**Chapter 8 - AI in Action II**

Artificial Intelligence (AI) has become a driving force in modern engineering, enhancing productivity, precision, and innovation across a broad range of industries. Engineers have leveraged AI to monitor infrastructure, predict maintenance needs, optimize processes, and even enable autonomous systems. AI is transforming the field of engineering by enabling smarter, more efficient processes across multiple industries. From civil engineering's structural health monitoring to autonomous vehicles and predictive maintenance in mechanical systems, AI continues to enhance productivity, safety, and innovation. As AI technology evolves, its applications in engineering will continue to grow, driving further advancements and shaping the future of how we design, build, and maintain the world around us. Below is a detailed exploration of how AI is applied in different branches of engineering, accompanied by real-life examples illustrating its transformative power.

**1. Civil Engineering (Structural Health Monitoring)**
AI is critical in civil engineering, particularly in monitoring and maintaining the health of large-scale infrastructures, such as bridges, tunnels, and buildings. By integrating machine learning algorithms with sensors embedded in these structures, AI systems can predict failures before they occur, thus preventing catastrophic incidents.

**Example**: Following the tragic collapse of the **Morandi Bridge** in Italy in 2018, efforts to incorporate AI into structural health monitoring systems gained traction. In collaboration with Italian authorities, the **University of Cambridge** developed an AI-powered system to monitor bridges across Italy. Machine learning models analyze real-time sensor data, detecting early signs of deterioration such as cracks or stress points. This proactive maintenance strategy significantly reduces the risk of failures in aging infrastructure and prolongs their lifespan.

**2. Mechanical Engineering (Predictive Maintenance)**
AI has revolutionized mechanical engineering by enabling predictive maintenance, which helps prevent equipment breakdowns and minimize downtime. AI-driven models analyze data from sensors on industrial machines, predicting when components are likely to fail and allowing for timely maintenance. This reduces maintenance costs, improves efficiency, and extends the lifespan of equipment.

**Example**: **General Electric (GE)** employs AI-based predictive maintenance in its aviation division to monitor jet engines. The system uses machine learning algorithms to analyze real-time data, such as temperature, pressure, and vibrations from engines, predicting when components will wear out. According to GE, the introduction of predictive maintenance reduced unexpected engine failures by 30% and slashed maintenance costs by 20%. Similarly, **Rolls-Royce** uses AI to monitor airplane engines, further reducing downtime and repair costs.

**3. Electrical Engineering (Smart Grid Management)**
AI is a cornerstone of smart grid technology, optimizing energy distribution and enhancing the efficiency of power systems. AI systems analyze real-time data from energy consumption patterns, weather conditions, and energy generation sources to predict demand and adjust electricity distribution accordingly. This improves reliability, helps integrate renewable energy sources, and minimizes power outages.

**Example**: The **National Grid** in the UK has successfully implemented AI to optimize electricity flow across its power grids. During the 2018 cold snap, known as the "Beast from the East," AI-enabled systems helped prevent widespread blackouts by analyzing energy usage patterns and adjusting distribution in real-time. These predictive models allow for a more efficient and balanced use of electricity, especially when incorporating renewable energy sources such as wind and solar power.

**4. Chemical Engineering (Process Optimization)**
In chemical engineering, AI enhances process optimization by analyzing complex variables such as temperature, pressure, and chemical reactions in real time. This enables more efficient production, higher yields, and reduced waste. AI-driven simulations also aid in the development of new chemical compounds, accelerating research and development.

**Example**: **BASF**, the world's largest chemical producer, has adopted AI-based process optimization at its Ludwigshafen production plant in Germany. The AI system uses machine learning to monitor production variables, reducing energy consumption by 10%, improving product yield by 5%, and cutting emissions. These optimizations have helped BASF to achieve higher efficiency while minimizing its environmental footprint.

**5. Automotive Engineering (Autonomous Vehicles)**
AI is the backbone of autonomous vehicle technology, which has the potential to transform the automotive industry. Self-driving cars rely on deep learning algorithms to process data from cameras, LiDAR, radar, and GPS systems to make real-time decisions about navigation, collision avoidance, and route optimization.

**Example**: **Tesla's Autopilot** is a well-known application of AI in the automotive industry. The system uses deep learning models to interpret data from various sensors, enabling semi-autonomous driving on highways. Tesla reported that by 2020, over 3 billion miles had been driven using Autopilot, demonstrating the system's reliability and robustness. Additionally, **Waymo**, Google's autonomous vehicle project, has conducted over 20 million miles of self-driving on public roads, further showcasing AI's ability to handle complex driving environments and make real-time decisions.

**6. Aerospace Engineering (AI in Flight Control Systems)**
AI enhances aerospace engineering by improving flight control systems, enabling more precise navigation, optimizing fuel consumption, and enhancing safety. AI models analyze vast amounts of data from sensors on aircraft to assist pilots in making better decisions during flight.

