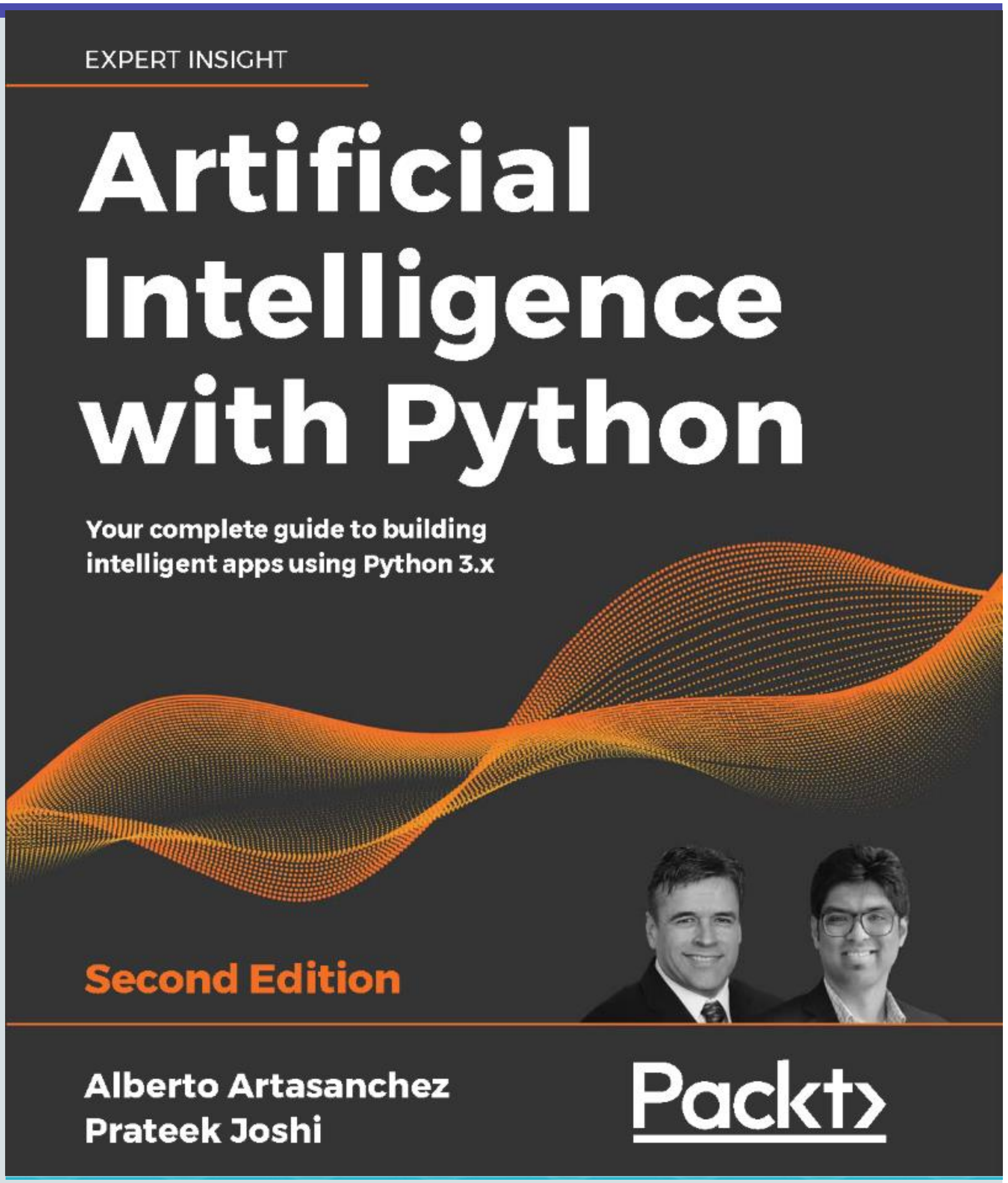
University of South-Eastern Norway
School of Business

Lecture 5:Chatbots

Sinuo Wu

Course: AI for Business Applications (AI3000)


EXPERT INSIGHT

Artificial Intelligence with Python

Your complete guide to building intelligent apps using Python 3.x

Second Edition

Alberto Artasanchez
Prateek Joshi

Packt

University of South-Eastern Norway
School of Business

University of South-Eastern Norway
School of Business

Do it before we start:

1. **Install Unity Hub and Editor within Unity Hub**

2. **Install Visual Studio**

Not necessary if you have another tool to write #C script

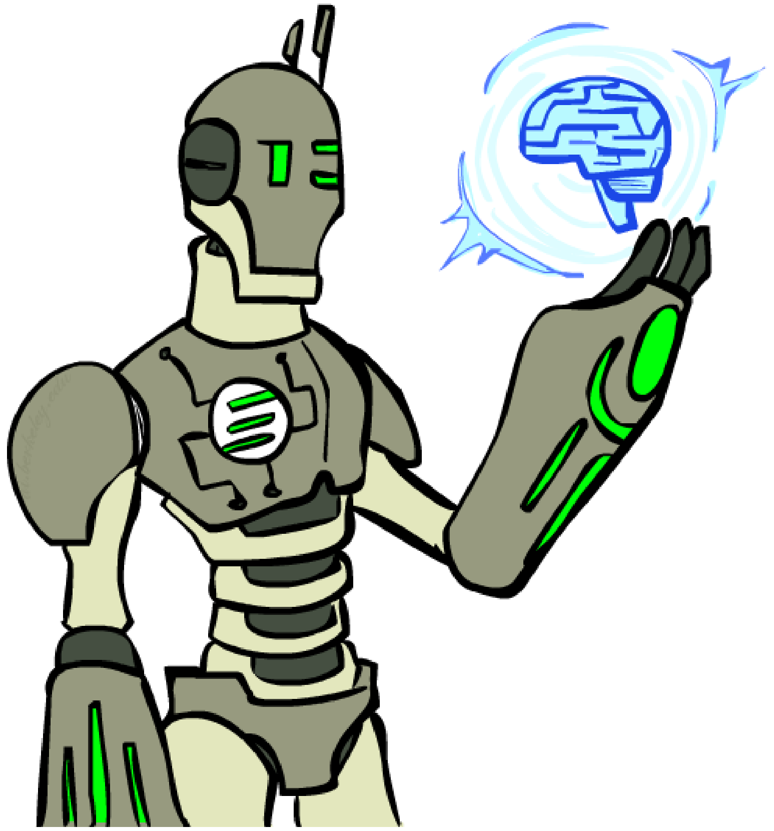
Quiz

- Enter room number or Scan the QR code

Today

- Chatbot
- Introduction to ChatGPT
- Develop Flappy Bird game with ChatGPT
- Review some cases with ChatGPT

- Coding with the help of ChatGPT



Chatbots

University of
South-Eastern Norway
School of Business

Related Concepts

- **Agent** – A system
- **Intents** – The reason as why people start chatting
- **Context** – Environment, objects, actors and process
- **Entities** – Groups a set of words under a defined entity
- **Integrations** – Integrate with conversation and messaging platforms
- **Fulfillment** – Connecting service

Related Concepts

- **Utterances** – Different ways of description
- **Wake words** – Such as “hi, Siri”
- **Launch words** – Order, Tell me, Add, Turn on, Polish…
- **Slot values** – Words that will be converted into parameters
- **Error Planning** – Default behavior for unforeseen cases
- **Webhooks** – Reserve API

