import pandas as pd

from sklearn.ensemble import RandomForestRegressor

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

from sklearn.metrics import mean\_absolute\_error

from sklearn.preprocessing import StandardScaler

# Load data

def load\_data(path: str = "/path/to/csv/"):

"""

This function takes a path string to a CSV file and loads it into

a Pandas DataFrame.

:param path (optional): str, relative path of the CSV file

:return df: pd.DataFrame

"""

df = pd.read\_csv(f"{path}")

df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')

return df

# Create target variable and predictor variables

def create\_target\_and\_predictors(

data: pd.DataFrame = None,

target: str = "estimated\_stock\_pct"

):

"""

This function takes in a Pandas DataFrame and splits the columns

into a target column and a set of predictor variables, i.e. X & y.

These two splits of the data will be used to train a supervised

machine learning model.

:param data: pd.DataFrame, dataframe containing data for the

model

:param target: str (optional), target variable that you want to predict

:return X: pd.DataFrame

y: pd.Series

"""

# Check to see if the target variable is present in the data

if target not in data.columns:

raise Exception(f"Target: {target} is not present in the data")

X = data.drop(columns=[target])

y = data[target]

return X, y

# Train algorithm

def train\_algorithm\_with\_cross\_validation(

X: pd.DataFrame = None,

y: pd.Series = None

):

"""

This function takes the predictor and target variables and

trains a Random Forest Regressor model across K folds. Using

cross-validation, performance metrics will be output for each

fold during training.

:param X: pd.DataFrame, predictor variables

:param y: pd.Series, target variable

:return

"""

# Create a list that will store the accuracies of each fold

accuracy = []

# Enter a loop to run K folds of cross-validation

for fold in range(0, K):

# Instantiate algorithm and scaler

model = RandomForestRegressor()

scaler = StandardScaler()

# Create training and test samples

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

# Scale X data, we scale the data because it helps the algorithm to converge

# and helps the algorithm to not be greedy with large values

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train model

trained\_model = model.fit(X\_train, y\_train)

# Generate predictions on test sample

y\_pred = trained\_model.predict(X\_test)

# Compute accuracy, using mean absolute error

mae = mean\_absolute\_error(y\_true=y\_test, y\_pred=y\_pred)

accuracy.append(mae)

print(f"Fold {fold + 1}: MAE = {mae:.3f}")

# Finish by computing the average MAE across all folds

print(f"Average MAE: {(sum(accuracy) / len(ac