` and the evaluation metrics `score_types_reg` in the code below, carry out cross-validation with each model, by passing the evaluation metrics to `scoring` argument of `cross_validate`. Use a pipeline with the model as an estimator if you are applying any transformations.
2. Show results as a dataframe.
3. Interpret the results. How do the models compare to the baseline? Which model seems to be performing well with different metrics?

```
def mape(true, pred):
    return 100.0 * np.mean(np.abs((pred - true) / true))
```

```
# make a scorer function that we can pass into cross-validation
mape_scorer = make_scorer(mape, greater_is_better=False)

models = {
    "Ridge": Ridge(),
    "Random Forest": RandomForestRegressor(),
}

score_types_reg = {
    "neg_mean_squared_error": "neg_mean_squared_error",
    "neg_root_mean_squared_error": "neg_root_mean_squared_error",
    "neg_mean_absolute_error": "neg_mean_absolute_error",
    "r2": "r2",
    "mape_scorer": mape_scorer,
}
```

(Optional) 3.4 Hyperparameter optimization

rubric={points:1}

Your tasks:

1. Carry out hyperparameter optimization using `RandomizedSearchCV` and `Ridge` with the following `param_dist`. The `alpha` hyperparameter of `Ridge` controls the fundamental tradeoff. Choose the metric of your choice for hyperparameter optimization.
2. Are you getting better scores compared to the default values?

```
from scipy.stats import loguniform

param_dist = {"ridge__alpha": loguniform(1e-3, 1e3)}
```

3.5 Test results

rubric={points:4}

Your tasks:

1. Try the best model on the test set.
2. Briefly comment on the results. (1 to 2 sentences)

3.6 Model interpretation

rubric={points:4}

Ridge is a linear model and it learns coefficients associated with each feature during `fit()`.

Your tasks:

1. Visualize coefficients learned by the `Ridge` model above as a pandas dataframe with two columns: features and coefficients. If you attempted 3.4, use the `Ridge` model with best hyperparameters. Otherwise use the `Ridge` model with default hyperparameters.
2. Increasing which feature values would result in higher housing price?

Applied Machine Learning

Homework 6: Putting it all together

Table of contents

Imports

```
import os

%matplotlib inline
import sys

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import xgboost as xgb
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
```

```
    f1_score,
    make_scorer,
    plot_confusion_matrix,
)
from sklearn.model_selection import (
    GridSearchCV,
    RandomizedSearchCV,
    cross_val_score,
    cross_validate,
    train_test_split,
)
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, StandardScaler
from sklearn.svm import SVC
```

Introduction

At this point we are at the end of supervised machine learning part of the course. So in this homework, you will be working on an open-ended mini-project, where you will put all the different things you have learned so far together to solve an interesting problem.

A few notes and tips when you work on this mini-project:

Tips

1. This mini-project is open-ended, and while working on it, there might be some situations where you'll have to use your own judgment and make your own decisions (as you would be doing when you work as a data scientist). Make sure you explain your decisions whenever necessary.
2. **Do not include everything you ever tried in your submission** -- it's fine just to have your final code. That said, your code should be reproducible and well-documented. For example, if you chose your hyperparameters based on some hyperparameter optimization experiment, you should leave in the code for that experiment so that someone else could re-run it and obtain the same hyperparameters, rather than mysteriously just setting the hyperparameters to some (carefully chosen) values in your code.
3. If you realize that you are repeating a lot of code try to organize it in functions. Clear presentation of your code, experiments, and results is the key to be successful in this lab. You may use code from lecture notes or previous lab solutions with appropriate attributions.

4. If you are having trouble running models on your laptop because of the size of the dataset, you can create your train/test split in such a way that you have less data in the train split. If you end up doing this, please write a note to the grader in the submission explaining why you are doing it.

Assessment

We plan to grade fairly and leniently. We don't have some secret target score that you need to achieve to get a good grade. **You'll be assessed on demonstration of mastery of course topics, clear presentation, and the quality of your analysis and results.** For example, if you just have a bunch of code and no text or figures, that's not good. If you do a bunch of sane things and get a lower accuracy than your friend, don't sweat it.

A final note

Finally, this style of this "project" question is different from other assignments. It'll be up to you to decide when you're "done" -- in fact, this is one of the hardest parts of real projects. But please don't spend WAY too much time on this... perhaps "a few hours" (2-8 hours???) is a good guideline for a typical submission. Of course if you're having fun you're welcome to spend as much time as you want! But, if so, try not to do it out of perfectionism or getting the best possible grade. Do it because you're learning and enjoying it. Students from the past cohorts have found such kind of labs useful and fun and I hope you enjoy it as well.

1. Understanding the problem

rubric={points:4}

In this mini project, you will be working on a classification problem of predicting whether a credit card client will default or not. For this problem, you will use Default of Credit Card Clients Dataset. In this data set, there are 30,000 examples and 24 features, and the goal is to estimate whether a person will default (fail to pay) their credit card bills; this column is labeled "default.payment.next.month" in the data. The rest of the columns can be used as features. You may take some ideas and compare your results with the associated research paper, which is available through the UBC library.

Your tasks:

1. Spend some time understanding the problem and what each feature means. You can find this information in the documentation on the dataset page on Kaggle. Write a few sentences on your initial thoughts on the problem and the dataset.
2. Download the dataset and read it as a pandas dataframe.

2. Data splitting

rubric={points:2}

Your tasks:

1. Split the data into train and test portions.

3. EDA

rubric={points:10}

Your tasks:

1. Perform exploratory data analysis on the train set.
2. Include at least two summary statistics and two visualizations that you find useful, and accompany each one with a sentence explaining it.
3. Summarize your initial observations about the data.

4. Pick appropriate metric/metrics for assessment.

(Optional) 4. Feature engineering

rubric={points:1}

Your tasks:

1. Carry out feature engineering. In other words, extract new features relevant for the problem and work with your new feature set in the following exercises. You may have to go back and forth between feature engineering and preprocessing.

5. Preprocessing and transformations

rubric={points:10}

Your tasks:

1. Identify different feature types and the transformations you would apply on each feature type.
2. Define a column transformer, if necessary.

6. Baseline model

rubric={points:2}

Your tasks:

1. Try `scikit-learn`'s baseline model and report results.

7. Linear models

rubric={points:12}

Your tasks:

1. Try logistic regression as a first real attempt.
2. Carry out hyperparameter tuning to explore different values for the complexity hyperparameter `C`.
3. Report validation scores along with standard deviation.
4. Summarize your results.

8. Different classifiers

rubric={points:15}

Your tasks:

1. Try at least 3 other models aside from logistic regression. At least one of these models should be a tree-based ensemble model (e.g., lgbm, random forest, xgboost).
2. Summarize your results. Can you beat logistic regression?

(Optional) 9. Feature selection

rubric={points:1}

Your tasks:

Make some attempts to select relevant features. You may try `RFECV` or forward selection. Do the results improve with feature selection? Summarize your results. If you see improvements in the results, keep feature selection in your pipeline. If not, you may abandon it in the next exercises.

