To train models using the three different algorithms—Linear Regression, K-Means Clustering, and Decision Trees—you can follow the steps below. I'll provide Python code snippets for each algorithm.

**1. Linear Regression for Economic Trend Prediction**

This example uses Linear Regression to predict the median income based on other economic features.

python

Copy code

import pymongo

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Connect to MongoDB and load data

client = pymongo.MongoClient("mongodb://localhost:27017/")

db = client["your\_database\_name"] # Replace with your database name

collection = db["Calgary\_Census\_2021"]

raw\_data = list(collection.find())

data = pd.json\_normalize(raw\_data)

# Select features and target

X = data[['Economics.Labour\_Force\_Status.Employed',

'Economics.Labour\_Force\_Status.Unemployed']] # Example features

y = data['Economics.Median\_Total\_Income\_Individuals']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Linear Regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions and evaluate

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Linear Regression MSE: {mse}")

**2. K-Means Clustering for Grouping Demographics**

This example uses K-Means Clustering to group similar demographic data points.

python

Copy code

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Select demographic features

X = data[['Demographics.Age\_Characteristics.Total\_Population\_Age\_Groups.0\_to\_14\_years',

'Demographics.Age\_Characteristics.Total\_Population\_Age\_Groups.15\_to\_64\_years',

'Demographics.Age\_Characteristics.Total\_Population\_Age\_Groups.65\_years\_and\_over']]

# Normalize the data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Train K-Means Clustering model

kmeans = KMeans(n\_clusters=3, random\_state=42)

clusters = kmeans.fit\_predict(X\_scaled)

# Add cluster labels to DataFrame

data['Cluster'] = clusters

print(data[['Cluster', 'Demographics.Age\_Characteristics.Total\_Population\_Age\_Groups.0\_to\_14\_years']].head())

**3. Decision Trees for Classification Tasks**

This example uses Decision Trees to classify data based on marital status.

python

Copy code

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Select features and target

X = data[['Demographics.Age\_Characteristics.Average\_Age',

'Demographics.Age\_Characteristics.Median\_Age']]

y = data['Demographics.Marital\_Status.Population\_Aged\_15\_and\_Over.Married\_or\_Common\_Law']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Decision Tree model

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions and evaluate

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Decision Tree Accuracy: {accuracy}")

**Summary**

1. **Linear Regression** is used for predicting continuous outcomes, such as predicting the median income based on employment data.
2. **K-Means Clustering** groups data into clusters based on similarity, such as grouping age demographics.
3. **Decision Trees** classify data based on rules, such as predicting whether a person is married or in a common-law relationship based on age characteristics.

These models give you different insights into the data depending on the kind of analysis you need to perform. Be sure to adjust feature selection and preprocessing based on your specific data structure and requirements.