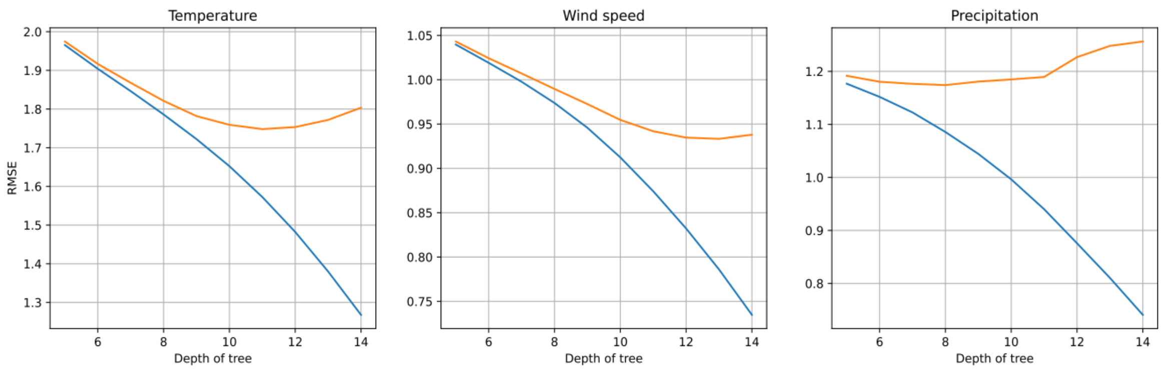
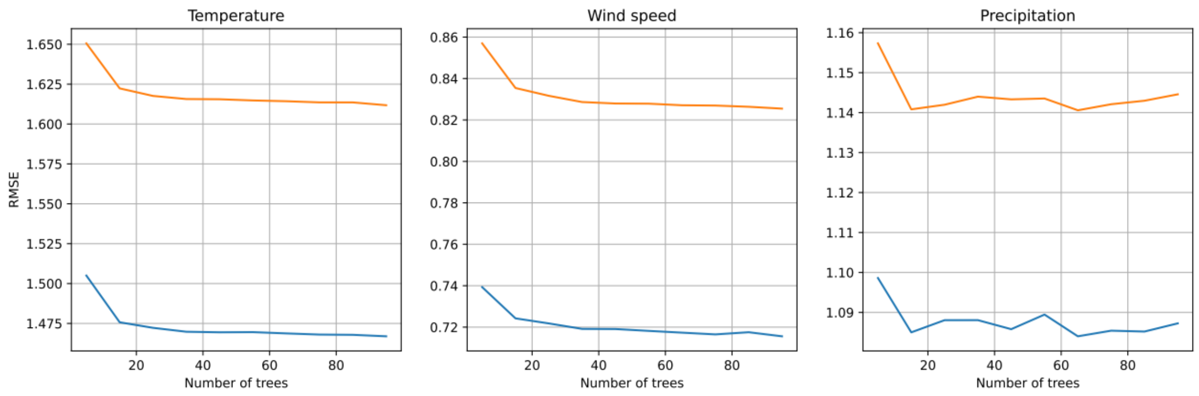
```python
#!/usr/bin/python3
from myTools import *
from sklearn import tree
from sklearn.model_selection import train_test_split, KFold, cross_validate
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import pickle

# read training files
input = pd.read_csv("InputNotScaled.csv", header=None)
target = pd.read_csv("TargetNotScaled.csv", header=None)

# setup for plotting
models = "tempTree windTree precipTree tempForest windForest precipForest tempNN windNN precipNN tempBR windBR precipBR".split()
rmses = []

# extract distinct test set, not to be touched
xTrain, xTest, yTrain, yTest = train_test_split(input, target, test_size=0.1, random_state = 0, shuffle=True)

# normalize input and target data
inputScaler = preprocessing.MinMaxScaler()
xTrain[xTrain.columns] = inputScaler.fit_transform(xTrain)
xTest[xTest.columns] = inputScaler.transform(xTest)
targetNNScaler = preprocessing.MinMaxScaler()
yTrainNN = yTrain.copy()[[0, 2, 3]]
yTrainNN[yTrainNN.columns] = targetNNScaler.fit_transform(yTrainNN)
# save scalers
# pickle.dump(inputScaler, open("inputScaler", "wb"))
# pickle.dump(targetNNScaler, open("targetNNScaler", "wb"))

# call these functions to find the best hyperparameters for each model in an experimental setup
trees(xTrain, yTrain)
randomForest(xTrain, yTrain)
neuralNet(xTrain, yTrainNN)
bayRidge(xTrain, yTrain)


# create best models - uncomment to train the models
# # trees
# tempTree = tree.DecisionTreeRegressor(max_depth=11)
# tempTree.fit(xTrain, yTrain[[0]])
# rmses.append(mean_squared_error(tempTree.predict(xTest), yTest[[0]])**0.5)
# windTree = tree.DecisionTreeRegressor(max_depth=13)
# windTree.fit(xTrain, yTrain[[2]])
# rmses.append(mean_squared_error(windTree.predict(xTest), yTest[[2]])**0.5)
# precipTree = tree.DecisionTreeRegressor(max_depth=8)
# precipTree.fit(xTrain, yTrain[[3]])
# rmses.append(mean_squared_error(precipTree.predict(xTest), yTest[[3]])**0.5)
# print("done with trees")

# # random forests
# tempForest = RandomForestRegressor(95, max_depth=11)
# tempForest.fit(xTrain, yTrain[[0]])
# rmses.append(mean_squared_error(tempForest.predict(xTest), yTest[[0]])**0.5)
# windForest = RandomForestRegressor(95, max_depth=13)
# windForest.fit(xTrain, yTrain[[2]])
# rmses.append(mean_squared_error(windForest.predict(xTest), yTest[[2]])**0.5)
# precipForest = RandomForestRegressor(65, max_depth=7)
# precipForest.fit(xTrain, yTrain[[3]])
# rmses.append(mean_squared_error(precipForest.predict(xTest), yTest[[3]])**0.5)
# print("done with forests")

# # neural net
# nn = MLPRegressor([12, 9, 6], max_iter=18)
# nn.fit(xTrain, yTrainNN[[0, 2, 3]])
# pred = targetNNScaler.inverse_transform(nn.predict(xTest))