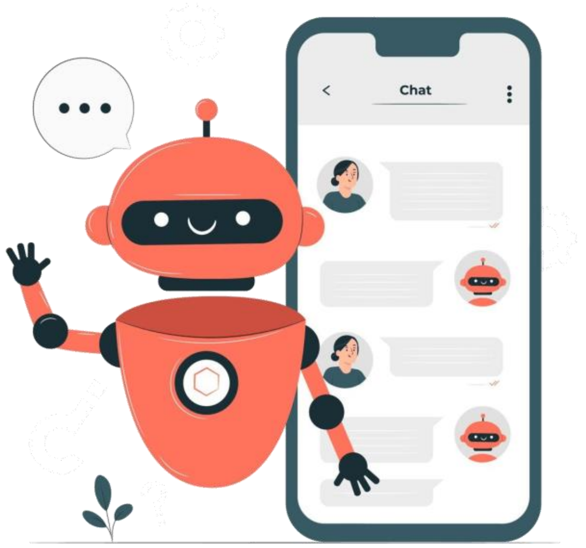
Chatbot Platforms

- Dialog Flow (Google)
- Azure Bot Service (Microsoft)
- Lex (AWS)
- Wit.ai (Facebook)
- Watson (IBM)
- ChatGPT (Open AI)

Create a Chatbot

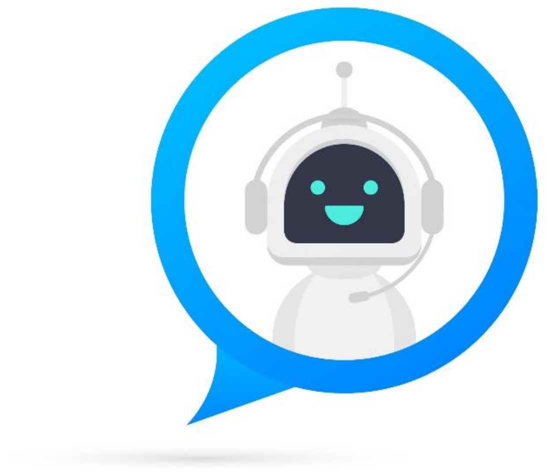
- **Step 1.** Sign up with your google account and setup the Dialog flow
- **Step 2.** Integrate your chatbot into a website via Python (or widget)
- **Step 3.** Set up a webhook
- **Step 4.** Enable webhooks for intents
- **Step 5.** Set up training phrases, parameters and actions for an intent
- **Step 6.** Building fulfillment responses from a webhook
- **Step 7.** Checking responses from a webhook

Examples



AI Response

Smart home

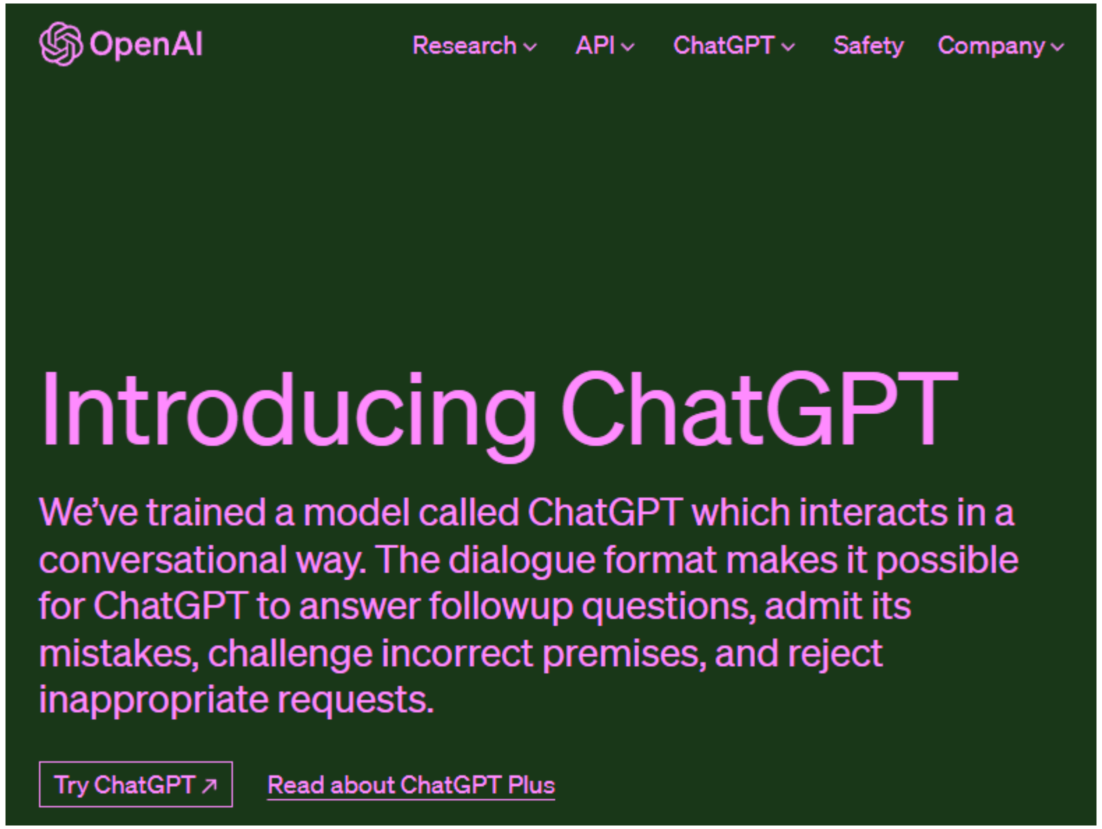


AI Phone call

University of South-Eastern Norway
School of Business

ChatGPT

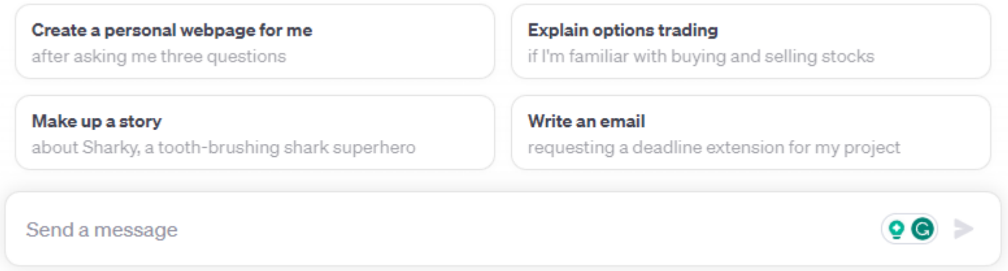
Introduction



Introducing ChatGPT

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.

- ChatGPT is a cutting-edge AI language model developed by OpenAI.
- It is designed to understand and generate human-like text, making it a useful tool for various applications.



Create a personal webpage for me
after asking me three questions

Explain options trading
if I'm familiar with buying and selling stocks

Make up a story
about Sharky, a tooth-brushing shark superhero

Write an email
requesting a deadline extension for my project

Capabilities

- **Natural Language Understanding**: ChatGPT can comprehend and interpret human language, enabling it to answer questions, engage in conversations, and provide information on a wide range of topics.
- **Content Generation**: It can generate human-like text for tasks like content creation, creative writing, and drafting emails or reports.
- **Translation**: ChatGPT can translate text between multiple languages, facilitating cross-lingual communication and content localization.
- **Text Summarization**: It can summarize long documents or articles, extracting the most important information efficiently.
- **Programming Assistance**: Developers can use ChatGPT for coding help, code generation, and debugging assistance in various programming languages.
- **Knowledge Base**: ChatGPT has access to a vast amount of information, up to its knowledge cutoff date in September 2021, making it a valuable resource for research and factual inquiries.
- **Conversational Agent**: It can serve as a chatbot, engaging in natural, human-like conversations and providing customer support, information, or entertainment.
- **Customization**: Users can fine-tune ChatGPT for specific tasks and domains, tailoring its responses to meet their needs.

How it works

Data: ChatGPT is trained on a vast text dataset. (Dataset update in September 2021)

Algorithms: Deep learning, neural networks, and natural language processing.

Tokenization: Text is broken into tokens for numerical processing.

Transformer: Utilizes the Transformer architecture for sequence tasks.

Training: Learns language via predicting the next word.

Fine-Tuning: Tailored for specific tasks if needed.

Inference: Generates text based on user prompts.

Iterative Learning: Improves with user interactions and updates.