**Example**: **Airbus** has integrated AI into the flight control systems of its A350 aircraft. The AI algorithms analyze data from thousands of sensors to recommend the most fuel-efficient flight paths and adjustments during flight, reducing fuel consumption by up to 2%. These optimizations have saved millions of dollars annually in fuel costs, highlighting AI's critical role in improving operational efficiency.

**7. Environmental Engineering (Climate Modeling and Disaster Prediction)**
AI is a powerful tool in environmental engineering, especially in modeling climate change impacts and predicting natural disasters like floods, hurricanes, and earthquakes. By analyzing satellite data and meteorological inputs, AI models help improve the accuracy of disaster predictions and aid in disaster preparedness and mitigation.

**Example**: The **US National Oceanic and Atmospheric Administration (NOAA)** partnered with Google in 2020 to improve its hurricane prediction models using machine learning. These AI-enhanced models increased the accuracy of hurricane path predictions by 30%, allowing for better preparedness and response. AI's role in climate modeling helps governments and communities better manage resources and reduce the impact of natural disasters.

**8. Biomedical Engineering (AI in Medical Devices)**
In biomedical engineering, AI has become essential in developing intelligent medical devices that enhance diagnostics, surgical procedures, and patient monitoring. AI is also applied in medical imaging, where it aids in the early detection of diseases such as cancer.

**Example**: The **Da Vinci Surgical System**, a robotic surgery system, uses AI to assist surgeons during

complex procedures. The AI system interprets the surgeon's movements, guiding robotic arms to perform surgeries with greater precision, enabling smaller incisions and faster recovery times. To date, over 7.2 million surgeries have been performed using Da Vinci systems, demonstrating AI's significant impact on improving surgical outcomes.

**9. Robotics (Manufacturing Automation)**
AI-powered robots are increasingly used in manufacturing to automate tasks such as assembly, welding, quality control, and packaging. These robots learn from past performance, allowing them to improve their efficiency and accuracy over time, making production processes faster and more reliable.

**Example**: **Foxconn**, the manufacturer of products like the iPhone, has integrated AI-powered robots into its assembly lines. These robots perform various tasks, including assembly and quality inspection. Foxconn reports that AI-driven systems have increased manufacturing efficiency by 30% and reduced defect rates by 20%, leading to more reliable and cost-effective production.

Besides the above examples, AI has found numerous applications in **Building Services Engineering**, improving energy efficiency, safety, comfort, and maintenance of building systems. Here are some key examples of AI applied in building services, along with explanations:

**1. Energy Management and Optimization**
AI is extensively used in optimizing energy consumption within buildings, including heating, ventilation, and air conditioning (HVAC) systems, lighting, and electricity usage. AI-driven energy management systems analyze data from sensors and smart meters to adjust energy use dynamically based on factors such as occupancy, weather conditions, and time of day. This results in improved energy efficiency and reduced operational costs.

**Example**: **Siemens' Navigator Platform** uses AI to monitor and optimize energy use in buildings. It collects data from various building systems, such as HVAC and lighting, and uses machine learning to analyze energy consumption patterns. The system can then automatically adjust temperature settings, lighting levels, or air conditioning based on occupancy, reducing energy waste and cutting costs. Siemens reports up to 30% savings in energy consumption through AI-driven solutions like this.

**2. Smart HVAC Systems**
AI enhances HVAC systems by providing real-time control and predictive maintenance capabilities. Smart thermostats and building automation systems leverage AI algorithms to predict occupancy patterns and weather conditions, ensuring optimal temperature control while reducing energy consumption. AI can also monitor system performance and detect potential failures before they occur, reducing downtime and maintenance costs.

**Example**: **Google's DeepMind** partnered with **Google Data Centers** to apply AI to optimize the cooling systems in their facilities. The AI model reduced the energy needed for cooling by 40%, leading to an overall 15% improvement in energy efficiency. This principle can also be applied in commercial buildings where HVAC systems are major energy consumers.

**3. Predictive Maintenance for Building Equipment**
AI-powered predictive maintenance systems monitor the health of equipment such as elevators, escalators, pumps, and generators by analyzing sensor data and operational parameters. Machine learning models detect anomalies, predict failures, and recommend maintenance schedules, helping to prevent costly equipment breakdowns.

**Example**: **KONE** uses AI for predictive maintenance in elevators and escalators. Their AI-powered 24/7 Connected Services platform collects data from thousands of connected devices, monitoring their status

in real-time. The AI algorithms analyze this data to predict when parts will need servicing, ensuring repairs happen before any significant malfunction, increasing uptime and reliability.

**4. Indoor Air Quality (IAQ) Monitoring and Control**
AI is applied to monitor and improve indoor air quality in buildings. AI-driven systems analyze air quality data in real-time, adjusting ventilation, filtration, and humidification systems to maintain healthy indoor air quality while optimizing energy use.