10. Hyperparameter optimization

rubric={points:15}

Your tasks:

Make some attempts to optimize hyperparameters for the models you've tried and summarize your results. You may pick one of the best performing models from the previous exercise and tune hyperparameters only for that model. You may use `sklearn`'s methods for hyperparameter optimization or fancier Bayesian optimization methods.

- GridSearchCV
- RandomizedSearchCV
- scikit-optimize

11. Interpretation and feature importances

rubric={points:15}

Your tasks:

1. Use the methods we saw in class (e.g., `eli5`, `shap`) (or any other methods of your choice) to explain feature importances of one of the best performing models. Summarize your observations.

12. Results on the test set

rubric={points:5}

Your tasks:

1. Try your best performing model on the test data and report test scores.
2. Do the test scores agree with the validation scores from before? To what extent do you trust your results? Do you think you've had issues with optimization bias?

(Optional) 13. Explaining predictions

rubric={points:1}

Your tasks

1. Take one or two test predictions and explain them with SHAP force plots.

14. Summary of results

rubric={points:10}

Your tasks:

1. Report your final test score along with the metric you used.
2. Write concluding remarks.
3. Discuss other ideas that you did not try but could potentially improve the performance/interpretability .

Assignment 1: Let the Tensors Flow

In this assignment, you will implement and train your first deep model on a well-known image classification task. You will also familiarize yourself with the Tensorflow (https://www.tensorflow.org/) library we will be using for this course.

General Assignment Notes

- You will need to do some extra reading for these assignments. Sorry, but there is no way around this. This should be significantly reduced by attending the practical exercise sessions, but things are re-iterated here for people who missed them (that's why there's so much text ;)).
- Assignments are posed in a very open-ended manner. Often you only "need" to complete a rather basic task. However, you will get *far* more out of this class by going beyond these basics. Some suggestions for further explorations are usually contained in the assignment description. Ideally though, you should really see what interests *you* and explore those directions further. Share and discuss any interesting findings in class and on Mattermost!
- Please don't stop reading at "Bonus"; see above. Don't be intimidated by all the text; pick something that interests you/lies within your capabilities and *just spend some time on it*.

Setting Up

To work on your own machine

Install Python (https://www.python.org/) (3.x -- depending on your OS you might need to install a not-so-recent version as the newest ones may be incompatible with TF) if you haven't done so, and install Tensorflow 2 (https://www.tensorflow.org/install/). This should be as simple as writing

{% highlight bash %} pip install tensorflow {% endhighlight %}

in your console. If you want GPU support (and have an appropriate GPU), you will need to follow extra steps (see the website). For now, there should be no need since you will usually use your own machine only for development and small tests.

If you want to do everything in Colab (see below), you don't need to install Tensorflow yourself.

Google Colab

Google Colab (https://colab.research.google.com) is a platform to facilitate teaching of machine learning/deep learning. There are tutorials available on-site. Essentially, it is a Jupyter notebook environment with GPU-supported Tensorflow available.

If you want to, you can develop your assignments within this environment. See below for some notes. Notebooks support Markup, so you can also write some text about what your code does, your observations etc. This is a really good idea!

Running code on Colab should be fairly straightforward; there are tutorials available in case you are not familiar with notebooks. There are just some caveats:

- You can check which TF version you are running via `tf.__version__` . Make sure this is 2.x!

- You will need to get external code (like `datasets.py`, see below) in there somehow. One option would be to simply copy and paste the code into the notebook so that you have it locally available. Another would be to run a cell with `from google.colab import files; files.upload()` and choose the corresponding file, this will load it "into the runtime" to allow you to e.g. import from it. Unfortunately you will need to redo this every time the runtime is restarted.
- Later you will need to make data available as well. Since the above method results in temporary files, the best option seems to be to upload them to Google Drive and use `from google.colab import drive; drive.mount('/content/drive')`. You might need to "authenticate" which can be a bit fiddly. After you succeed, you have your drive files available like a "normal" file system. If you find better ways to do this (or the above point), please share them with the class!
- In newer Colab versions, there is actually a button in the "Files" tab to mount the drive -- should be a bit simpler than importing `drive`. The following should work now:

    1. Find the folder in your Google Drive where the notebook is stored, by default this should be `Colab Notebooks`.
    2. Put your data, code (like `datasets.py` linked in one of the tutorials further below) etc. into the same folder (feel free to employ a more sophisticated file structure, but this folder should be your "root").
    3. Mount the drive via the button, it should be mounted into `/content`.
    4. Your working directory should be `content`, verify this via `os.getcwd()`.
    5. Use `os.chdir` to change your working directory to where the notebook is (and the other files as well, see step 2), e.g. `/content/drive/My Drive/Colab Notebooks`.
    6. You should now be able to do stuff like `from datasets import MNISTDataset` in your notebook (see MNIST tutorial further below).

Tensorflow Basics

NOTE: The Tensorflow docs went through significant changes recently. In particular, most introductory articles were changed from using low-level interfaces to high-level ones (particularly Keras). We believe it's better to start with low-level interfaces that force you to program every step of building/training a model yourself. This way, you actually need to understand what is happening in the code. High-level interfaces do a lot of "magic" under the hood. We will proceed to these interfaces after you learn the basics.

Get started with Tensorflow. There are many tutorials on diverse topics on the website, as well as an API documentation (https://www.tensorflow.org/api_docs/python/tf). The following should suffice for now:

- Read up on tensors and operations (https://www.tensorflow.org/tutorials/customization/basics) to get a basic idea of the stuff we are working with (you can ignore the section on Datasets). For more info, check the guide (https://www.tensorflow.org/guide/tensor).
- A special and very important kind of tensor are [variables].
- Automatic differentiation (https://www.tensorflow.org/tutorials/customization/autodiff) is probably the most important concept in Tensorflow!
- See how to fit a simple linear model (https://www.tensorflow.org/guide/basic_training_loops). You may ignore the Keras stuff (second part).

Linear Model for MNIST

MNIST is a collection of handwritten digits and a popular (albeit by now trivialized) benchmark for image classification models. You will see it A LOT.

Go through this basic MNIST tutorial (http://blog.ai.ovgu.de/posts/jens/2019/002_tf20_basic_mnist/index.html) we wrote just for you! It's a logistic (softmax) regression "walkthrough" both in terms of concepts and code. You will of course be tempted to just copy this code; please make sure you understand what each line does.

Play around with the example code snippets. Change them around and see if you can predict what's going to happen. Make sure you understand what you're dealing with!

Building A Deep Model

If you followed the tutorial linked above, you have already built a linear classification model. Next, turn this into a *deep* model by adding a hidden layer between inputs and outputs. To do so, you will need to add an additional set of weights and biases (after having chosen a size for the layer) as well as an activation function.
There you go! You have created a Multilayer Perceptron. **Hint:** Initializing variables to 0 will not work for multilayer perceptrons. You need to initialize values randomly instead (e.g. random_uniform between -0.1 and 0.1). Why do you think this is the case?

Next, you should explore this model: Experiment with different hidden layer sizes, activation functions or weight initializations. See if you can make any observations on how changing these parameters affects the model's performance. Going to extremes can be very instructive here. Make some plots!

Also, reflect on the Tensorflow interface: If you followed the tutorials you were asked to, you have been using a very low-level approach to defining models as well as their training and evaluation. Which of these parts do you think should be wrapped in higher-level interfaces? Do you feel like you are forced to provide any redundant information when defining your model? Any features you are missing so far?

