Unit 5: Biomedical Engineering

Outcome: Students will predict if a material is suitable for orthopedic implants using simple material properties.

Learning Objectives

Students will:

1. Understand the role of materials in medical implants.
2. Learn about common properties that make materials bio-compatible.
3. Use the Materials Project database to explore real materials.
4. Train a machine learning model to classify materials.
5. Predict the suitability of a new material for implants.

Materials Needed

- Google accounts + Colab access
- API key from Materials Project (teacher can provide)
- Provided dataset (based on curated Materials Project data)

Lesson Structure

Introduction (25 minutes)

1. You will need to do some research to determine where these bones are located and the common name by which they are known.

Scientific Name	Common Name	Scientific Name	Common Name
Cranium		Metacarpal	
Mandible		Phalanges	
Clavicle		Sacrum	
Sternum		Coccyx	
Spinal Column		Pelvis	
Scapula		Femur	
Humerus		Patella	
Radius		Fibula	
Ulna		Tibia	
Ribs		Tarsal	
Carpal		Metatarsal	

2. Research a material selected for a real world application of a joint based upon the following information. Define each property and the maxima and minima
    - Ductility
    - Load
    - Fatigue
    - Friction
    - Modulas of Elasticity

3. List three features (properties) that make a good implant material

4. List the chemical formula of three metallic implant materials

5. **What is the "Materials Project"?**

6. **Define the following properties from the Materials Project:**

```
"material_id",

"density",

"elements",

"Nelements",
```

```
"Volume",

"band_gap",

"Energy_above_hull",

"Is_magnetic",

"homogeneous_poisson"
```

7. What is machine learning in this context?

Activity in Google Colab (60 minutes)

Cell 1: Introduction (Text)

Create a Colab Notebook with this text header

Predicting Orthopedic Implant Materials with ML

We'll use real data from the Materials Project to train a machine learning model to identify suitable materials for implants.

Cell 2: Install and Import Libraries

```
!pip install pymatgen pandas scikit-learn -q
!pip install mp-api --upgrade
from mp_api.client import MPRester
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

Cell 3: Load Material Data from Materials Project and find Properties:

Choose two metal elements from your list above and the following features (properties)
material_id, band_gap (eV), density (g/cc), elasticity_modulus (GPa), poisson_ratio

Define these features here:

```
from mp_api.client import MPRester
import pandas as pd

API_KEY = "btxkK35J5mjY39BXe9DEHE7mBPniUsFP"

# Step 1: Get metallic materials with Co and Ti
with MPRester(API_KEY) as mpr:
    summaries = mpr.materials.summary.search(
        elements=["Co", "Ti"],
        band_gap=(0, 0.1),
        is_stable=True,
        fields=["material_id", "band_gap", "density"]
    )

material_ids = [entry.material_id for entry in summaries]

# Step 2: Get elasticity properties
with MPRester(API_KEY) as mpr:
    elastic_data = mpr.materials.elasticity.search(
        material_ids=material_ids,
        fields=["material_id", "bulk_modulus", "shear_modulus",
"homogeneous_poisson"]
    )

# Create lookup
elastic_lookup = {
    e.material_id: {
        "poisson_ratio": e.homogeneous_poisson,
        "elasticity_modulus (GPa)": e.bulk_modulus or e.shear_modulus
    }
    for e in elastic_data
}

# Merge results
```

```
final_data = []
for s in summaries:
    elastic = elastic_lookup.get(s.material_id, {})
    final_data.append({
        "material_id": s.material_id,
        "band_gap (eV)": s.band_gap,
        "density (g/cc)": s.density,
        "poisson_ratio": elastic.get("poisson_ratio"),
        "elasticity_modulus (GPa)": elastic.get("elasticity_modulus
(GPa)"),
        "implant_suitable": "Unknown"
    })