Limitations

- **Lack of True Understanding**: ChatGPT doesn't truly understand context or facts; it generates responses based on patterns.
- **Inaccuracy**: It can provide incorrect or nonsensical information.
- **Sensitivity to Input**: Responses may vary based on phrasing, and it doesn't fact-check.
- **Inappropriate Content**: Risk of generating offensive or harmful content.
- **Need for Supervision**: Human oversight is necessary to ensure safe and unbiased responses.
- **Knowledge Limitation**: It lacks real-time information and updates. (Updated Sep 2021)

Use Cases

Group 1. Cooperate with ChatGPT to write a poem or a song, you decide the topic, length, and emotion.

Group 2. Cooperate with ChatGPT to write advertising for Hønefoss. You should describe this city to ChatGPT first.

Note: ChatGPT will pick up one representative from each group to read the result.

Ethics

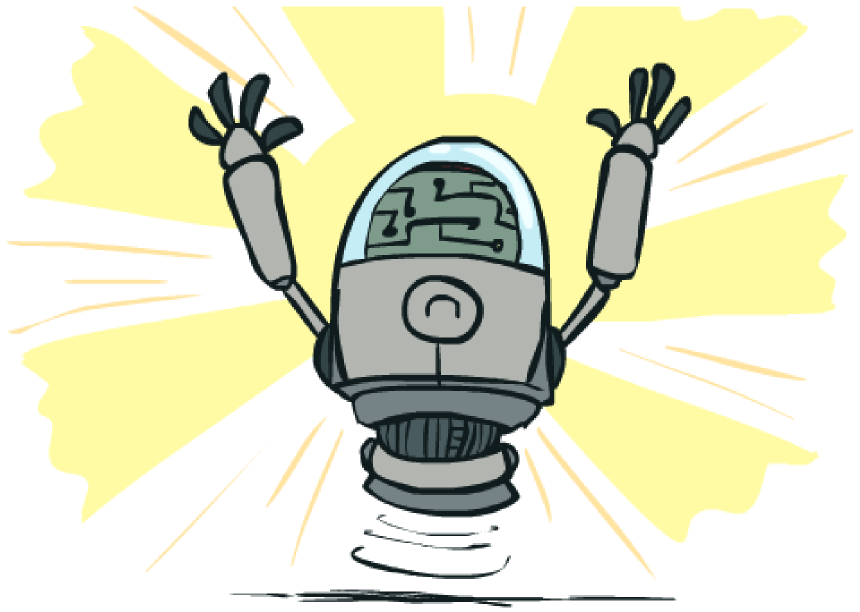
We will discuss it at our last lecture

Have a Break!

University of South-Eastern Norway
School of Business

Practice

Develop Flappy Bird Game



University of South-Eastern Norway
School of Business

Inspiration

https://www.youtube.com/watch?v=TXqJUufquns

Inspiration

https://www.youtube.com/watch?v=L6vW0um5XTg

Flappy Bird

What do you do as a designer?

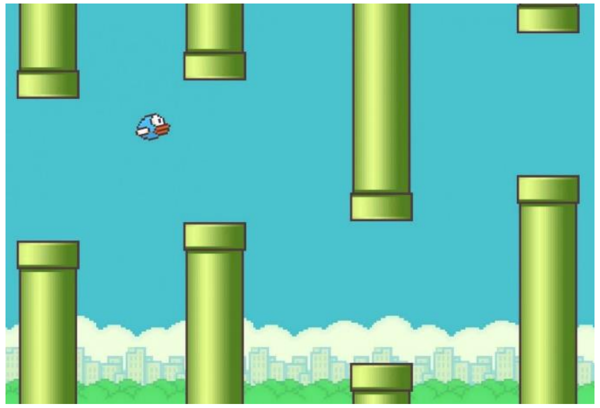
Leader of the AI team!

Team members and Platform:

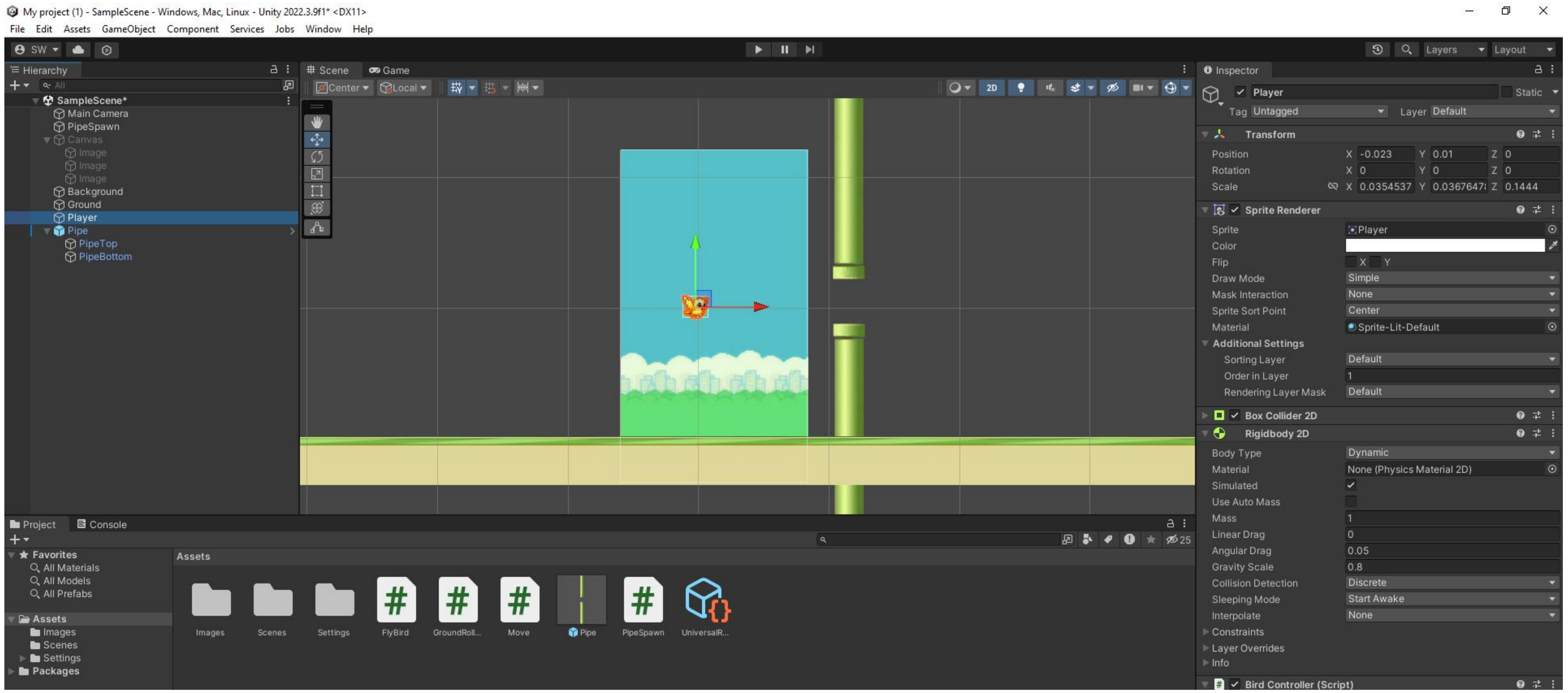
UI Design: Mid Journey (AI)

Coding: ChatGPT (AI)

Game Development Platform: Unity (Game Engine)



Flappy Bird



Player Static
Tag Untagged Layer Default

Transform

Position	X -0.023	Y 0.01	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 0.0354537	Y 0.0367647	Z 0.1444

Sprite Renderer

Sprite	Player
Color	
Flip	X Y
Draw Mode	Simple
Mask Interaction	None
Sprite Sort Point	Center
Material	Sprite-Lit-Default
Additional Settings	
Sorting Layer	Default
Order in Layer	1
Rendering Layer Mask	Default