**Example**: **IBM's Watson AI** has been applied to monitor air quality in commercial buildings. By analyzing sensor data, the AI system can predict air quality issues, automatically adjust ventilation rates, and ensure compliance with indoor air quality standards. These systems are particularly beneficial in smart office buildings, hospitals, and schools where air quality is critical for occupant health.

**5. Lighting Systems and Occupancy Detection**
AI enables intelligent lighting systems that adjust brightness based on occupancy, daylight availability, and user preferences. AI-based occupancy detection systems use sensors and cameras to detect people's presence in different areas of a building, automatically adjusting lighting and other building systems accordingly. This helps reduce energy consumption while ensuring a comfortable environment.

**Example**: **Philips Lighting** uses AI in their smart lighting solutions, which adapt to the presence of people and daylight levels. The system not only saves energy but also provides insights into space usage. Large corporate buildings and hotels have adopted such systems to provide both energy efficiency and enhanced user experiences.

**6. AI-Driven Security Systems**
AI enhances building security through intelligent surveillance systems, which use image recognition and motion detection to identify potential threats. AI algorithms can automatically detect unusual activity, such as unauthorized access or suspicious behavior, and send alerts to security personnel in real-time.

**Example**: **Hikvision**, a leader in video surveillance, uses AI in its security cameras to analyze footage in real-time. AI-enabled cameras can detect intruders, recognize faces, and monitor crowd density. These capabilities are particularly useful in large commercial buildings, shopping malls, and industrial facilities to ensure safety and security.

**7. Smart Building Management Systems (BMS)**
AI integrates with Building Management Systems (BMS) to automate and optimize building operations, including energy management, security, HVAC, and lighting control. By continuously learning from operational data, AI can make better decisions about how to run the building most efficiently, ensuring minimal energy use and maximum comfort.

**Example**: **Johnson Controls** uses AI-driven building management systems that optimize building performance across multiple parameters, such as energy consumption, air quality, and security. The system learns from past operational data, making continuous improvements and adjustments to building systems to maintain optimal performance.

**8. Space Utilization and Occupancy Management**
AI analyzes data from sensors, cameras, and badge systems to monitor how spaces in a building are being used. This information helps facility managers optimize space usage, adjust cleaning schedules, or plan for future expansions based on real usage patterns.

**Example**: **Cisco's Connected Workplace** uses AI to monitor occupancy levels in offices and meeting rooms. The system tracks room usage and automatically adjusts heating, lighting, and ventilation, saving

energy when spaces are unoccupied. Additionally, AI provides insights into how office space is being used, enabling better planning for workspace needs.

**9. AI for Renewable Energy Integration**

As more buildings incorporate renewable energy sources, such as solar panels and wind turbines, AI is used to manage and optimize energy storage and distribution. AI systems predict energy generation and consumption patterns, ensuring efficient use of renewable energy while minimizing reliance on grid power.

**Example**: **Tesla's Powerwall** uses AI to manage energy storage in residential and commercial buildings. AI predicts energy usage and solar panel output, determining when to store excess energy or draw from the battery, thus maximizing the use of renewable energy and reducing dependence on the grid.

**Data Analytics**

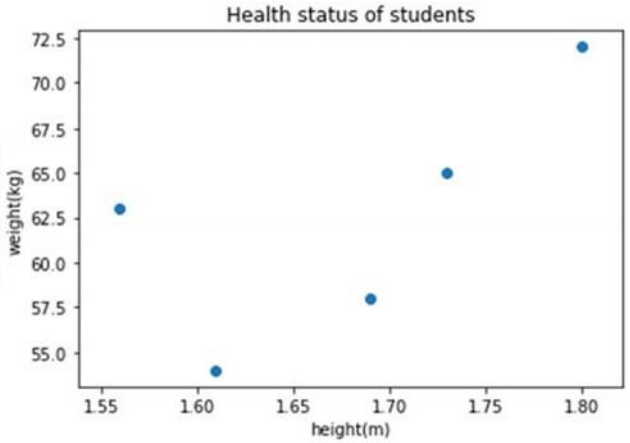
In statistics, **bivariate data** refers to a dataset with two variables. Each sample in this dataset contains values of both variables. A typical example is the health data we studied in a previous chapter. Each student is a sample with two variables height (m) and weight (kg).

```
import pandas as pd
df = pd.read_csv("health.csv")
x = df['height']
y = df['weight']
```

| | sex | height | weight |
|---|---|---|---|
| 0 | M | 1.73 | 65 |
| 1 | F | 1.61 | 54 |
| 2 | M | 1.80 | 72 |
| 3 | F | 1.56 | 63 |
| 4 | F | 1.69 | 58 |

To visualize the relationship between two variables from a bivariate data, we can draw a scatterplot. In a scatterplot, each data point represents a sample. The x-coordinate and y- coordinate represent the value of the two variables.

```
plt.scatter(x,y)
plt.title("Health status of students")
plt.xlabel("height(m)")
plt.ylabel("weight(kg)")
plt.show()
```



Here, we might want to study how the two variables are related to each other using measures of association. To have a rough idea, consider the example above. Do you think that weight and height are uncorrelated? Or do you believe a tall person should be heavier (positive correlation)? Or do you believe a tall person should be lighter (negative correlation)?