# rmses.append(mean_squared_error(pred[:, 0], yTest[[0]]) ** 0.5)
# rmses.append(mean_squared_error(pred[:, 1], yTest[[2]]) ** 0.5)
# rmses.append(mean_squared_error(pred[:, 2], yTest[[3]]) ** 0.5)

# # bayesian ridge
# brTemp = BayesianRidge()
# brTemp.fit(xTrain, yTrain[[0]])
# brWind = BayesianRidge()
# brWind.fit(xTrain, yTrain[[2]])
# brPrecip = BayesianRidge()
# brPrecip.fit(xTrain, yTrain[[3]])


# # saving models
# pickle.dump(tempTree, open("tempTree", "wb"))
# pickle.dump(windTree, open("windTree", "wb"))
# pickle.dump(precipTree, open("precipTree", "wb"))
# pickle.dump(tempForest, open("tempForest", "wb"))
# pickle.dump(windForest, open("windForest", "wb"))
# pickle.dump(precipForest, open("precipForest", "wb"))
# pickle.dump(nn, open("nn", "wb"))
# pickle.dump(brTemp, open("brTemp", "wb"))
# pickle.dump(brWind, open("brWind", "wb"))
# pickle.dump(brPrecip, open("brPrecip", "wb"))

# load models that were previously saved
tempTree = pickle.load(open("tempTree", "rb"))
windTree = pickle.load(open("windTree", "rb"))
precipTree = pickle.load(open("precipTree", "rb"))
tempForest = pickle.load(open("tempForest", "rb"))
windForest = pickle.load(open("windForest", "rb"))
precipForest = pickle.load(open("precipForest", "rb"))
nn = pickle.load(open("nn", "rb"))
brTemp = pickle.load(open("brTemp", "rb"))
brWind = pickle.load(open("brWind", "rb"))
brPrecip = pickle.load(open("brPrecip", "rb"))
```

```
# make predictions on test data and calculate rmses
pred = tempTree.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[0]])**0.5)
pred = windTree.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[2]])**0.5)
pred = precipTree.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[3]])**0.5)
pred = tempForest.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[0]])**0.5)
pred = windForest.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[2]])**0.5)
pred = precipForest.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[3]])**0.5)
pred = nn.predict(xTest)
pred = targetNNScaler.inverse_transform(nn.predict(xTest))
rmses.append(mean_squared_error(pred[:, 0], yTest[[0]])**0.5)
rmses.append(mean_squared_error(pred[:, 1], yTest[[2]])**0.5)
rmses.append(mean_squared_error(pred[:, 2], yTest[[3]])**0.5)
pred = brTemp.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[0]])**0.5)
pred = brWind.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[2]])**0.5)
pred = brPrecip.predict(xTest)
rmses.append(mean_squared_error(pred, yTest[[3]])**0.5)

#plot rmses
fig = plt.figure()
plt.bar(models, rmses, color=["orange", "orange", "orange", "deepskyblue", "deepskyblue", "deepskyblue", "grey", "grey", "grey", "black", "black", "black"])
plt.ylabel("RMSE")
plt.xlabel("Model")
plt.title("RMSEs model comparison")
plt.grid()
plt.show()
```

