```
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 = 21,
                                                    stratify=y)

knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.861
```

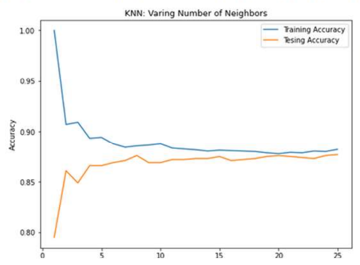
```
import matplotlib.pyplot as plt


train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1,26)
print(neighbors)
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

#print(train_accuracies.values())
print(test_accuracies.values())
my_train = list(train_accuracies.values())
my_test = list(test_accuracies.values())

plt.figure(figsize=(8,6))
plt.title("KNN: Varing Number of Neighbors")
plt.plot(neighbors, my_train, label="Training Accuracy")
plt.plot(neighbors, my_test, label="Tesing Accuracy")
plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 2
 25]
dict_values([0.795, 0.861, 0.849, 0.866, 0.866, 0.869, 0.871,
0.871, 0.872, 0.873, 0.875, 0.876, 0.875, 0.874, 0.873, 0.876,
```



```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

sales_df = pd.read_csv('sales_df.csv')

X = sales_df.drop(columns=['sales'])
y = sales_df['sales']

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Predicted values:", y_pred[:2])
print("Actual target :", y_test[:2])

r_squared = r2_score(y_test, y_pred)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print()

print("R-squared:", r_squared)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Predicted values: [53099.56399301 71056.14674591]
Actual target : [55261.28 67574.9 ]

R-squared: 0.9990147957135925
Root Mean Squared Error (RMSE): 2945.0531856107264
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LinearRegression

sales_df = pd.read_csv('sales_df.csv')

X = sales_df[['radio', 'social_media']]
y = sales_df['sales']

kf = KFold(n_splits=6, shuffle=True, random_state=5)

reg = LinearRegression()
cv_scores = cross_val_score(reg, X, y, cv=kf)

print("CV scores: ", cv_scores)

mean_score = np.mean(cv_scores)
print("Mean:", mean_score)

std_score = np.std(cv_scores)
print("Std:", std_score)

confidence_interval = np.quantile(cv_scores, [0.025, 0.975]
print("95% Confidence Interval:", confidence_interval)
```

```
CV scores:  [0.74451678 0.77241887 0.76842114 0.7410406  0.
Mean: 0.7536937414361207
Std: 0.012305389070474664
95% Confidence Interval: [0.74141863 0.77191916]
```

What is statistics?

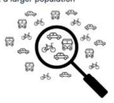
- The field of statistics – the practice and study of collecting and analyzing data.
- A summary statistics – a fact about or summary of some data.
- What can statistics do?
  - How likely is someone to purchase a product? Are people more likely to purchase it if they can use a different payment system?
  - How many occupants will your hotel have? How can you optimize occupancy?
  - How many sizes of jeans need to be manufactured so they can fit 95% of the population? Should the same number of each size be produced?
  - A/B tests: Which ad is more effective in getting people to purchase a product?

Types of statistics

Descriptive statistics
- *Describe* and summarize data

- 50% of friends drive to work
- 25% take the bus
- 25% bike

Inferential statistics
- Use a sample of data to make *inferences* about a larger population



What percent of people drive to work?

41

Being able to identify data types is important since the type of data you're working with will dictate what kinds of summary statistics and

Calculating variance

3. Sum squared distances

```
sum_sq_dists = np.sum(sq_dists)
print(sum_sq_dists)
```

```
1624.065542
```

4. Divide by number of data points - 1

```
variance = sum_sq_dists / (83 - 1)
print(variance)
```

```
19.805677
```

Use `np.var()`

```
np.var(msleep['sleep_total'], ddof=1)
```

```
19.805677
```

*Without `ddof=1`, **population variance** is calculated instead of **sample variance**:*

```
np.var(msleep['sleep_total'])
```

```
19.567055
```

```
np.sqrt(np.var(msleep['sleep_total'], ddof=1))
```

```
4.450357
```

```
np.std(msleep['sleep_total'], ddof=1)
```

```
4.450357
```

```
np.quantile(msleep['sleep_total'], 0.5)
```

```
10.1
```

```
# 75th percentile
seventy_fifth = salaries["Salary_USD"].quantile(0.75)

# 25th percentile
twenty_fifth = salaries["Salary_USD"].quantile(0.25)

# Interquartile range
salaries_iqr = seventy_fifth - twenty_fifth

print(salaries_iqr)
```

Measuring chance

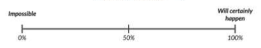
- We can measure the chances of an event using probability. We can calculate the probability of some event by taking the number of ways the event can happen and dividing it by the total number of possible outcomes.
- For example, if we flip a coin, it can land on either heads or tails. To get the probability of the coin landing on heads, we divide the 1 way to get heads by the two possible outcomes, heads and tails.

What's the probability of an event?

$$P(\text{event}) = \frac{\text{\# ways event can happen}}{\text{total \# of possible outcomes}}$$

Example: a coin flip

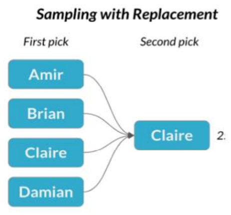
$$P(\text{heads}) = \frac{\text{1 way to get heads}}{\text{2 possible outcomes}} = \frac{1}{2} = 50\%$$



Independent events

*Two events are **independent** if the probability of the second event **isn't** affected by the outcome of the first event.*

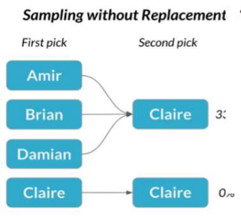
Sampling with replacement = each pick is independent



Dependent events

*Two events are **dependent** if the probability of the second event **is** affected by the outcome of the first event.*

Sampling without replacement = each pick is dependent



Central Limit Theorem in action

- The central limit theorem states that a sampling distribution of a sample statistic approaches the normal distribution as you take more samples, no matter the original distribution being sampled from.
- In this exercise, you'll focus on the sample mean and see the central limit theorem in action while examining the num_users column of amir_deals more closely, which contains the number of people who intend to use the product Amir is selling.