Bonus

There are numerous ways to explore your model some more. For one, you could add more hidden layers and see how this affects the model. You could also try your hand at some basic visualization and model inspection: For example, visualize some of the images your model classifies incorrectly. Can you find out *why* your model has trouble with these?

You may also have noticed that MNIST isn't a particularly interesting dataset -- even very simple models can reach very high accuracy and there isn't much "going on" in the images. Luckily, Zalando Research has developed Fashion MNIST (https://github.com/zalandoresearch/fashion-mnist). This is a more interesting dataset with the *exact same structure* as MNIST, meaning you can use it without changing anything about your code. You can get it by simply using `tf.keras.datasets.fashion_mnist` instead of regular MNIST. You can attempt pretty much all of the above suggestions for this dataset as well!

How to Hand In Your Assignment

In general, you should prepare a notebook (`.ipynb` file) with your solution. The notebook should contain sufficient outputs that show your code working, i.e. **we should not have to run your code to verify that you solved the task**. You can use markdown cells to add some text, e.g. to document interesting observations you made or problems you ran into.

- If you work on Colab, make sure to save your notebooks with outputs! Under Edit -> Notebook settings, make sure the box with "omit code cell output..." is
- *not** ticked.

You can form groups to do the assignments (up to three people). However, **each group member needs to upload the solution separately** (because the Moodle group feature is a little broken). At the very top of the notebook, include the names of all group members!

What to hand in this time

Assignment 2: Let the Tensors Board? & also `tf.data`

Visualizing the learning progress as well as the behavior of a deep model is extremely useful (if not necessary) for troubleshooting in case of unexpected outcomes (or just bad results). In this assignment, you will get to know TensorBoard, Tensorflow's built-in visualization suite, and use it to diagnose some common problems with training deep models. **Note:** TensorBoard seems to work best with Chrome-based browsers. Other browsers may take a *very* long time to load, or not display the data correctly.

Datasets

It should go without saying that loading numpy arrays and taking slices of these as batches (as we did in the last assignment) isn't a great way of providing data to the training algorithm. For example, what if we are working with a dataset that doesn't fit into memory?

The recommended way of handling datasets is via the `tf.data` module. Now is a good time to take some first steps with this module. Read the Programmer's Guide section on this. You can ignore the parts on high-level APIs as well as anything regarding TFRecords and `tf.Example` (we will get to these later) as well as specialized topics involving time series etc. If this is still too much text for you, here is a super short version that just covers building a dataset from numpy arrays (ignore the part where they use Keras ;)). For now, the main thing is that you understand how to do just that.
Then, try to adjust your MLP code so that it uses `tf.data` to provide minibatches instead of the class in `datasets.py`. Keep in mind that you should map the data into the [0,1] range (convert to float!) and convert the labels to `int32` (check the old `MNISTDataset` class for possible preprocessing)!

Here you can find a little notebook that displays some basic `tf.data` stuff (also for MNIST).

Note that the Tensorflow guide often uses the three operations `shuffle`, `batch` and `repeat`. Think about how the results differ when you change the order of these operations (there are six orderings in total). You can experiment with a simple `Dataset.range` dataset. What do you think is the most sensible order?

First Steps with TensorBoard

As before, you will need to do some extra reading to learn how to use TensorBoard. There are several tutorials on the Tensorflow website, accessed via Resources -> Tools. However, they use many high-level concepts we haven't looked at yet to build their networks, so you can find the basics here. This is a modified version of last week's linear model that includes some lines to do TensorBoard visualizations. It should suffice for now. Integrate these lines into your MLP from the last assignment to make sure you get it to work! Basic steps are just:

- Set up a file writer for some log directory.

- During training, run summary ops for anything you are interested in, e.g.
  - Usually scalars for loss and other metrics (e.g. accuracy).
  - Distributions/histograms of layer activations or weights.
  - Images that show what the data looks like.
- Run TensorBoard on the log directory.

Later, we will also see how to use TensorBoard to visualize the *computation graph* of a model.

Finally, check out the github readme for more information on how to use the TensorBoard app itself (first part of the "Usage" section is outdated -- this is not how you create a file writer anymore).

Note: You don't need to hand in any of the above -- just make sure you get TensorBoard to work.

Diagnosing Problems via Visualization

Download this archive containing a few Python scripts. All these contain simple MLP training scripts for MNIST. All of them should also fail at training. For each example, find out through visualization why this is. Also, try to propose fixes for these issues. You may want to write summaries *every* training step. Normally this would be too much (and slow down your program), however it can be useful for debugging.

- These scripts/models are relatively simple -- you should be able to run them on your local machine as long as it's not too ancient. Of course you will need to have the necessary libraries installed.
- Please don't mess with the parameters of the network or learning algorithm before experiencing the original. You can of course use any oddities you notice as clues as to what might be going wrong.
- Sometimes it can be useful to have a look at the inputs your model actually receives. `tf.summary.image` helps here. Note that you need to reshape the inputs from vectors to 28x28-sized images and add an extra axis for the color channel (despite there being only one channel). Check out `tf.reshape` and `tf.expand_dims`.
- Otherwise, it should be helpful to visualize histograms/distributions of layer activations and see if anything jumps out. Note that histogram summaries will crash your program in case of `nan` values appearing. In this case, see if you can do without the histograms and use other means to find out what is going wrong.
- You should also look at the *gradients* of the network; if these are "unusual" (i.e. extremely small or large), something is probably wrong. An overall impression of a gradient's size can be gained via `tf.norm(g)`; feel free to add scalar summaries of these values to TensorBoard. You can pass a `name` to the variables when defining them and use this to give descriptive names to your summaries.
- Some things to watch out for in the code: Are the activation functions sensible? What about the weight initializations? Do the inputs/data look "normal"?
- Note: The final two scripts (4 and 5) may actually work somewhat, but performance should still be significantly below what could be achieved by a "correct" implementation.

What to Hand In

- An MLP training script using `tf.data`.
- A description of what the six `shuffle/batch/repeat` orderings do (on a conceptual level) and which one you think is the most sensible for training neural networks.
- For each "failed" script above, a description of the problem as well as how to fix it (there may be multiple ways). You can just write some text here (markdown cells!), but feel free to reinforce your ideas with some code snippets.
- Anything else you feel like doing. :)

Note that you can only upload a single notebook on Moodle, so use the Markdown features (text cells instead of code) to answer the text-based questions.

Bonus

Like last week, play around with the parameters of your networks. Use Tensorboard to get more information about how some of your choices affect behavior. For example, you could compare the long-term behavior of saturating functions such as *tanh* with *relu*, how the gradients vary for different architectures etc.

If you want to get deeper into the data processing side of things, check the Performance Guide.

Peer Quizzes Follow the registration instructions provided on the theory exercise channel on Mattermost. Contribute PeerQuiz platform with at least

- one question related to the course material and
- one question related to the programming assignment.

Answer and rate as many questions that are posted from peers as you want.

Assignment 3: Keras & CNNs

In this assignment, you will get to know Keras, the high-level API in Tensorflow. You will also create a better model for the MNIST dataset using convolutional neural networks.