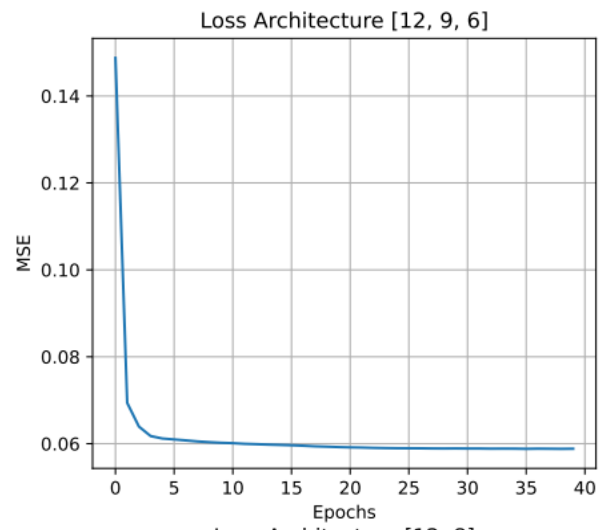
Output trees(xTrain, yTrain)

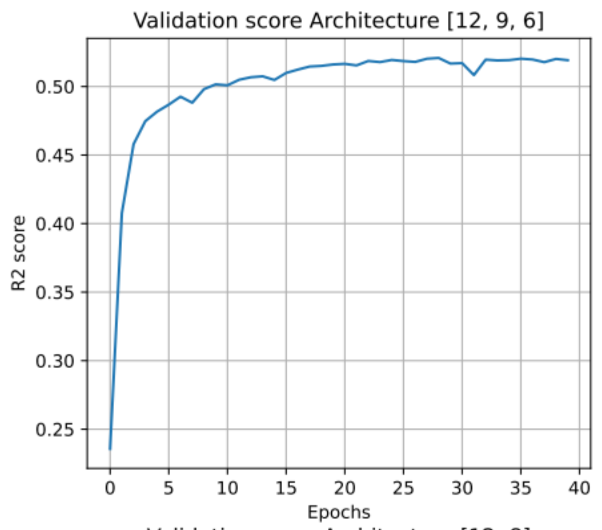


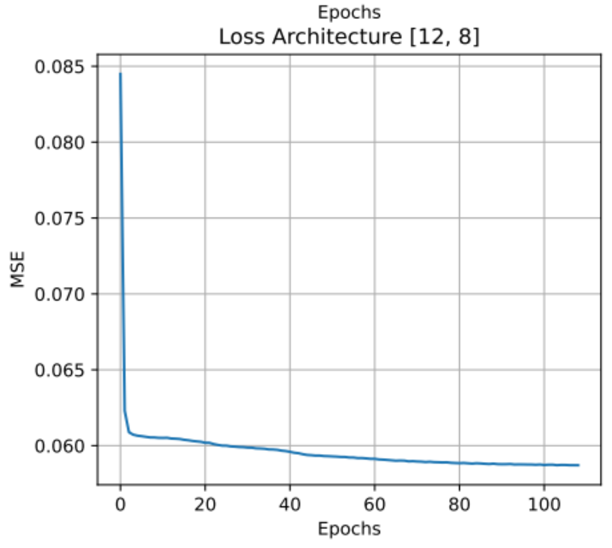
Output randomForest(xTrain,

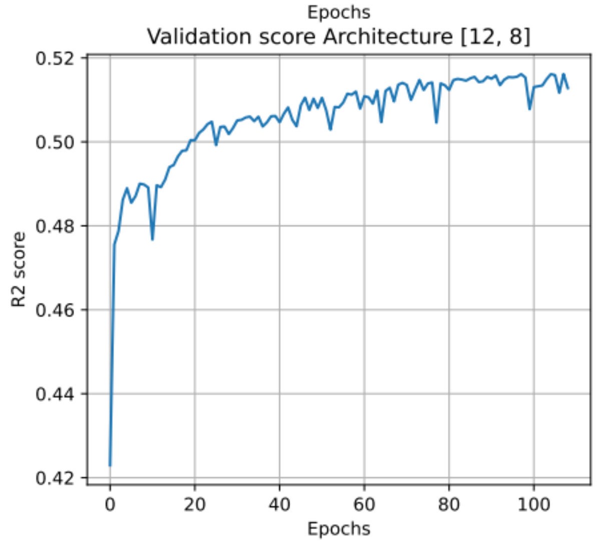


Output neuralNet(xTrain, yTrain)