Box Collider 2D

Rigidbody 2D

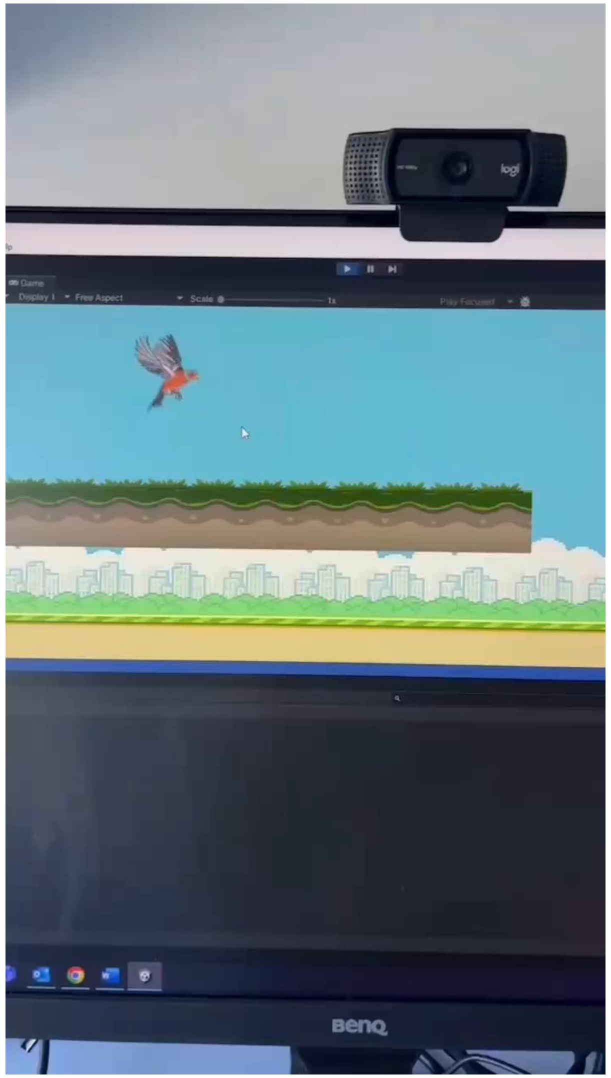
Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	0.8
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None
Constraints	
Layer Overrides	
Info	

Bird Controller (Script)

Experience

Goal: Develop the game without any coding by myself!

Result: Failed?



University of South-Eastern Norway
School of Business

Flappy Bird

Descriptions

We are going to make a game like Flappy Bird in Unity. Can you describe all the steps that will be required to go from an empty project to a working copy of Flappy bird? Functionality should include:

1. The player avatar is a bird.
2. The player avatar is constantly flying.
3. Gravity is always acting on the player avatar pulling it down towards the ground.
4. The player can click with the mouse button to flap their wings to get some upward velocity.
5. Obstacles in the form of pipes move from right to left.
6. Pipes are constantly spawned with a gap at different heights that the player must fly through and avoid hitting.
7. The ground should move from right to left to give the illusion of the bird is flying from left to right.
8. When the player avatar hits one of the pipes the game is over, the screen shows the image of “game over”.
9. When the player hits the ground, the game is over, the screen shows the image of “game over”.
10. The player can click on a button called “Start” to restart the game when it is over.

Flappy Bird

It takes some time to figure out…

1. You may receive different solution/coding when you repeat the whole process.
2. Write a script to move the pipes from the right to the left first, and then write another script to spawn the pipes!
3. Write a script to spawn the pipes, and make sure the pipes are constantly spawned with a gap at different heights.
4. The game may get stuck with the cloned ground, so you may have to fix some issues from external help or with your own ideas.
5. When you set up some functions through Unity, YouTube works better than ChatGPT, as you cannot find the correct location it offers to you through description.
6. If you are familiar with Unity and the logic of game design, ChatGPT can be your best assistant.

Have a Break!

Report

One representative from each group report for:

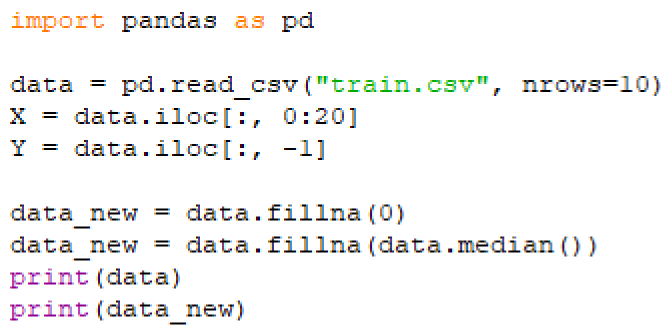
1. What is your experience
2. Show your game

Review Cases With ChatGPT

University of
South-Eastern Norway
School of Business

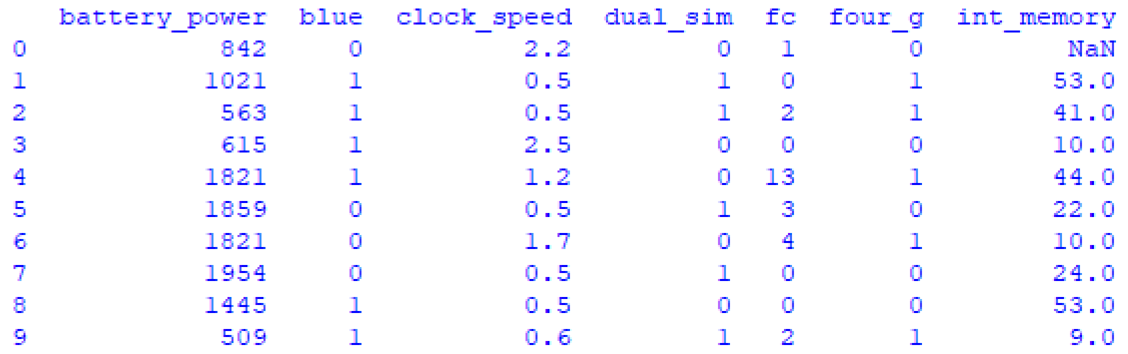
Imputation

Using median values

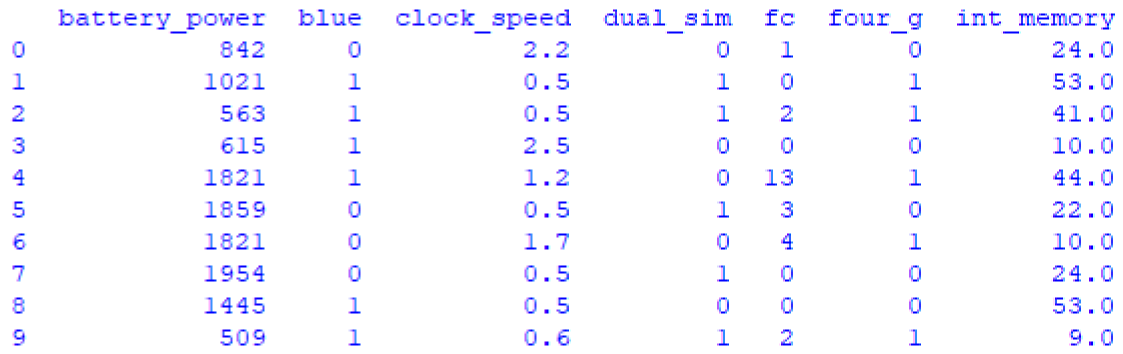


```
import pandas as pd

data = pd.read_csv("train.csv", nrows=10)
X = data.iloc[:, 0:20]
Y = data.iloc[:, -1]

data_new = data.fillna(0)
data_new = data.fillna(data.median())
print(data)
print(data_new)
```

Description: The name of the file is "train" in the form of CSV. I need you to print the original data and new data where all missing values in this file are replaced by the median value. Now send me the coding.



```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory
0            842     0          2.2         0   1       0         NaN
1           1021     1          0.5         1   0       1        53.0
2            563     1          0.5         1   2       1        41.0
3            615     1          2.5         0   0       0        10.0
4           1821     1          1.2         0  13       1        44.0
5           1859     0          0.5         1   3       0        22.0
6           1821     0          1.7         0   4       1        10.0
7           1954     0          0.5         1   0       0        24.0
8           1445     1          0.5         0   0       0        53.0
9            509     1          0.6         1   2       1         9.0
```