Keras

The low-level TF functions we used so far are nice to have full control over everything that is happening, but they are cumbersome to use when we just need "everyday" neural network functionality. For such cases, Tensorflow has integrated Keras to provide abstractions over many common workflows. Keras has *tons* of stuff in it; we will only look at some of it for this assignment and get to know more over the course of the semester. In particular:

- `tf.keras.layers` provides wrappers for many common layers such as dense (fully connected) or convolutional layers. This takes care of creating and storing weights, applying activation functions, regularizers etc.
- `tf.keras.Model` in turn wraps layers into a cohesive whole that allows us to handle whole networks at once.
- `tf.optimizers` make training procedures such as gradient descent simple.
- `tf.losses` and `tf.metrics` allow for easier tracking of model performance.

Unfortunately, none of the TF tutorials are *quite* what we would like here, so you'll have to mix-and-match a little bit:

- This tutorial covers most of what we need, i.e. defining a model and using it in a custom training loop, along with optimizers, metrics etc. You can skip the part about GANs. Overall, the loop works much the same as before, except:
  - You now have all model weights conveniently in one place.
  - You can use the built-in loss functions, which are somewhat less verbose than `tf.nn.sparse_softmax_cross_entropy_with_logits`.
  - You can use `Optimizer` instances instead of manually subtracting gradients from variables.
  - You can use `metrics` to keep track of model performance.
- There are several ways to build Keras models, the simplest one being `Sequential`. For additional examples, you can look at the top of this tutorial, or this one, or maybe this one... In each, look for the part `model = tf.keras.Sequential...`. You just put in a list of layers that will be applied in sequence. Check the API to get an impression of what layers there are and which arguments they take.

Later, we will see how to wrap entire model definitions, training loops and evaluations in a hand-full of lines of code. For now, you might want to rewrite your MLP code with these Keras functions and make sure it still works as before.

An example notebook can be found here.

CNN for MNIST

You should have seen that (with Keras) modifying layer sizes, changing activation functions etc. is simple: You can generally change parts of the model without affecting the rest of the program (training loop etc). In fact, you can change *the full pipeline from input to model output* without having to change anything else (restrictions apply).

Replace your MNIST MLP by a CNN. The tutorials linked above might give you some ideas for architectures. Generally:

- Your data needs to be in the format `width x height x channels`. So for MNIST, make sure your images have shape `(28, 28, 1)`, not `(784,)`!
- Apply a bunch of `Conv2D` and possibly `MaxPool2D` layers.
- `Flatten`.
- Apply any number of `Dense` layers and the final classification (logits) layer.
- Use Keras!
- A reference CNN implementation *without* Keras can be found here!

  Note: Depending on your machine, training a CNN may take *much* longer than the MLPs we've seen so far. Here, using Colab's GPU support could be useful **(Edit -> Notebook settings -> Hardware Accelerator)**. Also, processing the full test set in one go for evaluation might be too much for your RAM. In that case, you could break up the test set into smaller chunks and average the results (easy using keras metrics) -- or just make the model smaller.

You should consider using a better optimization algorithm than the basic `SGD`. One option is to use adaptive algorithms, the most popular of which is called Adam. Check out `tf.optimizers.Adam`. This will usually lead to much faster learning without manual tuning of the learning rate or other parameters. We will discuss advanced optimization strategies later in the class, but the basic idea behind Adam is that it automatically chooses/adapts a per-parameter learning rate as well as incorporating momentum. Using Adam, your CNN should beat your MLP after only a few hundred steps of training. The general consensus is that a well-tuned gradient descent with momentum and learning rate decay will outperform adaptive methods, but you will need to invest some time into finding a good parameter setting -- we will look into these topics later.

If your CNN is set up well, you should reach extremely high accuracy results. This is arguably where MNIST stops being interesting. If you haven't done so, consider working with Fashion-MNIST instead (see Assignment 1). This should present more of a challenge and make improvements due to hyperparameter tuning more obvious/meaningful. You could even try CIFAR10 or CIFAR100 as in one of the tutorials linked above. They have 32x32 3-channel color images with much more variation. These datasets are also available in `tf.keras.datasets`. **Note:** For some reason, the CIFAR labels are organized somewhat differently -- shaped `(n, 1)` instead of just `(n,)`. You should do something like `labels = labels.reshape((-1,))` or this will mess up the loss function.

What to Hand In

- A CNN (built with Keras) trained on MNIST (or not, see below). Also use Keras losses, optimizers and metrics, but do still use a "custom" training loop (with `GradientTape`).

- You are *highly* encouraged to move past MNIST at this point. E.g. switching to CIFAR takes minimal effort since it can also be downloaded through Keras. You can still use MNIST as a "sanity check" that your model is working, but you can skip it for the submission.
- Document any experiments you try. For example:
  - Really do play with the model parameters. As a silly example, you could try increasing your filter sizes up to the input image size -- think about what kind of network you are ending up with if you do this! On the other extreme, what about 1x1 filters?
    - You can do the same thing for pooling. Or replace pooling with strided convolutions. Or...
  - If you're bored, just try to achieve as high of a (test set) performance as you can on CIFAR. This dataset is still commonly used as a benchmark today. Can you get ~97% (test set)?
  - You could try to "look into" your trained models. E.g. the convolutional layers output "feature maps" that can also be interpreted as images (and thus plotted one image per filter). You could use this to try to figure out what features/patterns the filters are recognizing by seeing for what inputs they are most active.

Assignment 4: Graphs & DenseNets

Note: Find the notebook from the exercises here.

Graph-based Execution

So far, we have been using so-called "eager execution" exclusively: Commands are run as they are defined, i.e. writing `y = tf.matmul(X, w)` actually executes the matrix multiplication.

In Tensorflow 1.x, things used to be different: Lines like the above would only *define the computation graph* but not do any actual computation. This would be done later in dedicated "sessions" that execute the graph. Later, eager execution was added as an alternative way of writing programs and is now the default, mainly because it is much more intuitive/allows for a more natural workflow when designing/testing models.

Graph execution has one big advantage: It is very efficient because entire models (or even training loops) can be executed in low-level C/CUDA code without ever going "back up" to Python (which is slow). As such, TF 2.0 still retains the possibility to run stuff in graph mode if you so wish -- let's have a look!

As expected, there is a tutorial on the TF website as well as this one which goes intro extreme depth on all the subtleties. The basic gist is:

- You can annotate a Python function with `@tf.function` to "activate" graph execution for this function.
- The first time this function is called, it will be *traced* and converted to a graph.
- Any other time this function is called, *the Python function will not be run; instead the traced graph is executed*.
- The above is not entirely true -- functions may be *retraced* under certain (important) conditions, e.g. for every new "input signature". This is treated in detail in the article linked above.
- Beware of using Python statements like `print`, these will not be traced so the statement will only be called during the tracing run itself. If you want to print things like tensor values, use `tf.print` instead. Basically, traced TF functions only do "tensor stuff", not general "Python stuff".

Go back to some of your pevious models and sprinkle some `tf.function` annotations in there. You might need to refactor slightly -- you need to actually wrap things into a function!