In descriptive statistics, we have introduced variance and standard deviation as measures of dispersion, meaning how far the values apart from the mean are. For the variables $x$ and $y$, their variance and standard deviation are $\sigma_x^2$, $\sigma_y^2$ and $\sigma_x$, $\sigma_y$ respectively. In order to measure their association, we further introduce a measure called population covariance:

$$\sigma_{xy} = \frac{\sum(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{N}$$

where $x^{(i)}$, $y^{(i)}$ are the values of variables $x$, $y$ of the $i$-th sample, $\bar{x}$, $\bar{y}$ are the arithmetic means of the two variables, $N$ is the population size.

This is to say, for each sample we evaluate product of difference between each variable with its mean. Notice that this can be positive or negative. The covariance is the average value of these products. Notice that the covariance is commutative, meaning that $\sigma_{xy} = \sigma_{yx}$. Also, the covariance of a variable with its own self results in the variance.

For **sample covariance**, we may simply replace $N$ by $n - 1$ where $n$ is the sample size. In pandas, we can use the function `cov()` to show the sample covariance between two variables, or to give a covariance table.

```
x.cov(y)
```

```
0.43849999999999995
```

```
df.cov()
```

| | height | weight |
|---|---|---|
| **height** | 0.00907 | 0.4385 |
| **weight** | 0.43850 | 47.3000 |

However, the value of covariance also depends on the scale of the variables. For example, if we use centimetre and pounds as the units for height and weight, the covariance will be difference. In order to study the association between two variables without considering scale and unit, we can standardize the covariance by the standard deviation of the two variables. This gives the **correlation coefficient**:

$$r_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

The complete formula is written as:

$$r_{xy} = \frac{\sum(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sqrt{\sum(x^{(i)} - \bar{x})^2}\sqrt{\sum(y^{(i)} - \bar{y})^2}}$$

The result of correlation coefficient $r_{xy}$ is a value between –1 and 1 inclusively. We can interpret the relationship between the two variables by the value of $r_{xy}$:

$r_{xy} = 0$ indicates no linear relationship, or in other words uncorrelated.

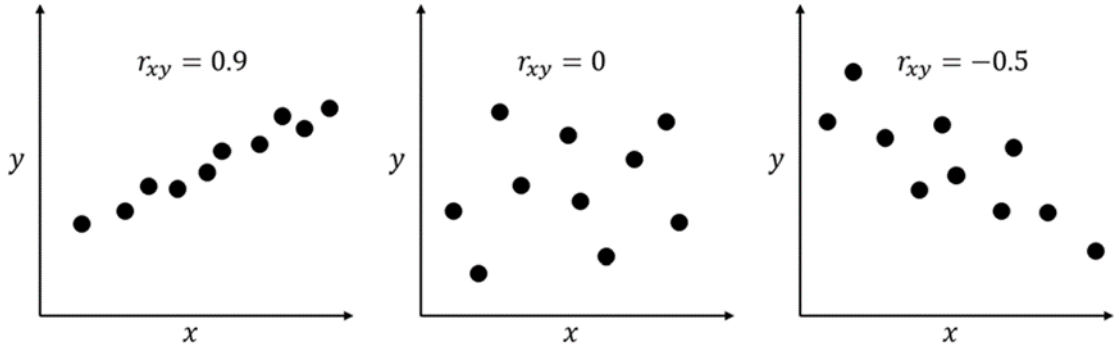
$r_{xy} = 1$ indicates a perfect positive linear relationship: as one variable increases in its values, the other variable also increases in its values through an exact linear rule.

$r_{xy} = –1$ indicates a perfect negative linear relationship: as one variable increases in its values, the other variable decreases in its values through an exact linear rule.

Other than these special cases, we can also distinguish the linear relationship between the variables as follows:

| | weak | moderate | strong |
|---|---|---|---|
| positive | $0 < r_{xy} < 0.3$ | $0.3 < r_{xy} < 0.7$ | $0.7 < r_{xy} < 1$ |
| negative | $-0.3 < r_{xy} < 0$ | $-0.7 < r_{xy} < -0.3$ | $-1 < r_{xy} < -0.7$ |

To illustrate this idea, compare the three figures below. In the left figure, we can easily draw a straight line with positive slope with the points lying close to it. In the right figure, we can also draw a line with negative slope, but the points are farther apart from it. However, for the middle figure we can hardly draw a linear relationship between the points.