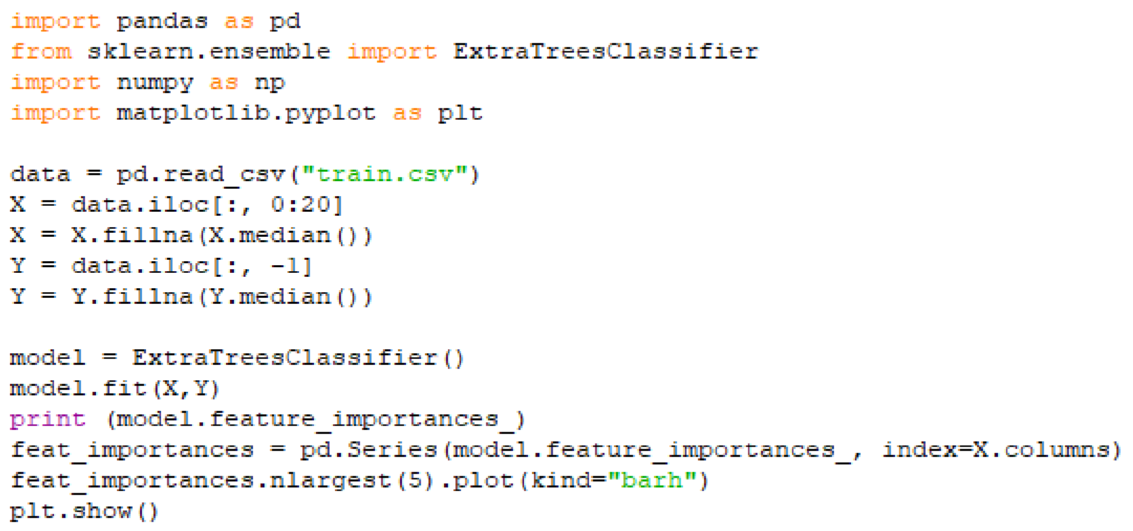


```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory
0            842     0          2.2         0   1       0        24.0
1           1021     1          0.5         1   0       1        53.0
2            563     1          0.5         1   2       1        41.0
3            615     1          2.5         0   0       0        10.0
4           1821     1          1.2         0  13       1        44.0
5           1859     0          0.5         1   3       0        22.0
6           1821     0          1.7         0   4       1        10.0
7           1954     0          0.5         1   0       0        24.0
8           1445     1          0.5         0   0       0        53.0
9            509     1          0.6         1   2       1         9.0
```

University of South-Eastern Norway
School of Business

Approach – Feature Importance

Task: Select the top 3 important features from the given data



```python
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("train.csv")
X = data.iloc[:, 0:20]
X = X.fillna(X.median())
Y = data.iloc[:, -1]
Y = Y.fillna(Y.median())

model = ExtraTreesClassifier()
model.fit(X,Y)
print (model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind="barh")
plt.show()
```

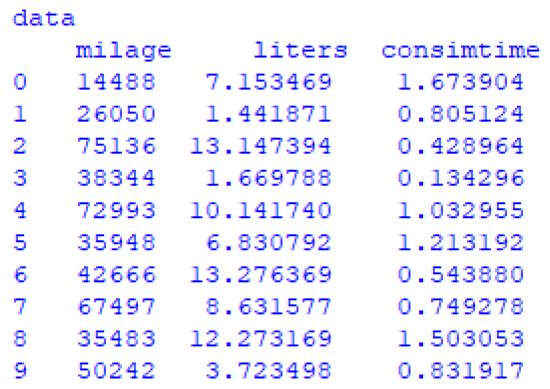
Description: The name of the file is "train" in the form of CSV. Select the first 20 columns of the DataFrame data as the feature matrix X. Select the last column of the DataFrame data as the target variable Y. Handle the missing value with the median value. I need you to select the top 5 important features from the given data, use ExtraTreesClassifer, and visualize the result with Matplotlib.

Scaling

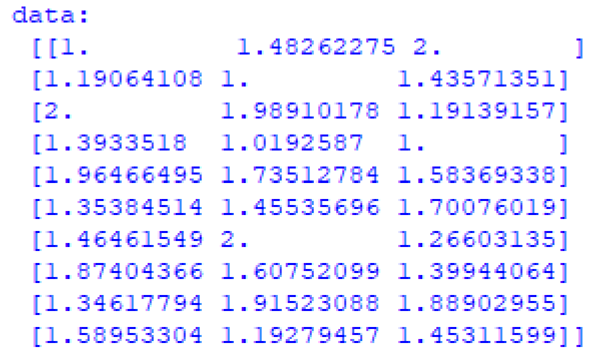
Description: The name of the file is "dating" in the form of txt. Select the first 3 columns of the DataFrame data as the feature matrix X. Select the last column of the DataFrame data as the target variable Y. Use Normalization to scaling all features, print the original data and the new data

Normalization

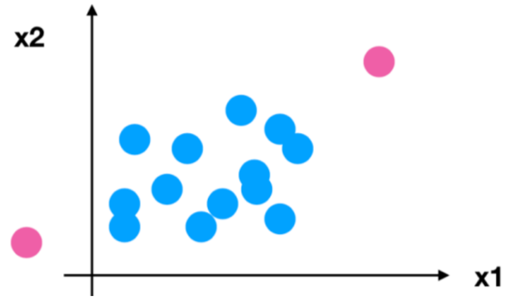
$$X' = \frac{x - min}{max - min} \qquad X'' = X' * (mx - mi) + mi$$



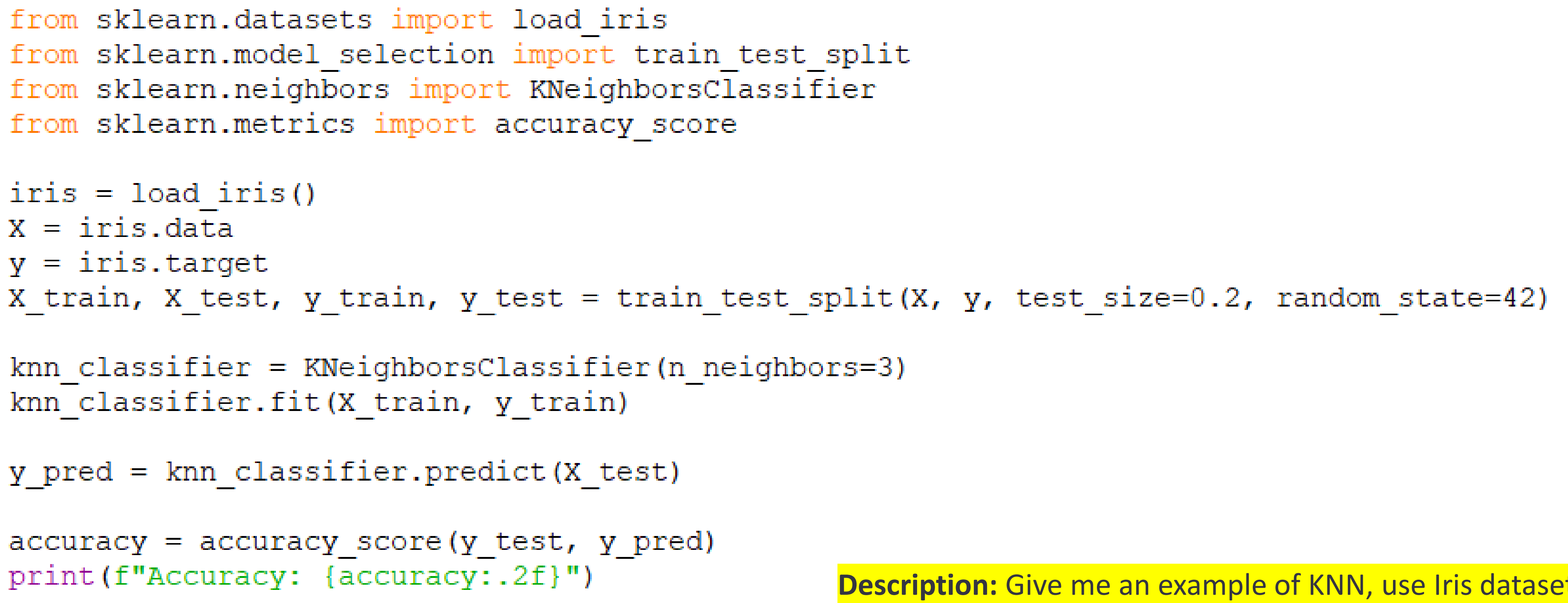
```
data
    milage      liters   consimtime
0   14488    7.153469    1.673904
1   26050    1.441871    0.805124
2   75136   13.147394    0.428964
3   38344    1.669788    0.134296
4   72993   10.141740    1.032955
5   35948    6.830792    1.213192
6   42666   13.276369    0.543880
7   67497    8.631577    0.749278
8   35483   12.273169    1.503053
9   50242    3.723498    0.831917
```



```
data:
 [[1.         1.48262275 2.        ]
 [1.19064108 1.         1.43571351]
 [2.         1.98910178 1.19139157]
 [1.3933518  1.0192587  1.        ]
 [1.96466495 1.73512784 1.58369338]
 [1.35384514 1.45535696 1.70076019]
 [1.46461549 2.         1.26603135]
 [1.87404366 1.60752099 1.39944064]
 [1.34617794 1.91523088 1.88902955]
 [1.58953304 1.19279457 1.45311599]]
```