- The most straightforward target for decoration is a "training step" function that takes a batch of inputs and labels, runs the model, computes the loss and the gradients and applies them.
- In theory, you could wrap a whole training loop (including iteration over a dataset) with a `tf.function`. If you can get this to work on one of your previous models *and actually get a speedup*, you get a cookie. :)

DenseNet

Previously, we saw how to build neural networks in a purely sequential manner -- each layer receives one input and produces one output that serves as input to the next layer. There are many architectures that do not follow this simple scheme. You might ask yourself how this can be done in Keras. One answer is via the so-called functional API. There is an in-depth guide here. Reading just the intro should be enough for a basic grasp on how to use it, but of course you can read more if you wish.

Next, use the functional API to implement a DenseNet. You do *not* need to follow the exact same architecture, in fact you will probably want to make it smaller for efficiency reasons. Just make sure you have one or more "dense blocks" with multiple layers (say, three or more) each. You can also leave out batch normalization (this will be treated later in the class) as well as "bottleneck layers" (1x1 convolutions) if you want.

Bonus: Can you implement DenseNet with the Sequential API? You might want to look at how to implement custom layers (shorter version here)...

What to Hand In

- DenseNet. Thoroughly experiment with (hyper)parameters. Try to achieve the best performance you can on CIFAR10/100.
- For your model(s), compare performance with and without `tf.function`. You can also do this for non-DenseNet models. How does the impact depend on the size of the models?

The next two parts are just here for completeness/reference, to show other ways of working with Keras and some additional TensorBoard functionalities. Check them out if you want -- we will also (try to) properly present them in the exercise later.

Bonus: High-level Training Loops with Keras

As mentioned previously, Keras actually has ways of packing entire training loops into very few lines of code. This is good whenever you have a fairly "standard" task that doesn't require much besides iterating over a dataset and computing a loss/gradients at each step. In this case, you don't need the customizability that writing your own training loops gives you.

As usual, here are some tutorials that cover this:

- The gist is covered in the beginner quickstart: Build the model, compile with an optimizer, a loss and optional metrics and then run `fit` on a dataset. That's it!
- They also have the same thing with a bit more detail.
- The above covers the bare essentials, but you could also look at how to build a CNN for CIFAR10.

There are also some interesting overview articles in the "guide" section but this should suffice for now. Once again, go back to your previous models and re-make them with these high-level training loops! Also, from now on, feel free to run your models like this if you want (and can get it to work for your specific case).

Bonus: TensorBoard Computation Graphs

You can display the computation graphs Tensorflow uses internally in TensorBoard. This can be useful for debugging purposes as well as to get an impression what is going on "under the hood" in your models. More importantly, this can be combined with *profiling* that lets you see how much time/memory specific parts of your model take.

To look at computation graphs, you need to *trace* computations explicitly. See the last part of this guide for how to trace `tf.function` -annotated computations. Note: It seems like you have to do the trace the first time the function is called (e.g. on the first training step).

Assignment 5: Text Classification with RNNs (Part 1)

In this assignment and the next, we are switching to a different modality of data: Text. Namely, we will see how to assign a single label to input sequences of arbitrary length. This has many applications, such as detecting hate speech on social media or detecting spam emails. Here, we will look at sentiment analysis, which is supposed to tell what kind of emotion is associated with a piece of text.

In part 1, we are mainly concerned with implementing RNNs at the low level so that we understand how they work in detail. The models themselves will be rather rudimentary. We will also see the kinds of problems that arise when working with sequence data, specifically text. Next week, we will build better models and deal with some of these issues.

The notebook associated with the practical exercise can be found here.

The Data

We will be using the IMDB movie review dataset. This dataset comes with Keras and consists of 50,000 movie reviews with binary labels (positive or negative), divided into training and testing sets of 25,000 sequences each.

A first look

The data can be loaded the same way as MNIST or CIFAR -- `tf.keras.datasets.imdb.load_data()` . If you print the sequences, however, you will see that they are numbers, not text. Recall that deep learning is essentially a pile of linear algebra. As such, neural networks cannot take text as input, which is why it needs to be converted to numbers. This has already been done for us -- each word has been replaced by a number, and thus a movie review is a sequence of numbers (punctuation has been removed).

If you want to restore the text, `tf.keras.datasets.imdb.get_word_index()` has the mapping -- see the notebook for how you can use this, as well as some additional steps you need to actually get correct outputs.

Representing words

Our sequences are numbers, so they can be put into a neural network. But does this make sense? Recall the kind of transformations a layer implements: A linear map followed by a (optional) non-linearity. But that would mean, for example, that the word represented by index 10 would be "10 times as much" as the word represented by index 1. And if we simply swapped the mapping (which we can do, as it is completely arbitrary), the roles would be reversed! Clearly, this does not make sense.

A simple fix is to use one-hot vectors: Replace a word index by a vector with as many entries as there are words in the vocabulary, where all entries are 0 except the one corresponding to the

respective word, which is 1 -- see the notebook.

Thus, each word gets its own "feature dimension" and can be transformed separately. With this transformation, our data points are now sequences of one-hot vectors, with shape `(sequence_length, vocabulary_size)`.

Variable sequence lengths

Of course, not all movie reviews have the same length. This actually represents a huge problem for us: We would like to process inputs in batches, but tensors generally have to be "rectangular", i.e. we cannot have different sequence lengths in the same batch! The standard way to deal with this is *padding*: Appending additional elements to shorter sequences such that all sequences have the same length.

In the notebook, this is done in a rather crude way: All sequences are padded to the length of the longest sequence in the dataset.

Food for thought #1: Why is this wasteful? Can you think of a smarter padding scheme that is more efficient? Consider the fact that RNNs can work on arbitrary sequence lengths, and that training minibatches are pretty much independent of each other.

Dealing with extremes

Once we define the model, we will run into two issues with our data:

1. Some sequences are very long. This increases our computation time as well as massively hampering gradient flow. It is highly recommended that you limit the sequence length (200 could be a good start). You have two choices:
   A. *Truncate* sequences by cutting off all words beyond a limit. Both `load_data` and `pad_sequences` have arguments to do this. We recommend the latter as you can choose between "pre" or "post" truncation.
   B. *Remove* all sequences that are longer than a limit from the dataset. Radical!
2. Our vocabulary is large, more than 85,000 words. Many of these are rare words which only appear a few times. There are two reasons why this is problematic:

   A. The one-hot vectors are huge, slowing down the program and eating memory.
   B. It's difficult for the network to learn useful features for the rare words.

   `load_data` has an argument to keep only the `n` most common words and replace less frequent ones by a special "unknown word" token (index 2 by default). As a start, try keeping only the 20,000 most common words or so.

Food for thought #2: Between truncating long sequences and removing them, which option do you think is better? Why?

Food for thought #3: Can you think of a way to avoid the one-hot vectors completely? Even if you cannot implement it, a conceptual idea is fine.

With these issues taken care of, we should be ready to build an RNN!

Building The Model

A Tensorflow RNN "layer" can be confusing due to its black box character: All computations over a full sequence of inputs are done internally. **To make sure you understand how an RNN "works", you are asked to implement one from the ground up, defining variables yourself and using basic operations such as `tf.matmul` to define the computations at each time step and over a full input sequence.** There are some related tutorials available on the TF website, but all of these use Keras.

