```
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")
y = df['targetPrice']
del df['targetPrice']
x = df[df.columns[3:13]]
# print(df)

list = [10, 20, 40, 60, 100]


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)

print("Bagging Regressor:")
for num in list:
    reg = BaggingRegressor(base_estimator=LinearRegression(), n_estimators=num,
random_state=999)
    reg.fit(x_train, y_train)
    train_acc = reg.score(x_train, y_train)
    test_acc = reg.score(x_test, y_test)
    print(num)
    mse = mean_squared_error(y_test, reg.predict(x_test))
    print(mse)
    print(train_acc)
    print(test_acc)
    print(" ")

print("Random Forest Regressor:")
for num in list:
    reg = RandomForestRegressor(n_estimators=num, random_state=999)
    reg.fit(x_train, y_train)
    train_acc = reg.score(x_train, y_train)
    test_acc = reg.score(x_test, y_test)
```

```
    print(num)
    mse = mean_squared_error(y_test, reg.predict(x_test))
    print(mse)
    print(train_acc)
    print(test_acc)
    print(" ")

print("Ada Boost Regressor:")
for num in list:
    reg = AdaBoostRegressor(base_estimator=LinearRegression(), n_estimators=num,
random_state=999)
    reg.fit(x_train, y_train)
    train_acc = reg.score(x_train, y_train)
    test_acc = reg.score(x_test, y_test)
    print(num)
    mse = mean_squared_error(y_test, reg.predict(x_test))
    print(mse)
    print(train_acc)
    print(test_acc)
    print(" ")

print("Gradient Boost Regressor:")
for num in list:
    reg = GradientBoostingRegressor(n_estimators=num, random_state=999)
    reg.fit(x_train, y_train)
    train_acc = reg.score(x_train, y_train)
    test_acc = reg.score(x_test, y_test)
    print(num)
    mse = mean_squared_error(y_test, reg.predict(x_test))
    print(mse)
    print(train_acc)
    print(test_acc)
    print(" ")

reg2 = LinearRegression()
reg3 = DecisionTreeRegressor()
print("Voting Regressor:")
reg = VotingRegressor([('lr', reg2), ('dt', reg3)])
reg.fit(x_train, y_train)
train_acc = reg.score(x_train, y_train)
test_acc = reg.score(x_test, y_test)
mse = mean_squared_error(y_test, reg.predict(x_test))
print(mse)
print(train_acc)
print(test_acc)

k = df['Date']
# print(k)
scaler.fit(x)
x = scaler.transform(x)
y = y.to_frame()
scaler1 = MinMaxScaler()
scaler1.fit(y)
y = scaler1.transform(y)
reg = RandomForestRegressor(n_estimators=60, random_state=999)
reg.fit(x, y)
y_pred = reg.predict(x)

y_p = []
```

```
for i in y_pred:
    y_p.append(i)
# # print(y_p)
data = {'pred': y_pred}
predFrame = pd.DataFrame(data)
# print(predFrame)
frames = [df, predFrame]
df = pd.concat(frames, axis=1)
# print(df)
frames = [test_data, predFrame]
test_data = pd.concat(frames, axis=1)
print(test_data)
y_p = df['pred']
y_p = y_p.to_frame()
y_p = scaler1.inverse_transform(y_p)
y = scaler1.inverse_transform(y)
# # axes2 = axes1.twinx()
# # axes1.plot(k, y_pred, color='g')
# # # axes2.plot(k, y, color='b')
# # axes1.set_xlabel('K')
# # axes1.set_ylabel("Predictions", color='g')
# # axes2.set_ylabel("Actual", color='b')
plt.plot(k, y_p)
plt.plot(k, y)
plt.xticks([k[0], k[399], k[799], k[1199]])
plt.legend(['Predicted', 'Actual'])
plt.show()

start = False
newK = []
ypnew = []
ynew = []
count = 0
for i in k:
    if i == '2021-11-01':
        start = True
    if start:
        newK.append(k[count])
        ypnew.append(y_p[count])
        ynew.append(y[count])
    count += 1

plt.show()
plt.plot(newK, ypnew)
plt.plot(newK, ynew)
plt.xticks([newK[0], newK[30], '2022-02-01'])
plt.legend(['Predicted', 'Actual'])
plt.show()

# # income = 0
# # for i, rows in test_data.iterrows():
# #     vol = rows['Volume']
# #     if rows['Profit'] == 1:
# #         income += rows[]
```