We can evaluate the coefficient of correlation using pandas easily. Let's go back to the example of health data of five students. Read the csv file as a DataFrame first. We can then regard height and weight as the two variables as two Series `x` and `y`. We can evaluate the correlation coefficient by `corr()`. Notice that this operation is commutative meaning that `x.corr(y)` and `y.corr(x)` give the same result. The coefficient 0.669 indicates that height and weight are (moderate) positively correlated.

```
x.corr(y)
```

```
0.6694765721676758
```

```
df.corr()
```

| | height | weight |
|---|---|---|
| **height** | 1.000000 | 0.669477 |
| **weight** | 0.669477 | 1.000000 |

In case there are more than two variables in the dataset, it would be inconvenient to compare every pair of them. Instead, we might apply `corr()` to the whole DataFrame. This will give a table of correlation coefficient between each column pairwisely. Notice that non-numerical variables (e.g. sex) is ignored. The table is symmetry with the diagonal values equal to 1 since the correlation coefficient of a variable with itself must be 1.

To further illustrate the idea of correlation between different variables, we will study a famous example. In 1978, David Harrison Jr. and Daniel L. Rubinfeld published a paper called "Hedonic housing prices and the demand for clean air". To support their findings, they referred to the data for census tracts in the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. This dataset is clean with lots of variables including the following:

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940

8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk – 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's

This dataset has been store in "boston.csv". It contains 506 rows and 14 columns. We can first read it as a DataFrame.

```
df = pd.read_csv("boston.csv")
```

```
df.head()
```

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 | 4.98 | 24.0 |
| 1 | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 | 9.14 | 21.6 |
| 2 | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| 3 | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| 4 | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 | 5.33 | 36.2 |

As this dataframe contains quite a number of variables, we can show correlations between each pair of variables in table form.

```
df.corr()
```

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRIM | 1.000000 | -0.200469 | 0.406583 | -0.055892 | 0.420972 | -0.219247 | 0.352734 | -0.379670 | 0.625505 | 0.582764 | 0.289946 | -0.385064 | 0.455621 | -0.388305 |
| ZN | -0.200469 | 1.000000 | -0.533828 | -0.042697 | -0.516604 | 0.311991 | -0.569537 | 0.664408 | -0.311948 | -0.314563 | -0.391679 | 0.175520 | -0.412995 | 0.360445 |
| INDUS | 0.406583 | -0.533828 | 1.000000 | 0.062938 | 0.763651 | -0.391676 | 0.644779 | -0.708027 | 0.595129 | 0.720760 | 0.383248 | -0.356977 | 0.603800 | -0.483725 |
| CHAS | -0.055892 | -0.042697 | 0.062938 | 1.000000 | 0.091203 | 0.091251 | 0.086518 | -0.099176 | -0.007368 | -0.035587 | -0.121515 | 0.048788 | -0.053929 | 0.175260 |
| NOX | 0.420972 | -0.516604 | 0.763651 | 0.091203 | 1.000000 | -0.302188 | 0.731470 | -0.769230 | 0.611441 | 0.668023 | 0.188933 | -0.380051 | 0.590879 | -0.427321 |
| RM | -0.219247 | 0.311991 | -0.391676 | 0.091251 | -0.302188 | 1.000000 | -0.240265 | 0.205246 | -0.209847 | -0.292048 | -0.355501 | 0.128069 | -0.613808 | 0.695360 |
| AGE | 0.352734 | -0.569537 | 0.644779 | 0.086518 | 0.731470 | -0.240265 | 1.000000 | -0.747881 | 0.456022 | 0.506456 | 0.261515 | -0.273534 | 0.602339 | -0.376955 |
| DIS | -0.379670 | 0.664408 | -0.708027 | -0.099176 | -0.769230 | 0.205246 | -0.747881 | 1.000000 | -0.494588 | -0.534432 | -0.232471 | 0.291512 | -0.496996 | 0.249929 |
| RAD | 0.625505 | -0.311948 | 0.595129 | -0.007368 | 0.611441 | -0.209847 | 0.456022 | -0.494588 | 1.000000 | 0.910228 | 0.464741 | -0.444413 | 0.488676 | -0.381626 |
| TAX | 0.582764 | -0.314563 | 0.720760 | -0.035587 | 0.668023 | -0.292048 | 0.506456 | -0.534432 | 0.910228 | 1.000000 | 0.460853 | -0.441808 | 0.543993 | -0.468536 |
| PTRATIO | 0.289946 | -0.391679 | 0.383248 | -0.121515 | 0.188933 | -0.355501 | 0.261515 | -0.232471 | 0.464741 | 0.460853 | 1.000000 | -0.177383 | 0.374044 | -0.507787 |
| B | -0.385064 | 0.175520 | -0.356977 | 0.048788 | -0.380051 | 0.128069 | -0.273534 | 0.291512 | -0.444413 | -0.441808 | -0.177383 | 1.000000 | -0.366087 | 0.333461 |
| LSTAT | 0.455621 | -0.412995 | 0.603800 | -0.053929 | 0.590879 | -0.613808 | 0.602339 | -0.496996 | 0.488676 | 0.543993 | 0.374044 | -0.366087 | 1.000000 | -0.737663 |
| MEDV | -0.388305 | 0.360445 | -0.483725 | 0.175260 | -0.427321 | 0.695360 | -0.376955 | 0.249929 | -0.381626 | -0.468536 | -0.507787 | 0.333461 | -0.737663 | 1.000000 |