For this assignment, you are asked **not** to use the `RNNCell` classes nor any related Keras functionality. Instead, you should study the basic RNN equations and "just" translate these into code. You can still use Keras optimizers, losses etc. **You can also use `Dense` layers instead of low-level ops, but make sure you know what you are doing.** You might want to proceed as follows:

- On a high level, **nothing about the training loop changes!** The RNN gets an input and computes an output. The loss is computed based on the difference between outputs and targets, and gradients are computed and applied to the RNN weights, with the loss being backpropagated *trough time*.
- The differences come in how the RNN computes its output. The basic recurrency can be seen in equation 10.5 of the deep learning book, with more details in equations 10.8-10.11. The important idea is that, at each time step, the RNN essentially works like an MLP with a single hidden layer, but two inputs (last state and current input). In total, you need to "just":
  - Loop over the input, at each time step taking the respective slice. Your per-step input should be `batch x features` just like with an MLP!
  - At each time step, compute the new state based on the previous state as well as the current input.
  - Compute the per-step output based on the new state.
- What about comparing outputs to targets? Our targets are simple binary labels. On the other hand, we have one output *per time step*. The usual approach is to discard all outputs except the one for the very last step. Thus, this is a "many-to-one" RNN (compare figure 10.5 in the book).
- For the output and loss, you actually have two options:
  1. You could have an output layer with 2 units, and use sparse categorical cross-entropy as before (i.e. softmax activation). Here, whichever output is higher "wins".
  2. You can have *a single* output unit and use binary cross-entropy (i.e. sigmoid activation). Here, the output is usually thresholded at 0.5.

Food for thought #4: How can it be that we can *choose* how many outputs we have, i.e. how can both be correct? Are there differences between both choices as well as (dis)advantages relative to each other?

Open Problems

Initial state

To compute the state at the first time step, you would need a "previous state", but there is none. To fix this, you can define an "initial state" for the network. A common solution is to simply use a

tensor filled with zeros. You could also add a trainable variable and learn an initial state instead!

Food for thought #5: All sequences start with the same special "beginning of sequence" token (coded by index 1). Given this fact, is there a point in learning an initial state? Why (not)?

Computations on padded time steps

Recall that we padded all sequences to be the same length. Unfortunately, the RNN is not aware that we did this. This can be an issue, as we are basically computing new states (thus computing outputs as well as influencing future states) based on "garbage" inputs.

Food for thought #6: `pad_sequences` allows for `pre` or `post` padding. Try both to see the difference. Which option do you think is better? Recall that we use the final time step output from our model.

Food for thought #7: Can you think of a way to prevent the RNN from computing new states on padded time steps? One idea might be to "pass through" the previous state in case the current time step is padding. Note that, within a batch, some sequences might be padded for a given time step while others are not.

Slow learning

Be aware that it might take several thousand steps for the loss to start moving at all, so don't stop training too early if nothing is happening. Experiment with weight initializations and learning rates. For fast learning, the goal is usually to set them as large as possible without the model "exploding".

A major issue with our "last output summarizes the sequence" approach is that the information from the end has to backpropagate all the way to the early time steps, which leads to extreme vanishing gradient issues. You could try to use the RNN output more effectively. Here are some ideas:

- Instead of only using the final output, average (or sum?) the logits (pre-sigmoid) of all time steps and use this as the output instead.
- Instead of the logits, average the *states* at all time steps and compute the output based on this average state. Is this different from the above option?
- Compute logits and sigmoids for each output, and average the per-step probabilities.

Food for thought #8: What could be the advantage of using methods like the above? What are disadvantages? Can you think of other methods to incorporate the full output sequence instead of just the final step?

What to hand in

- A low-level RNN implementation for sentiment classification. If you can get it to move away from 50% accuracy on the training set, that's a success. Be wary of overfitting, however, as this doesn't mean that the model is generalizing! If the test (or validation) loss isn't moving, try using a smaller network. Also note that you may sometimes get a higher test accuracy, while the test loss is *also* increasing (how can this be?)!

- Consider the various questions posed throughout the assignment and try to answer them! You can use text cells to leave short answers in your notebook.

Assignment 6: Text Classification with RNNs (Part 2)

Building on the last assignment, this time we want to iron out some of the issues that were left. In particular:

- less wasteful padding,
- using embeddings to avoid one-hot vectors,
- avoiding computations on padded time steps,
- using Keras to simplify our code,
- using more advanced architectures such as LSTMs, stacked RNNs or bidirectional RNNs.

The notebook associated with the practical exercise can be found here.

Improving training efficiency

Within-batch padding

In the last assignment, we padded all sequences to the longest one in the dataset because we need "rectangular" input tensors. However, at the end of the day, only each *batch* of inputs needs to be the same length. If the longest sequence in a batch has length 150, the other sequences need only be padded to that length, not the longest in the whole dataset!

Thus, if we can find a way to delay the padding after we have formed batches, we can gain some efficiency. Unfortunately, we cannot even create a `tf.data.Dataset.from_tensor_slices` to apply batching to!

Luckily, there are other ways to create datasets. We will be using `from_generator`, which allows for creating datasets from elements returned by arbitrary Python generators. Even better, there is also a `padded_batch` transformation function which batches inputs *and* pads them to the longest length in the batch (what would happen if we tried the regular `batch` method?). See the notebook for a usage example!

Note: Tensorflow also has `RaggedTensor`. These are special tensors allowing different shapes per element. You can find a guide here. You could directly create a dataset `from_tensor_slices` by supplying a ragged tensor, which is arguably easier than using a generator. Unfortunately, ragged tensors are not supported by `padded_batch`. Sad! **However**, many tensorflow operations support ragged tensors, so padding can become unnecessary in many places! You can check the guide for an example with a Keras model. You can try this approach if you want, but for the rest of the assignment we will continue with the padded batches (the ragged version will likely be very slow).

Level 2: Bucketing

There is still a problem with the above approach. In our dataset, there are many short sequences and few very long ones. Unfortunately, it is very likely that all (or most) batches contain at least

one rather long sequence. That means that all the other (short) sequences have to be padded to the long one! Thus, in the worst case, our per-batch padding might only gain us very little. It would be great if there was a way to sort the data such that only sequences of a similar length are grouped in a batch... Maybe there is something in the notebook?

Note: If you truncated sequences to a relatively small value, like 200, bucketing may provide little benefit. The reason is that there will be so many sequences at the exact length 200 that the majority of batches will belong to this bucket. However, if you decide to allow a larger value, say length 500, bucketing should become more and more effective (noticeable via shorter time spent per batch).

Embeddings

Previously, we represented words by one-hot vectors. This is wasteful in terms of memory, and also the matrix products in our deep models will be very inefficient. It turns out, multiplying a matrix with a one-hot vector simply *extracts the corresponding column from the matrix*.

Keras offers an `Embedding` layer for an efficient implementation. Use this instead of the one-hot operation! Note that the layer adds additional parameters, however it can actually result in *fewer* parameters overall if you choose a small enough embedding size (recall the lecture discussion on using linear hidden layers).