Types of data

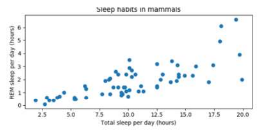
Numeric (Quantitative)
- **Continuous (Measured)**
  - Airplane speed
  - Time spent waiting in line
- **Discrete (Counted)**
  - Number of pets
  - Number of packages shipped

Categorical (Qualitative)
- **Nominal (Unordered)**
  - Married/unmarried
  - Country of residence
- **Ordinal (Ordered)**
  - ○ Strongly disagree
  - ○ Somewhat disagree
  - ○ Neither agree nor disagree
  - ● Somewhat agree
  - ○ Strongly agree

Categorical data can be represented as numbers

Nominal (Unordered)
- Married/unmarried (`1` / `0`)
- Country of residence (`1` , `2` , ...)

Ordinal (Ordered)
- Strongly disagree (`1`)
- Somewhat disagree (`2`)
- Neither agree nor disagree (`3`)
- Somewhat agree (`4`)
- Strongly agree (`5`)



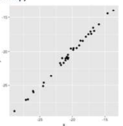
Correlation coefficient

- Quantifies the linear relationship between two variables
- Number between -1 and 1
- Magnitude corresponds to strength of relationship
- Sign (+ or -) corresponds to direction of relationship

- x = explanatory/independent variable
- y = response/dependent variable

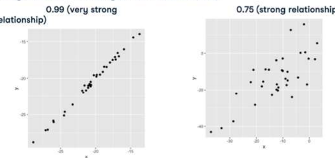
Magnitude = strength of relationship

0.99 (very strong relationship)



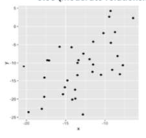
Magnitude = strength of relationship

0.99 (very strong relationship) 0.75 (strong relationship)



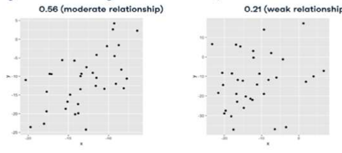
Magnitude = strength of relationship

0.56 (moderate relationship)



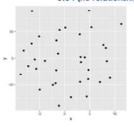
Magnitude = strength of relationship

0.56 (moderate relationship) 0.21 (weak relationship)



Magnitude = strength of relationship

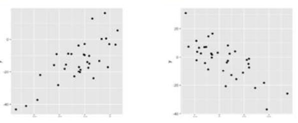
0.04 (no relationship)



- Knowing the value of `x` doesn't tell us anything about `y`

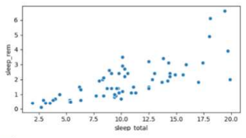
Sign = direction

0.75: as `x` increases, `y` increases *-0.75: as `x` increases, `y` decreases*



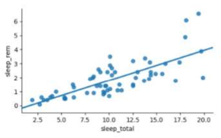
Visualizing relationships

```
import seaborn as sns
sns.scatterplot(x="sleep_total", y="sleep_rem", data=msleep)
plt.show()
```



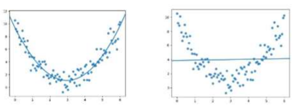
Adding a trendline

```
import seaborn as sns
sns.lmplot(x="sleep_total", y="sleep_rem", data=msleep, ci=None)
plt.show()
```



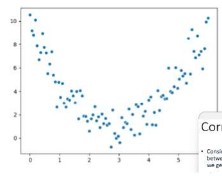
Non-linear relationships

What we see: *What the correlation coefficient sees:*



Correlation caveats

- Consider the graph. There is clearly a relationship between x and y, but when we calculate the correlation, we get 0.18.
- This is because the relationship between the two variables is a quadratic relationship, not a linear relationship. The correlation coefficient measures the strength of linear relationships, and linear relationships only.



Correlation

Log transformation

- When data is highly skewed like this, we can apply a log transformation.
- We'll create a new column called log_bodywt which holds the log of each body weight. We can do this using np.log. If we plot the log of bodyweight versus awake time, the relationship looks much more linear than the one between regular bodyweight and awake time. The correlation between the log of bodyweight and awake time is about 0.57, which is much higher than the 0.3 we had before.

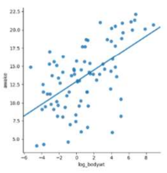
Log transformation

```
msleep['log_bodywt'] = np.log(msleep['bodywt'])

sns.lmplot(x='log_bodywt',
           y='awake',
           data=msleep,
           ci=None)
plt.show()
```

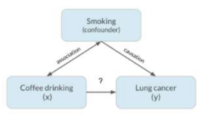
```
msleep['log_bodywt'].corr(msleep['awake'])
```

```
0.5687943
```



A phenomenon called confounding can lead to spurious correlations. Let's say we want to know if drinking coffee causes lung cancer. Looking at the data, we find that coffee drinking and lung cancer are correlated, which may lead us to think that drinking more coffee will give you lung cancer.

Confounding



However, there is a third, hidden variable at play, which is smoking. It is also known that smoking causes lung cancer.

In reality, it turns out that coffee does not cause lung cancer and is only associated with it, but it appeared causal due to the third variable, smoking. This third variable is called a confounder, or lurking variable. This means that the relationship of interest between coffee and lung cancer is a spurious correlation.