1. Make a new folder `ModelSerialization.py` in the same directory as your app store.

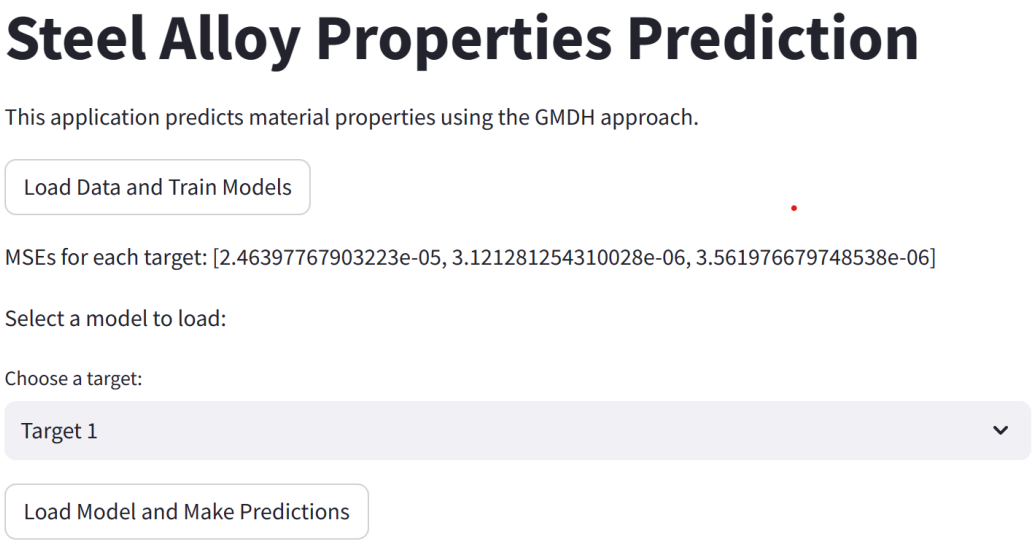
```
from joblib import dump, load

# Function to save a model
def save_model(model, filename):
    dump(model, filename)
    print(f"Model saved successfully as {filename}.")

# Function to load a model
def load_model(filename):
    model = load(filename)
    print(f"Model loaded successfully from {filename}.")
    return model
```

2. Then, in your Streamlit app, ensure you import it correctly:

from `ModelSerialization` import `save_model`




To read or load a model saved as a `.joblib` file, such as `gmdh_model_target_2.joblib`, you can use the `load` function from the `joblib` library. This function is designed to deserialize and load objects saved into `.joblib` files back into your Python environment.

Here's how you can do it step-by-step:

## Ensure you have `joblib` installed

If not already installed, you can install `joblib` using pip:

pip install joblib

Streamlit_code

Streamlit_results

1. **Importing Libraries:**

```
import streamlit as st
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from joblib import dump, load
from gmdhpy import gmdh
from ModelSerialization import save_model, load_model
import os
import matplotlib.pyplot as plt
```

2**. Function to Load Data:**

load_data(filepath): This function uses pandas to load data from a CSV file located at the specified filepath. The data is then returned.

The @st.cache decorator is used to cache the data, meaning that once the data is loaded, it won't be reloaded on subsequent runs unless the input changes, which enhances performance.

3. **Function to Train Models and Save Them:**

train_and_save_models(data, results_path): This function takes the loaded data and a path to save results. It processes the features and outputs, scales the features using StandardScaler, and splits the data into training and test sets. It then iterates over each target variable, trains a GMDH (Group Method of Data Handling) model, evaluates it using mean squared error (MSE), and saves each model to disk. It returns a list of MSEs for each target and the number of targets.

4. **Function to Plot MSE Values:**

plot_mses(mses): This function takes a list of MSEs and plots them using matplotlib. It shows how the MSE changes for each target variable, helping in visual assessment of model performance.

**5. Streamlit Interface:**

main(): The main function of the Streamlit application. It sets up the web interface, allowing users to interact with the application:

Displays the loaded data and MSE results.

Provides a dropdown to select a model for a specific target and a button to load and display details about the model.

All paths for data and results are hard-coded but could be made dynamic based on user input or configuration files.

**Usage in Streamlit:**

Load and view data.

Train models on this data.

View the MSE of these models.

Select and load trained models to view their parameters or use them for further predictions.

This setup is for demonstrating machine learning workflows, from data loading to model training and evaluation, in a user-friendly web application. It's particularly useful for showcasing the capabilities of GMDH models in predicting material properties based on alloy compositions.

```
pip install joblib
```