Split data

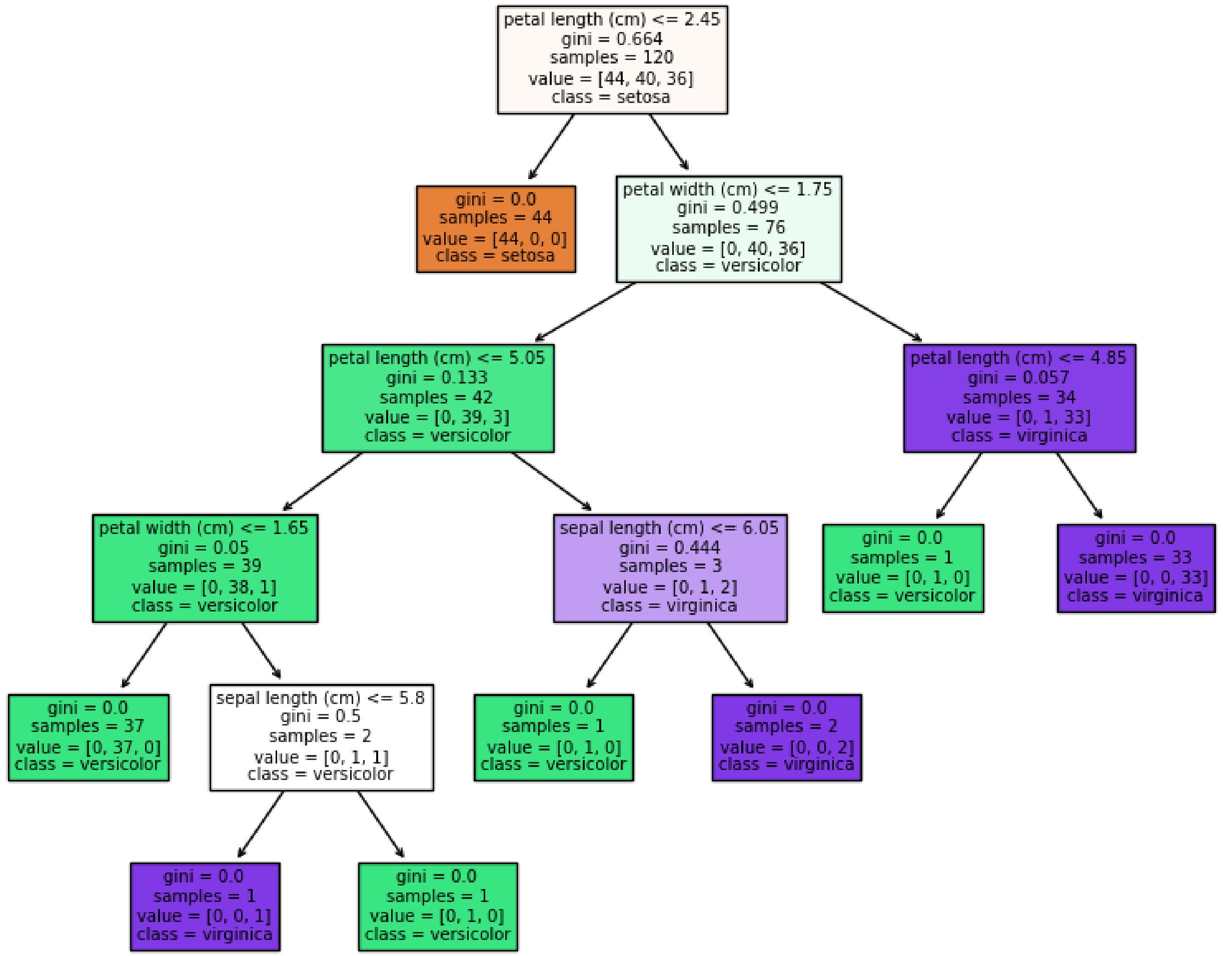


```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)

y_pred = knn_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Description: Give me an example of KNN, use Iris dataset.

Decision Trees