In particular, we would like to study how is the target variable median house price (MEDV) being correlated to two factors average room number (RM) and proportional of old buildings (AGE). First we can extract these columns from the DataFrame as three Series.

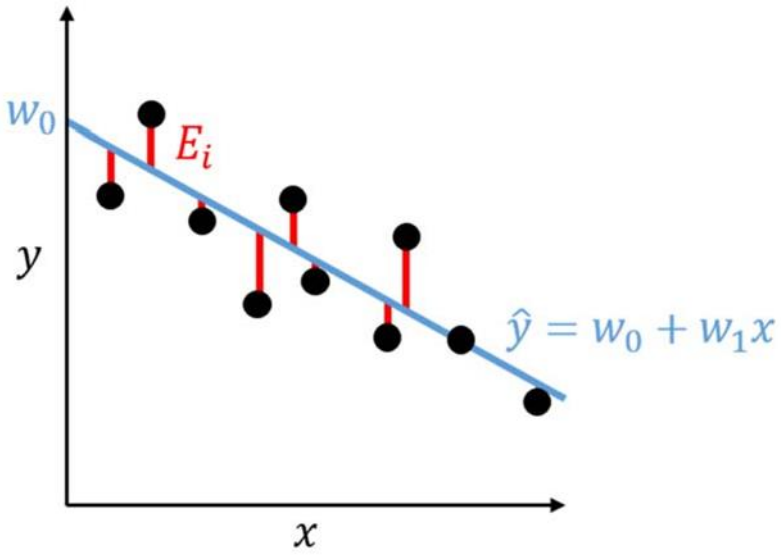
```
x1 = df['RM']
x2 = df['AGE']
x3 = df['MEDV']
```

To visualize the data, make a scatter plot for each pair of variables. We can see that the house price is positively correlated to the average room number, meaning that in general more rooms result in higher price. However, it is negatively correlated to the proportional of old buildings, meaning that for more old buildings in the district the house price is more likely to be lower.

```
plt.scatter(x1,x3)
plt.title("house price vs room number")
plt.xlabel("average room number")
plt.ylabel("median house price")
plt.show()
```


```
plt.scatter(x2,x3)
plt.title("house price vs age")
plt.xlabel("proportion of old buildings")
plt.ylabel("median house price")
plt.show()
```


In the previous section, we have introduced the correlation coefficient for measuring the association between two variables. The value of this coefficient suggests whether it is a linear relationship between the two variables. To further analyze the association and make reasonable prediction, we can use a statistical technique called **linear regression**. For simplicity, we are looking for a straight line that best fit the data set of two variables. Recall the figure in the previous section. We can draw a straight line graphically to fit the points. In general, we need to know how to find the equation of such line, and how to evaluate the performance of the line fitting the points.




We define $\hat{y}$ as the predicted value, which is a linear function $x$ given by:

$$\hat{y} = w_0 + w_1 x$$

where the weights $w_0$, $w_1$ represents the y-intercept and slope of the line. For the $i$-th sample, we define the error $E_i$ as the difference between the predicted value $\hat{y}^{(i)}$ and the actual value $y^{(i)}$. Our target is to find the weights with the least **sum of square error** (SSE). In other words, try to minimize:

$$\text{SSE} = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

In case the target variable $y$ is correlated to several variable, we can express the predicted value as the following expression giving a plane or hyperplane rather than a line.

$$\hat{y} = w_0 + w_1 x_1 + \cdots + w_m x_m$$

In such situation, we need to use a multiple regression model.