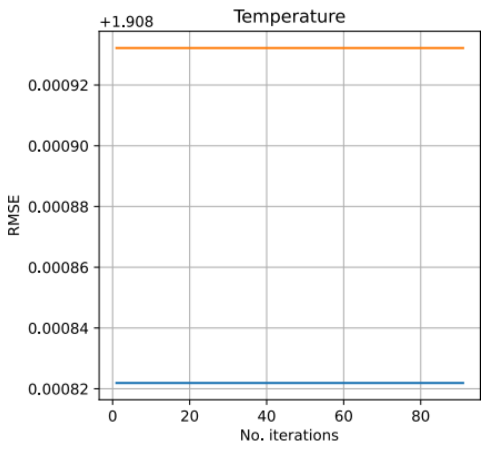


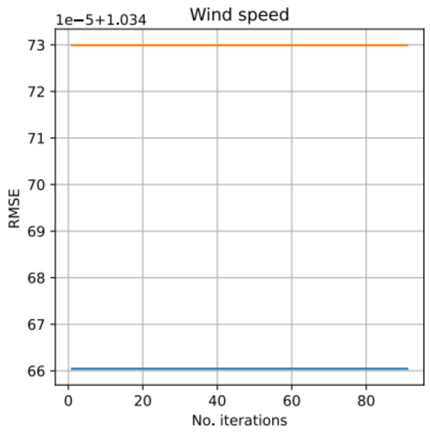


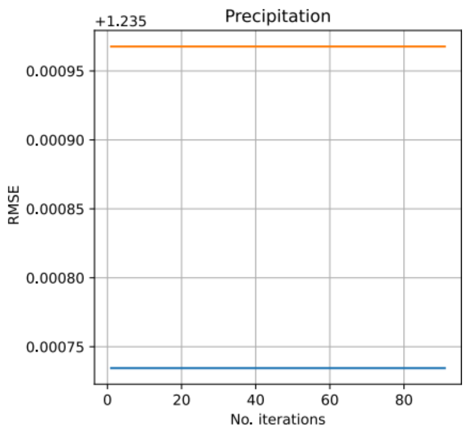




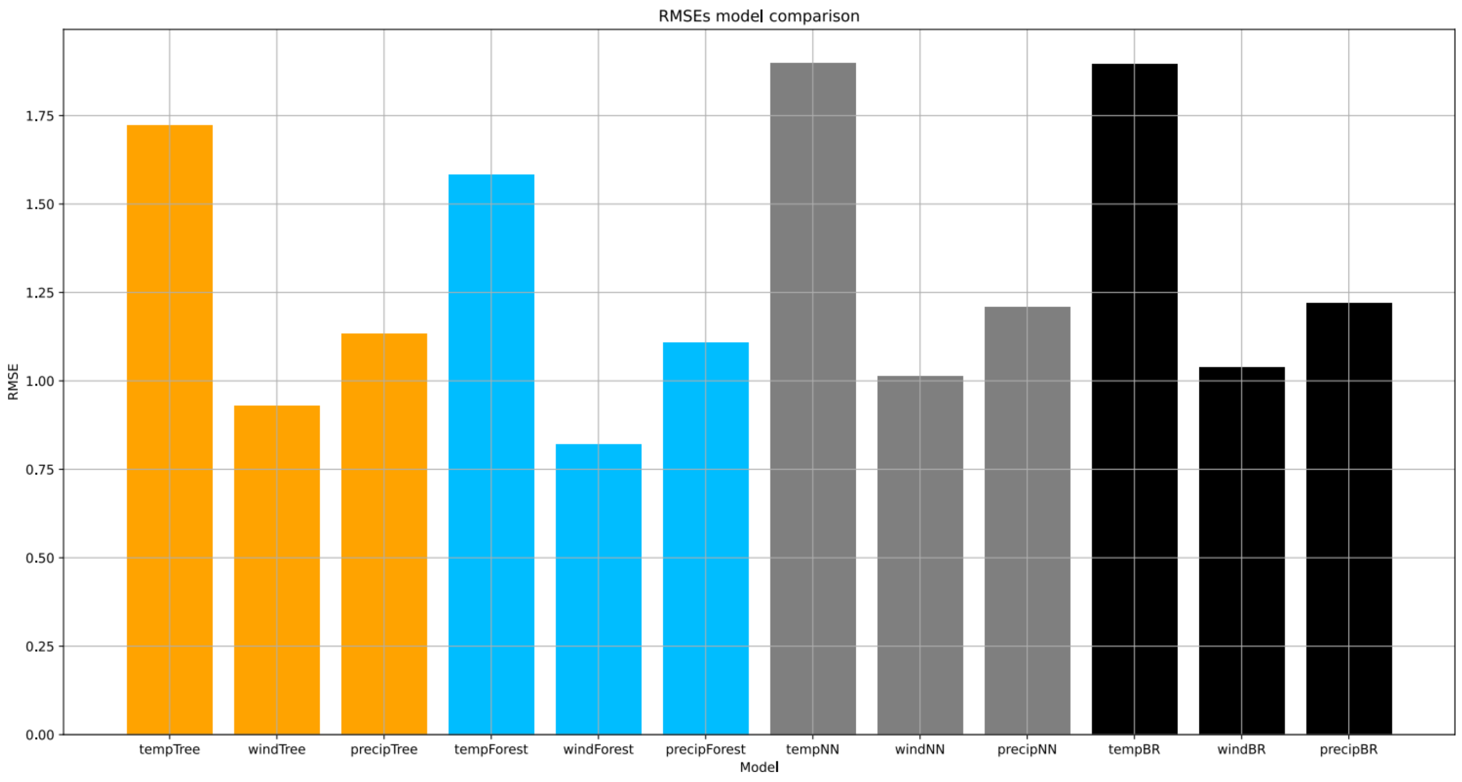
Output bayRidge(xTrain, yTrain)







Output plt.bar(...)


RMSEs model comparison
RMSE
tempTree
windTree
precipTree
tempForest
windForest
precipForest
tempNN
windNN
precipNN
tempBR
windBR
precipBR
Model