RNNs in Keras

Keras offers various RNN layers. These layers take an entire 3d batch of inputs (`batch x time x features`) and return either a complete output sequence, or only the final output time step. There are two ways to use RNNs:

1. The more general is to define a *cell* which implements the per-step computations, i.e. how to compute a new state given a previous state and current input. There are pre-built cells for simple RNNs, GRUs and LSTMs (`LSTMCell` etc.). The cells are then put into the `RNN` layer which wraps them in a loop over time.
2. There also complete classes like `LSTM` which already wrap the corresponding cell.

While the first approach gives more flexibility (we could define our own cells), it is *highly* recommended that you stick with the second approach, as this provides highly optimized implementations for common usage scenarios. Check the docs for the conditions under which these implementations can be used!

Once you have an RNN layer, you can use ist just like other layers, e.g. in a sequential model. Maybe you have an embedding layer, followed by an LSTM, followed by a dense layer that produces the final output. Now, you can easily create stacked RNNs (just put multiple RNN layers one after the other), use `Bidirectional` RNNs, etc. Also try LSTMs vs GRUs!

Masking

One method to prevent new states being computed on padded time steps is by using a *mask*. A mask is a binary tensor with shape `(batch x time)` with 1s representing "real" time steps and 0s representing padding. Given such a mask, the state computation can be "corrected" like this:

```
new_state = mask_at_t * new_state + (1 - mask_at_t) * old_state
```

Where the mask is 1, the new state will be used. Where it is 0, the old state will be propagated instead!

Masking with Keras is almost too simple: Pass the argument `mask_zero=True` to your embedding layer (the constructor, not the call)! You can read more about masking here. The short version is that tensors can carry a mask as an attribute, and Keras layers can be configured to use and/or modify these masks in some way. Here, the embedding layer "knows" to create a mask such that 0 inputs (remember that index 0 encodes padding) are masked as `False`, and the RNN layers are implemented to perform something like the formula above.

Add masking to your model! The result should be much faster learning (in terms of steps needed to reach a particular performance, not time), in particular with `post` padding (the only kind of padding supported by `padded_batch`). The effect will be more dramatic the longer your sequences are.

What to hand in

Implement the various improvements outlined in this assignment. Experiment with adding them one by one and judge the impact (on accuracy, training time, convenience...) of each. You can also carry out "ablation" studies where you take the full model with all improvements, and remove them one at a time to judge their impact.

You can also try using higher or smaller vocabulary sizes and maximum sequence lengths and investigate the impact of these parameters!

Additional notes for custom RNN loops

If for some reason you are not using Keras RNN layers, but rather your own loops over time, there are a few more things to be aware of when using `tf.function`:

1. There seems to be an issue related to data shapes when using `bucket_by_sequence_length` and the final batch in the dataset (which can be smaller than the others). If you receive strange errors about unknown data shapes, you can set `drop_remainder=True`, or use regular `padded_batch` instead of bucketing.
2. A `tf.function` is re-compiled every time it receives an input with a different "signature". This is defined as the shape and data type of the tensor. When every batch has a different sequence length, this causes the training loop to be re-compiled every step. You can fix this by supplying an `input_signature` to `tf.function` -- please check the API docs. You can also pass `experimental_relax_shapes=True` instead, although this seems to be a little less effective.

Assignment 7: Attention-based Neural Machine Translation

In this task, you will implement a simple NMT with attention for a language pair of your choice. We will follow the corresponding TF Tutorial on NMT.

Please do **not** just use the exemplary English-Spanish example to reduce temptation of simply copying the tutorial.
You can find data sets here. We recommend picking a language pair where you understand both languages (so if you do speak Spanish... feel free to use it ;)). This makes it easier (and more fun) for you to evaluate the results. However, keep in mind that some language pairs have a very large amount of examples, whereas some only have very few, which will impact the learning process and the quality of the trained models.

You may run into issues with the code in two places:

1. The downloading of the data inside the notebook might not work (it crashes with a 403 Forbidden error). In that case, you can simply download & extract the data on your local machine and upload the .txt file to your drive, and then mount it and load the file as you've done before.
2. The `load_data` function might crash. It expects each line to result in *pairs of sentences*, but there seems to be a third element which talks about attribution of the example (at least if you download a different dataset from the link above). If this happens, you can use `line.split('\t')[:-1]` to exclude this in the function.

Tasks:

- Follow the tutorial and train the model on your chosen language pair.
- You might need to adapt the preprocessing depending on the language.
- Implement other attention mechanisms and train models with them (there are Keras layers for both):
  - Bahdanau attention (`AdditiveAttention`)
  - Luong's multiplicative attention (`Attention`)

Compare the attention weight plots for some examples between the attention mechanisms. We recommend to add `,vmax=1.0` when creating the plot in `ax.matshow(attention, cmap='viridis')` in the `plot_attention` function so the colors correspond to the same attention values in different plots.

- Do you see qualitative differences in the attention weights between different attention mechanisms?
- Do you think that the model attends to the correct tokens in the input language (if you understand both languages)?

Here are a few questions for you to check how well you understood the tutorial.
Please answer them (briefly) in your solution!

- Which parts of the sentence are used as a token? Each character, each word, or are some words split up?
- Do the same tokens in different language have the same ID?
  e.g. Would the same token index map to the German word `die` and to the English word `die`?
- Is the decoder attending to all previous positions, including the previous decoder predictions?
- Does the encoder output change in different decoding steps?
- Does the context vector change in different decoding steps?
- The decoder uses teacher forcing. Does this mean the time steps can be computed in parallel?
- Why is a mask applied to the loss function?
- When translating the same sentence multiple times, do you get the same result? Why (not)? If not, what changes need to be made to get the same result each time?

Hand in all of your code, i.e. the working tutorial code along with all changes/additions you made. Include outputs which document some of your experiments. Also remember to answer the questions above! Of course you can also write about other observations you made.

Assignment 8: Word2Vec

In this week, we will look at "the" classic model for learning word embeddings. This will be another tutorial-based assignment. Find the link here (https://www.tensorflow.org/tutorials/text/word2vec).

The key points are:

- Getting to know an example of *self-supervised learning*, where we have data without labels, and are constructing a task directly from the data (often some kind of prediction task) in order to learn deep representations,
- Understanding how Softmax with a very large number of classes is problematic, and getting to know possible workarounds,
- Exploring the idea of word embeddings.

Questions for Understanding

As in the last assignment, answer these questions in your submission to make sure you understand what is happening in the tutorial code!

- Given the sentence "I like to cuddle dogs", how many skipgrams are created with a window size of 2?
- In general, how does the number of skipgrams relate to the size of the dataset (in terms of input-target pairs)?
- Why is it not a good idea to compute the full softmax for classification?
- The way the dataset is created, for a given `(target, context)` pair, are the negative samples (remember, these are randomly sampled) the same each time this training example is seen, or are they different?
- For the given example dataset (Shakespeare), would the code create `(target, context)` pairs for sentences that span multiple lines? For example, the last word of one line and the first word of the next line?
- Does the code generate skipgrams for padding characters (index 0)?
- The `skipgrams` function uses a "sampling table". In the code, this is shown to be a simple list of probabilities, and it is created without any reference to the actual text data. How/why does this work? I.e. how does the program "know" which words to sample with which probability?

