```
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import VotingRegressor
from sklearn.svm import SVR
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
df = pd.read_excel("finalDataset.xlsx")
df1 = pd.read_excel("finalDataset.xlsx")
y = df['targetPrice']
del df['targetPrice']
# print(df)
x = df[df.columns[3:15]]
# print(x)

scaler = MinMaxScaler()
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
train_size=0.7, random_state=22222)
volume = x_test['Volume']
test_data = x_test

scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
y_train = y_train.to_frame()
y_test = y_test.to_frame()
scaler1 = MinMaxScaler()
scaler1.fit(y_train)
y_train = scaler1.transform(y_train)
y_test = scaler1.transform(y_test)

test_data.to_excel("testData.xlsx")
test_data = pd.read_excel("testData.xlsx")
# print(test_data)
reg = RandomForestRegressor(n_estimators=60, random_state=999)
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
preds = {'preds': y_pred}
target = {'target': y_test}
data = pd.DataFrame(preds)
frames = [test_data, data]
df = pd.concat(frames, axis=1)
print(df)
income = 0
y_pred = df['preds']
y_pred = y_pred.to_frame()
y_test = scaler1.inverse_transform(y_test)

y_pred = scaler1.inverse_transform(y_pred)

for i, rows in df.iterrows():
    if y_pred[i][0] > rows['Close']:
```

```
        income += ((1000) * y_test[i][0] / rows['Close']) - 1000
print(income)
```