```python
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

# Create a Decision Tree classifier
dt = DecisionTreeClassifier()

# Train the classifier on the training data
dt.fit(X_train, y_train)

# Visualize the decision tree
plt.figure(figsize=(10, 8))
plot_tree(dt, feature_names=iris.feature_names, class_names=iris.target_names, fi
plt.show()
```


petal length (cm) <= 2.45 / gini = 0.664 / samples = 120 / value = [44, 40, 36] / class = setosa
gini = 0.0 / samples = 44 / value = [44, 0, 0] / class = setosa
petal width (cm) <= 1.75 / gini = 0.499 / samples = 76 / value = [0, 40, 36] / class = versicolor
petal length (cm) <= 5.05 / gini = 0.133 / samples = 42 / value = [0, 39, 3] / class = versicolor
petal length (cm) <= 4.85 / gini = 0.057 / samples = 34 / value = [0, 1, 33] / class = virginica
petal width (cm) <= 1.65 / gini = 0.05 / samples = 39 / value = [0, 38, 1] / class = versicolor
sepal length (cm) <= 6.05 / gini = 0.444 / samples = 3 / value = [0, 1, 2] / class = virginica
gini = 0.0 / samples = 1 / value = [0, 1, 0] / class = versicolor
gini = 0.0 / samples = 33 / value = [0, 0, 33] / class = virginica
gini = 0.0 / samples = 37 / value = [0, 37, 0] / class = versicolor
sepal length (cm) <= 5.8 / gini = 0.5 / samples = 2 / value = [0, 1, 1] / class = versicolor
gini = 0.0 / samples = 1 / value = [0, 1, 0] / class = versicolor
gini = 0.0 / samples = 2 / value = [0, 0, 2] / class = virginica
gini = 0.0 / samples = 1 / value = [0, 0, 1] / class = virginica
gini = 0.0 / samples = 1 / value = [0, 1, 0] / class = versicolor

Description: Give me an example of decision tree, use Iris dataset.

Case Study

Ever wonder what it's like to work at Facebook? Facebook and Kaggle are launching a machine learning engineering competition for 2016. Trail blaze your way to the top of the leaderboard to earn an opportunity at interviewing for one of the 10+ open roles as a software engineer, working on world class machine learning problems.

The goal of this competition is to predict which place a person would like to check in to. For the purposes of this competition, Facebook created an artificial world consisting of more than 100,000 places located in a 10 km by 10 km square. For a given set of coordinates, your task is to return a ranked list of the most likely places. Data was fabricated to resemble location signals coming from mobile devices, giving you a flavor of what it takes to work with real data complicated by inaccurate and noisy values. Inconsistent and erroneous location data can disrupt experience for services like Facebook Check In.

https://www.kaggle.com/competitions/facebook-v-predicting-check-ins

Data Explorer

857.5 MB

- sample_submission.csv.zip
- test.csv.zip
- train.csv.zip

Download from Canvas

Description: The name of the file is “FB Competition" in the form of CSV. Selected features in the column which are named "x", "y", "accuracy", and "time". Selected target value in the column which is named "place_id". Use samples where the value in accuracy is larger than 3. Standardize the features and predict Y using k-Nearest Neighbors.

Case Study 2

- Survive from the Titanic

  Steps?

  What features are important?

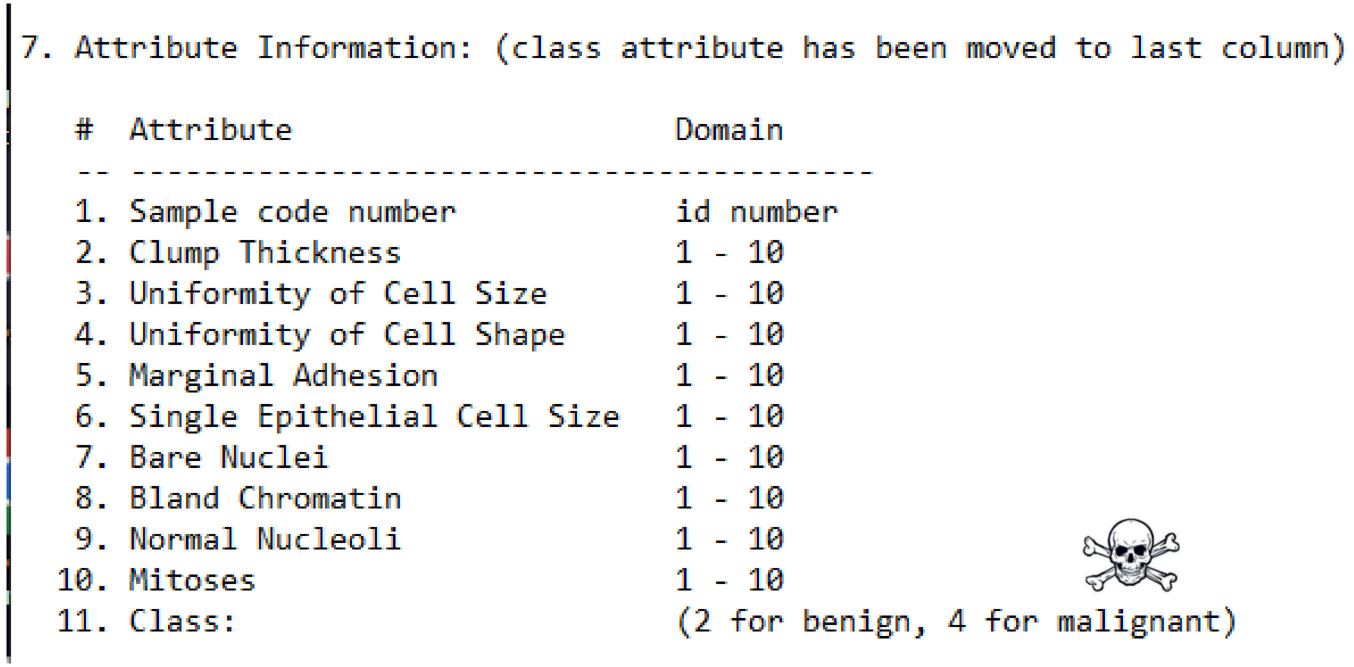
  How to build the model with decision tree?

  Use your model to predict if Rose would survive

  How bout Jack?

Example – Cancer Prediction

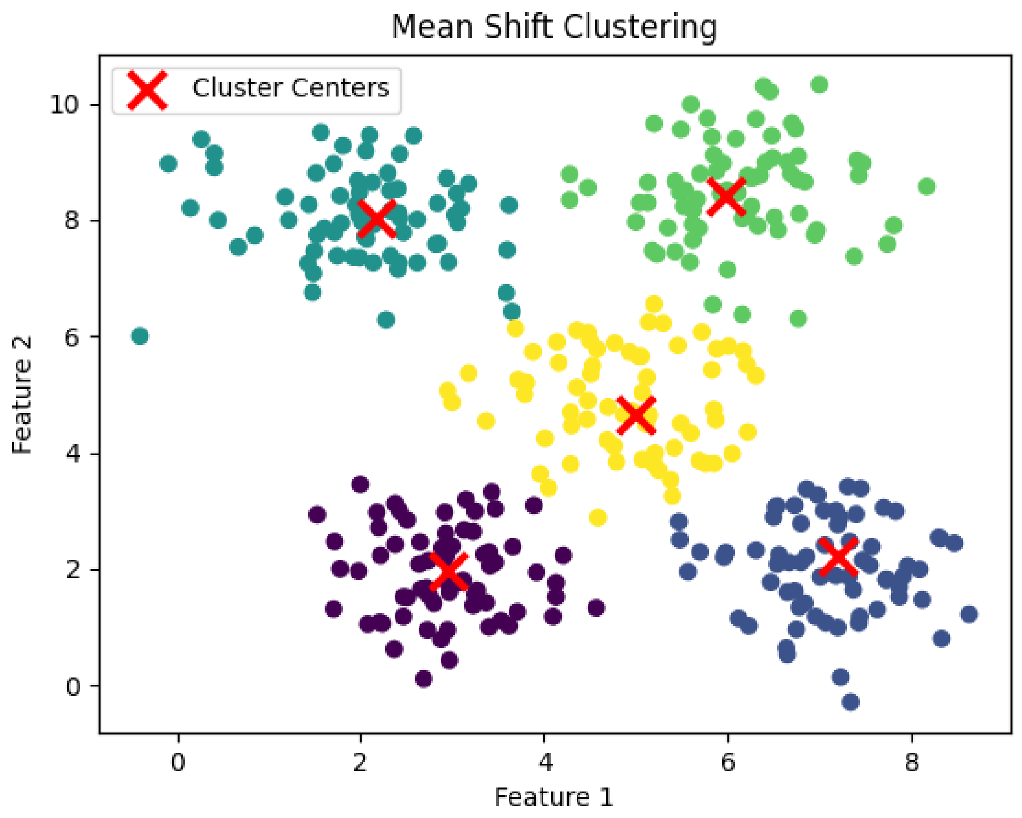
1. Load the dataset
2. Data processing (Define data into features and target)
3. Split data into train and test sets
4. Standardize the data using "StandardScaler"
5. Create and train a logistic regression model
6. Make predictions in the test set
7. Accuracy
8. Save the trained model



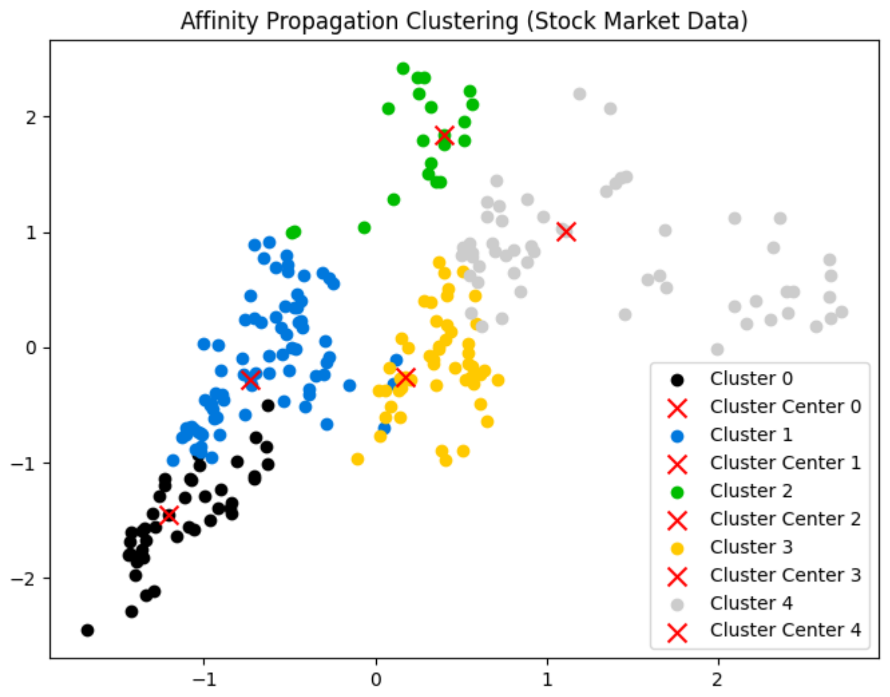
```
7. Attribute Information: (class attribute has been moved to last column)

