Here is a structured approach to perform predictive analysis using Scikit-learn on the provided data:

**1. Load and Split Data:**

**Load Data:**

* Use Pandas to load the structured data from MongoDB into a DataFrame.

**Split Data:**

* Split the data into training and testing sets using train\_test\_split from Scikit-learn.

python

Copy code

import pandas as pd

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

import pymongo

# Connect to MongoDB

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

db = client["your\_database\_name"]

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

# Load data into DataFrame

data = pd.DataFrame(list(collection.find()))

# For demonstration, let's assume we're focusing on predicting 'Median\_Total\_Income\_Individuals'

X = data.drop(columns=['Median\_Total\_Income\_Individuals']) # Features

y = data['Median\_Total\_Income\_Individuals'] # Target

# Split the data

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

**2. Model Selection:**

**Choose Algorithms:**

* Select an algorithm based on the type of data and analysis. For example:
  + **Linear Regression** for economic trend prediction.
  + **K-Means Clustering** for grouping demographics.
  + **Decision Trees** for classification tasks.

**Train Models:**

* Train the model using the training data.

python

Copy code

from sklearn.linear\_model import LinearRegression

# Initialize and train the model

model = LinearRegression()

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

**3. Evaluate Models:**

**Assess Performance:**

* Evaluate the model's performance using appropriate metrics.

python

Copy code

from sklearn.metrics import mean\_squared\_error, r2\_score

# Predict on test data

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

# Evaluate the model

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

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R^2 Score: {r2}")

**Hyperparameter Tuning:**

* Use GridSearchCV or RandomizedSearchCV to fine-tune the model parameters.

python

Copy code

from sklearn.model\_selection import GridSearchCV

# Example of hyperparameter tuning for a Decision Tree

param\_grid = {'max\_depth': [3, 5, 10], 'min\_samples\_split': [2, 5, 10]}

grid\_search = GridSearchCV(DecisionTreeRegressor(), param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

**4. Make Predictions:**

**Apply Models:**

* Use the trained model to predict on new or unseen data.

**Store Results:**

* Save the prediction results back to MongoDB or export them.

python

Copy code

# Assuming new data to predict

new\_data = X\_test.iloc[:5] # Example of new data

# Make predictions

predictions = model.predict(new\_data)

# Update MongoDB with predictions

for i, pred in enumerate(predictions):

collection.update\_one({"\_id": new\_data.index[i]}, {"$set": {"Predicted\_Income": pred}})

This approach will guide you through loading, training, evaluating, and saving predictive models using Scikit-learn with your provided dataset. Adjust the model and features as per your specific data analysis needs.