df = pd.DataFrame(final_data)
print(df.head())
```

Using Gemini and ChatGPTComplete the following program steps and paste in your program

---

Cell 4: Prepare the Data (make sure the data is numerical)

Cell 5: Train the ML Model (use a training set/data set of 80/20 and the Decision Tree Classifier)

Cell 6: Make Predictions on a New Material or Pull 5 real Ti-based materials, predict using your model.

Cell 7: Visualize the data (Graphs!!)

a. **Scatter Plot: Elasticity Modulus vs Density**
   Shows how these properties group materials into implant-suitable or not.

b. **Boxplot of Band Gap by Implant Suitability**
   Highlights differences in electrical properties that may affect biocompatibility

---

Cell 7: Interactive Challenges

- **Challenge A:** Try removing one of the four features and retrain. How does accuracy change?

- **Challenge B:** Add “elasticity_modulus / density” as a new feature—does it improve predictions?

- **Challenge C:** Use max_depth=3 in your Decision Tree—how does the model change? (Visualize optional)

Reflection & Wrap-Up

1. How do material properties influence implant suitability?
2. What surprised you about model predictions
3. Is that accuracy high enough for medical-grade decisions?

NGSS Standards

1. HS-PS2-6 – Forces and Interactions

> *"Communicate scientific and technical information about why the molecular-level structure is important in the functioning of designed materials."*

- **Connection**: Students study **bio-compatible materials** and learn how **atomic/molecular structure** influences their performance in implants.

---

2. HS-ETS1-3 – Engineering Design

> *"Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs..."*

- **Connection**: Students evaluate materials for implants using real-world design criteria (e.g., strength, toxicity, corrosion resistance) through **machine learning**.

---

3. HS-ETS1-4 – Engineering Simulations

> *"Use a computer simulation to model the impact of proposed solutions to a complex real-world problem."*

- **Connection**: Students use **machine learning models** to simulate predictions about **material suitability** — mirroring how engineers test materials computationally before physical trials.

---

4. Science and Engineering Practices (SEPs)

- **Analyzing and Interpreting Data**: Students examine material datasets and ML outputs.
- **Using Mathematics and Computational Thinking**: Students train and apply a ML model.
- **Obtaining, Evaluating, and Communicating Information**: Students interact with the Materials Project database and explain results.

---

5. Crosscutting Concepts

- **Structure and Function**: Central to understanding why certain materials work in implants.
- **Cause and Effect**: Students investigate how material properties influence bio-compatibility.
- **Systems and System Models**: Students model predictions of complex material behavior.

Summary Table

NGSS Code	Title	Relevance to Lab
HS-PS2-6	Structure of Materials	Understand why certain materials are safe for implants
HS-ETS1-3	Evaluate solutions	Weigh trade-offs in material design
HS-ETS1-4	Use simulations	Predict outcomes with ML
SEPs	Analyze data, use computation	Train and test models
Crosscutting Concepts	Structure-function, systems	Materials as systems with properties

---

Answer Key for Possible Code:

```
#  Predicting Orthopedic Implant Materials with ML


!pip install pymatgen pandas scikit-learn mp-api -q


from mp_api.client import MPRester

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score


API_KEY = "btxkK35J5mjY39BXe9DEHE7mBPniUsFP"
```

```
# Step 1: Query basic materials summary for Ti-Co system
with MPRester(API_KEY) as mpr:
    docs = mpr.materials.summary.search(
        chemsys="Ti",
        fields=[
            "material_id", "density", "elements", "nelements", "volume", "band_gap",
            "energy_above_hull", "is_magnetic", "homogeneous_poisson"
        ]
    )

# Convert to DataFrame
summary_data = pd.DataFrame([doc.dict() for doc in docs])

# Rename homogeneous_poisson to poisson_ratio for consistency
summary_data.rename(columns={"homogeneous_poisson": "poisson_ratio"},
inplace=True)

# Define criteria
criteria = {
    "band_gap": (0.5, 2.0),
    "density": (2.0, 10.0),
    "poisson_ratio": (0.2, 0.4),
    "energy_above_hull": (0, 0.1),
    "num_sites": (1, 60),
```

```
    "num_magnetic_sites": (0, 20),
}


# Step 2: Determine implant suitability
def is_implant_suitable(row):
    try:
        if not (criteria["band_gap"][0] <= row["band_gap"] <= criteria["band_gap"][1]):
            return False
        if not (criteria["density"][0] <= row["density"] <= criteria["density"][1]):
            return False
        if not (criteria["poisson_ratio"][0] <= row["poisson_ratio"] <=
criteria["poisson_ratio"][1]):
            return False
        if not (criteria["energy_above_hull"][0] <= row["energy_above_hull"] <=
criteria["energy_above_hull"][1]):
            return False
        if not (criteria["num_sites"][0] <= row["nelements"] <= criteria["num_sites"][1]):
            return False
        if row["is_magnetic"] and criteria["num_magnetic_sites"][0] == 0:
            return False
    except:
        return False
    return True
```

```
summary_data["implant_suitable"] = summary_data.apply(is_implant_suitable, axis=1)

# Step 3: Select features
features = ["density", "poisson_ratio", "band_gap", "energy_above_hull", "is_magnetic"]
df_clean = summary_data.dropna(subset=features)
X = df_clean[features]
y = df_clean["implant_suitable"]

# Step 4: Train/test split and model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 5: Evaluate
accuracy = accuracy_score(y_test, model.predict(X_test))
print(f"✅ Model trained. Accuracy: {accuracy:.2f}")

# Preview data
summary_data.head()



from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
import matplotlib.pyplot as plt


# Step 6: Confusion Matrix
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot()
plt.show()
```