   #  Attribute                     Domain
   -- -----------------------------------------
   1. Sample code number            id number
   2. Clump Thickness               1 - 10
   3. Uniformity of Cell Size       1 - 10
   4. Uniformity of Cell Shape      1 - 10
   5. Marginal Adhesion             1 - 10
   6. Single Epithelial Cell Size   1 - 10
   7. Bare Nuclei                   1 - 10
   8. Bland Chromatin               1 - 10
   9. Normal Nucleoli               1 - 10
  10. Mitoses                       1 - 10
  11. Class:                        (2 for benign, 4 for malignant)
```

The number of clusters (Mean Shift algorithm)

```python
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle

# Load data from input file
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Estimate the bandwidth of X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# Cluster data with MeanShift
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

# Extract the centers of clusters
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)

# Estimate the number of clusters
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print("\nNumber of clusters in input data =", num_clusters)

# Plot the points and cluster centers
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    # Plot points that belong to the current cluster
    plt.scatter(X[labels==i, 0], X[labels==i, 1], marker=marker, color='r')

    # Plot the cluster center
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
            markerfacecolor='black', markeredgecolor='black',
            markersize=15)

plt.title('Clusters')
plt.show()
```



Coding



Deprecation of the **"Open"** and **"Close"** attributes in the latest version of the **yfinance** library. Use DataFrame by calling the history symbol.

```
import datetime
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn import covariance, cluster
import yfinance as yf


# Input file containing company symbols
input_file = 'company_symbol_mapping.json'

# Load the company symbol map
with open(input_file, 'r') as f:
    company_symbols_map = json.loads(f.read())

symbols, names = np.array(list(company_symbols_map.items())).T

# Load the historical stock quotes
start_date = datetime.datetime(2019, 1, 1)
end_date = datetime.datetime(2019, 1, 31)
quotes = [yf.Ticker(symbol).history(start=start_date, end=end_date)
                for symbol in symbols]

# Extract opening and closing quotes
opening_quotes = np.array([quote.Open for quote in quotes]).astype(np.float)
closing_quotes = np.array([quote.Close for quote in quotes]).astype(np.float)

# Compute differences between opening and closing quotes
quotes_diff = closing_quotes - opening_quotes

# Normalize the data
X = quotes_diff.copy().T
X /= X.std(axis=0)

# Create a graph model
edge_model = covariance.GraphLassoCV()

# Train the model
with np.errstate(invalid='ignore'):
    edge_model.fit(X)

# Build clustering model using Affinity Propagation model
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()

# Print the results of clustering
print('\nClustering of stocks based on difference in opening and closing quotes:\n')
for i in range(num_labels + 1):
    print("Cluster", i+1, "==>", ', '.join(names[labels == i]))
```

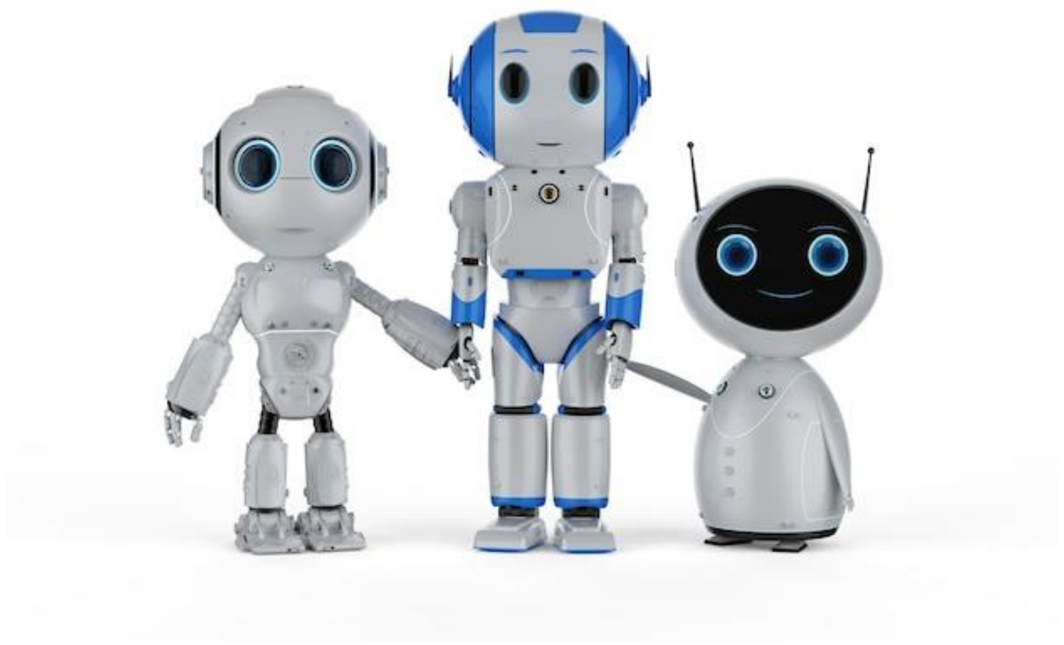
University of South-Eastern Norway
School of Business

Do not just copy-paste coding

Learn and understand it!

Have a Break!

Group Up!



Work on your assignment!

University of South-Eastern Norway
School of Business

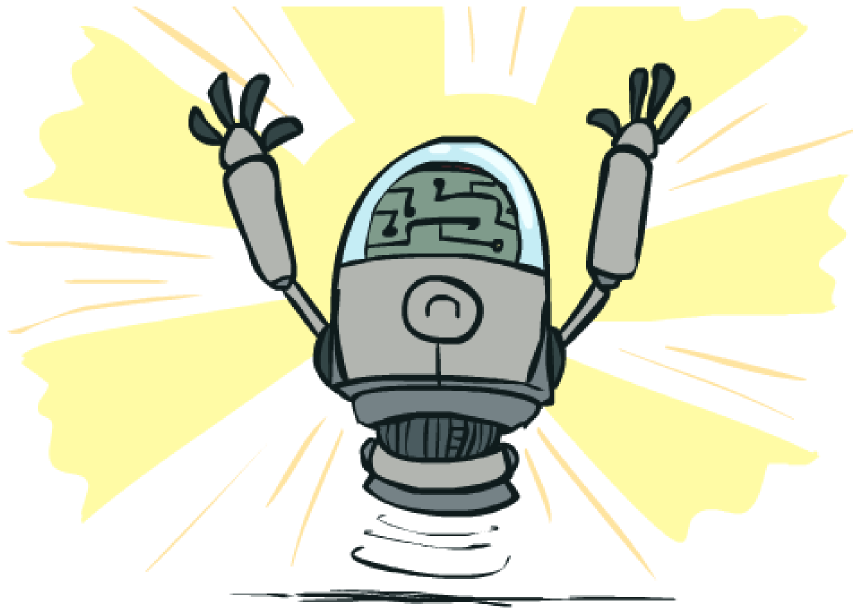
Guidance

https://www.youtube.com/watch?v=k59FJ3NzD7s

https://www.youtube.com/watch?v=jRAAaDll34Q

Room Change: B207

ASSIGNMENT PROCESS



THANKS