Possible Improvements & Extensions

- *If* the code generates skipgrams for padding characters: This is probably not a good idea. Can you prevent this from happening?
- *If* the code is not re-drawing negative samples each iteration: Can you change it so that it does? This may give less biased results.
- The candidate sampler may accidentally draw the true context word as one of the negative words. Can you find a way to detect and avoid this? Note that there is `tf.nn.sampled_softmax_loss` which supports such an argument. Using this would require significant re-writing of the code, however (e.g. getting rid of the `uniform_candidate_sampler` entirely).
- One of the most "impressive" features of these word embeddings is that, given a well-trained model, analogies can be performed via vector arithmetic. Try this:
  - Get the learned vectors (either target or context embeddings) for the words `king, queen, man, woman` . Of course, this assumes that these words are present in the training data.
  - Compute the vector `king - man + woman` .

    - Compute the similarity between the resulting vector and *all* word vectors. Which one gives the highest similarity? It "should" be `queen`. Note that it might actually be `king`, in which case `queen` should at least be second-highest. To compute the similarity, you should use cosine similarity.
    - You can try this for other pairs, such as `Paris - France + Germany = Berlin` etc.
- Use a larger vocabulary and/or larger text corpora to train the models. See how embedding quality and training effort changes. You can also implement a version using the "naive" full softmax, and see how the negative sampling increases in efficiency compared to the full version as the vocabulary becomes larger!

Optional: CBOW Model

The tutorial only covers the Skipgram model, however the same paper also proposed the (perhaps more intuitive) *Continuous Bag-of-Words* model. Here instead of predicting the context from the center word, it's the other way around. If you are looking for more of a challenge implementing a model by yourself, the changes should be as follows:

- In CBOW, each training example consists of *multiple* context words and a single target word. There is no equivalent to the `skipgrams` preprocessing function, but you can simply iterate over the full text data in small windows (there is `tf.data.Dataset.window` which may be helpful here) and for each window use the center word as the target and the rest as context.
- The context embedding is computed by embedding all context words separately, and then averaging their embeddings.

The rest stays pretty much the same. You will still need to generate negative examples through sampling, since the full softmax is just as inefficient as with the Skipgram model.

Compare the results of the CBOW model with the Skip

OPTIONAL ASSIGNMENT

Assignment 9: Self-Supervised Learning

In this assignment, we want to explore self-supervised methods for extracting features from unlabeled data, and then use those features for supervised tasks.

General Pipeline

No matter the exact kind of model, we usually do something like this:

1. Define a self-supervised task, such as autoencoding, denoising, predicting neighboring values, filling in blanks...
2. Build a network to solve the task. Often, this will be some kind of encoder-decoder architecture.
3. Train the model.
4. Build a small "classification head" on top of your self-supervised model. If that has an encoder-decoder structure, you will usually discard the decoder and put the classification head on top of the encoder.
5. Train the classification network on labeled data.

Your task

For a dataset of your choice, implement the above pipeline. Try **at least three different kinds** of self-supervised models; for each, train the model and then use the features for a classification task.
Also train a model directly on classification (no pre-training) and compare the performance to the self-supervised models. Also compare the different self-supervision methods with each other.

To make these comparisons fair, your models should have the same number of parameters. E.g. you might want to use the same "encoder" architecture for each task, and add a small classification head on top; then, the network that you train directly on classification should have the same architecture as the encoder and the classification head combined.

The remainder of this text discusses some issues to keep in mind when building autoencoders or similar models.

Autoencoders in Tensorflow

Building autencoders in Tensorflow is pretty simple. You need to define an encoding based on the input, a decoding based on the encoding, and a loss function that measures the distance between decoding and input. An obvious choice may be simply the mean squared error (but see below). To start off, you could try simple MLPs. Note that you are in no way obligated to choose the "reverse" encoder architecture for your encoder; e.g. you could use a 10-layer MLP as an encoder and a single layer as a decoder if you wanted. As a start, you should opt for an "undercomplete" architecture where the encoding is smaller than the data.

Note: The activation function of the last decoder layer is very important, as it needs to be able to map the input data range. Having data in the range [0, 1] allows you to use a sigmoid output activation, for example. Experiment with different activations such as sigmoid, relu or linear (i.e. no) activation and see how it affects the model. Your loss function should also "fit" the output function, e.g. a sigmoid output layer goes well with a binary (!) cross-entropy loss.

Note that you can use the Keras model APIs to build the encoder and decoder as different models, which makes it easy to later use the encoder separately. You can also have sub-models/layers participate in different models at the same time, e.g. an `encoder` model can be part of an `autoencoder` model together with a `decoder`, and of a classification model together with a `classifer_head`.

Convolutional Autoencoders

Next, you should switch to a convolutional encoder/decoder to make use of the fact that we are working with image data. The encoding should simply be one or more convolutional layers, with any filter size and number of filters (you can optionally apply fully-connected layers at the end). As an "inverse" of a `Conv2D`, `Conv2DTranspose` is commonly used. However, you could also use `UpSampling2D` along with regular convolutions. Again, there is no requirement to make the parameters of encoder and decoder "fit", e.g. you don't need to use the same filter sizes. However, you need to take care when choosing padding/strides such that the output has the same dimensions as the input. This can be a problem with MNIST (why?). It also means that the last convolutional (transpose) layer should have as many filters as the input space (e.g. one filter for MNIST or three for CIFAR).

Other models

Even other self-supervised models are often similar to autoencoders. For example, in a denoising autoencoder, the input is a noisy version of the target (so input and target are not the same anymore!), and the loss is computed between the output and this "clean" target. The architecture can remain the same, however.
Similarly, if the input has parts of the image removed and the task is to reconstruct those parts, the target is once again the full image, but an autoencoding architecture would in principle be appropriate once again.

To freeze or not to freeze

Say, you trained some encoder network on a self-supervised task and now build a classification head on top for labeled data. Now you want to train this model. But *which parts* do you actually train? It could be

- only the classification head, leaving the encoder untouched,
- the full network including the encoder,
- the classification head and some part of the encoder, say the last X layers...

There is a trade-off here:

- Allowing the encoder to be "fine-tuned" allows it to learn features that are suited for classification, in case the self-supervised features are not optimal.
- However, this might cause the encoder to overfit on the training set. Training only the classification head would keep the encoder features more general.
- The third option is a compromise between both.

Experiment! You can easily "freeze" layers or whole models by setting their `trainable` argument to `False`.

Pre-training for low supervision scenarios

Self-supervised models are useful in that they can learn from unlabeled data. This can significantly improve performance in settings where large amounts of data are available, but few labels. We can artificially evoke such a situation by just "pretending" that parts of our data has no labels. Try this:

- Train a self-supervised model as before.
- Take a small random subset of the training data. Make sure it is actually random, i.e. all labels are represented. You could go very low, e.g. 100 elements or so.
- Now train a classification net on only this small labeled subset!

As before, compare to a model that is trained directly on the classification task, but only on the labeled subset. If everything works as expected, your self-supervised model should significantly outperform the directly trained one (on the test set)! This is because the direct training massively overfits on the small dataset, whereas the self-supervised model was able to learn features on all available data. You will most likely want to freeze the encoder model, i.e. not fine-tune it -- if you did, the self-supervised model would overfit, as well.