## 7.3 Building a linear regression model

In scikit-learn, there is a linear regression model using the least square error method. We will revisit the Boston house price example to illustrate the techniques of such model. As we have found from the previous session, the average number of rooms (RM) has the highest positive correlation with the house price (MEDV). To create a simple regression model, we regard RM as one column in X and MEDV as y, extract the related columns from the DataFrame. Notice that X is a DataFrame which might contain one or more than one columns. For a simple linear regression model, we are only using one variable. On the other hand, y is a Series which is the only target variable to be predicted.

```
import pandas as pd
df = pd.read_csv('boston.csv')
X = df[['RM']]
y = df['MEDV']
```

Create a linear regression model by `LinearRegression` in scikit-learn, then use the fit method to train the model using the variable X and target y.

```
slr = LinearRegression()
slr.fit(X, y)
```

After evaluation, the slope $w_1$ and the y-intercept $w_0$ can be found by `coef_` and `intercept_` under the object. Notice that `coef_` is an array of numbers as there can be more than one variables $w_1$, $w_2$, $w_3$, … in a multiple regression model.

```
slr.intercept_
```

```
-34.670620776438554
```

```
slr.coef_
```

```
array([9.10210898])
```

Therefore we have $w_0 \approx -34.67$, $w_1 \approx 9.1$. The regression line is given by:

$$\hat{y} = -34.67 + 9.1x$$

where $\hat{y}$ is the predicted median house price and $x$ is the average number of rooms.

With this regression model, we can evaluate all the predict values $\hat{y}^{(i)}$ directly using `predict()`:

```
y_pred = slr.predict(X)
```

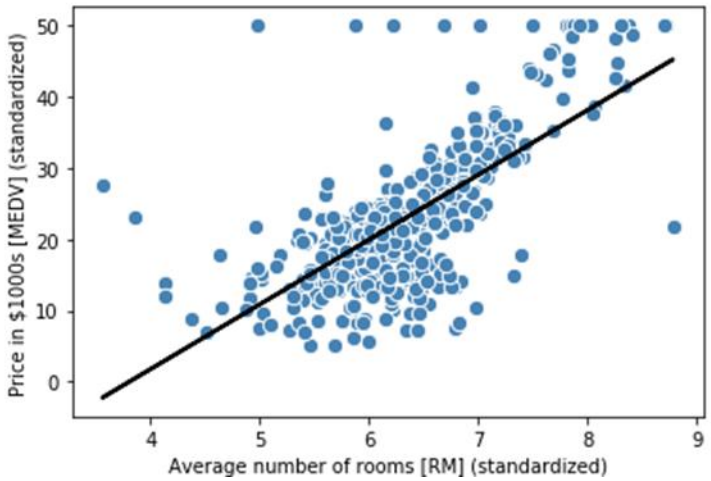
One may also make prediction of a single value. For example, the expected price (in $1000) of a house with 5 rooms is given by:

```
slr.predict([[5]])
```

```
array([10.83992413])
```

To visualize the data and also the regression, show the line graph and scatter plot:

```
import matplotlib.pyplot as plt
plt.scatter(X, y, c='steelblue', edgecolor='white', s=70)
plt.plot(X, slr.predict(X), color='black', lw=2)
plt.xlabel('Average number of rooms [RM] (standardized)')
plt.ylabel('Price in $1000s [MEDV] (standardized)')
plt.show()
```



Performance of linear regression model

In the previous section, we have introduced how to fit a regression model on training data. However, just as training a classification model, it is crucial to test the model on data that it hasn't seen during training to obtain a more unbiased estimate of its generalization performance. We will use scikit-learn to split the dataset into train data and test data, and compare their performance.

With train_test_split we can split both X and y into train data and test data:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)
```

The linear regression model is then trained with only the train data:

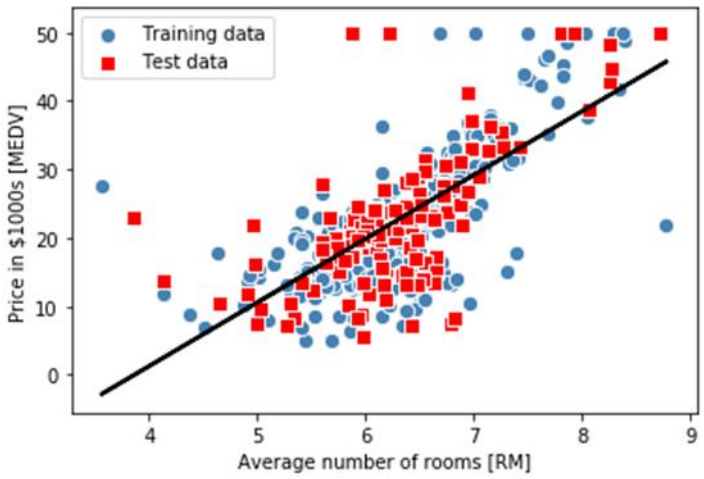
```
from sklearn.linear_model import LinearRegression
slr2 = LinearRegression()
slr2.fit(X_train, y_train)
```

After the model is trained, apply it to X_train and X_test separately to make prediction:

```
y_train_pred = slr2.predict(X_train)
y_test_pred = slr2.predict(X_test)
```

To compare, create scatter plot of the train data and test data with different styles, and also the line graph of the regression model on the same figure.

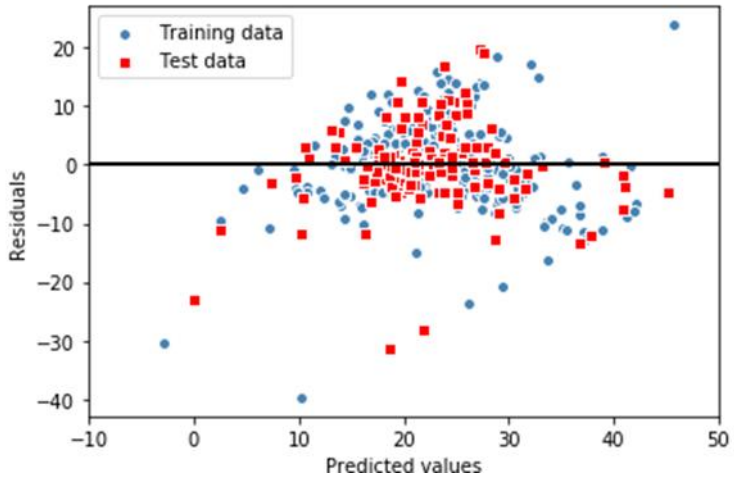
```
plt.scatter(X_train, y_train, c='steelblue', marker='o',
            edgecolor='white', s=70, label='Training data')
plt.scatter(X_test, y_test, c='red', marker='s',
            edgecolor='white', s=70, label='Test data')
plt.plot(X_train, y_train_pred, color='black', lw=2)
plt.legend(loc='upper left')
plt.xlabel('Average number of rooms [RM]')
plt.ylabel('Price in $1000s [MEDV]')
plt.show()
```




The figure, however, is not very effective in visualizing the performance of the model. Moreover, if it is a multiple regression model with variables $x_1$, $x_2$, $x_3$, … it would be impossible to visualize it on a 2D figure.

To visualize the performance of a regression model, we can use **residual plot**. The **residual** of a point refers to the difference between its actual and predicted values, i.e. $\hat{y}^{(i)} - y^{(i)}$. It is also referred as **error** of the prediction at a point. Notice that the residual can be positive or negative, mean that the model over-estimate or under-estimate the actual value. The case of zero residual means the prediction about this point is exact. Residual plot refers to the scatter plot of the residuals against predicted values. Despite the number of variables in X, the residual plot is always a 2D visualization.

```
plt.scatter(y_train_pred,  y_train_pred - y_train,
            c='steelblue', marker='o', edgecolor='white',
            label='Training data')
plt.scatter(y_test_pred,  y_test_pred - y_test,
            c='red', marker='s', edgecolor='white',
            label='Test data')
plt.xlabel('Predicted values')
plt.ylabel('Residuals')
plt.legend(loc='upper left')
plt.hlines(y=0, xmin=-10, xmax=50, color='black', lw=2)
plt.xlim([-10, 50])
plt.show()
```




In the case of a perfect prediction, the residuals would be exactly zero, which we will probably never encounter in realistic and practical applications. However, for a good regression model, we would expect the errors to be randomly distributed and the residuals to be randomly scattered around the centreline. If we see patterns in a residual plot, it means that our model is unable to capture some explanatory information, which has leaked into the residuals.

Another useful quantitative measure of a model's performance is the **mean squared error (MSE)** which is simply the averaged value of the SSE. The MSE is useful for comparing different regression models or for tuning their parameters via grid search and cross- validation, as it normalizes the SSE by the sample size:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

Compute MSE of the train and test predictions in our example:

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_train, y_train_pred)
```

```
42.15765086312225
```

```
mean_squared_error(y_test, y_test_pred)
```

```
47.03304747975518
```

We can see the MSE on the training dataset is significantly larger than the MSE on the test dataset. This indicates our model is overfitting the training data. However, please be aware that the MSE is unbounded in contrast to the classification accuracy, for example. In other words, the interpretation of the MSE depends on the dataset and feature scaling. For example, if the house prices were presented as multiples of 1,000 (with the K suffix), the same model would yield a lower MSE compared to a model that worked with unscaled features.

Thus, it may sometimes be more useful to report the coefficient of determination ($R^2$), which can be understood as a standardized version of the MSE, for better interpretability of the model's performance. Or, in other words, $R^2$ is the fraction of response variance that is captured by the model. The mathematical expression is as follows:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \bar{y})^2} = 1 - \frac{\text{MSE}}{Var(y)}$$

Notice that $R^2$ is a value in between 0 to 1. If the error is smaller, value of $R^2$ become bigger. In case of perfect fit, all the errors are 0 resulting $R^2 = 1$.

To computer $R^2$ using scikit-learn:

```
from sklearn.metrics import r2_score
```

```
r2_score(y_train, y_train_pred)
```

```
0.5026497630040827
```

```
r2_score(y_test, y_test_pred)
```

```
0.43514364832115193
```

We can see that the value of $R^2$ of the train data is higher than that of the test data. This suggests our regression model gives less error for the train data.