Homework7_YichiZhang

November 2, 2022

[1]:
```
## imports ##
import numpy as np
import pandas as pd

#Custom decision tree regressor
from statmodels.decisiontrees import DecisionTreeRegressor
#Custom random forest regressor
from statmodels.random_forest import RandomForestRegressor
#Custom gradient boosting regressor
from statmodels.gradientboosting import GradientBoostTreeRegressor
from statmodels.regression import LassoRegression
from statmodels.regression.utils.metrics import r2_score, mean_squared_error
from statmodels.regression.utils.preprocessing import StandardScaler
```

Load and split the data

[2]:
```
data = pd.read_csv("energydata_complete.csv")
data = data.set_index("date")
from sklearn.model_selection import train_test_split
X = data.iloc[:, 1:].values
Y = data.iloc[:, 0].values
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=.2, random_state=41)
```

1 a.1 Lasso regression model

[3]:
```
sc = StandardScaler()
X_train_transformed = sc.fit_transform(X_train)
X_test_transformed = sc.transform(X_test)
model_LassoRegression = LassoRegression(n_iter=10000, lr=1e-4, alpha=0.1)
model_LassoRegression.fit(X_train_transformed, Y_train)
```

[3]:
```
<statmodels.regression.regression.LassoRegression at 0x20a9d94d4f0>
```

[4]:
```
y_hat_LassoReg = model_LassoRegression.predict(X_train_transformed)
print('Lasso Regression - Root Mean Squared error:',
```

```
      round(mean_squared_error(y_true=Y_train, y_pred=y_hat_LassoReg, squared=␣
  ↪False), 4))
print('Lasso Regression - R-Squared:',
      round(r2_score(y_true=Y_train, y_pred= y_hat_LassoReg), 4))
```

```
Lasso Regression - Root Mean Squared error: 97.5554
Lasso Regression - R-Squared: 0.097
```

The Root Mean Squared error of Lasso regression model is 97.56. As the lower the RMSE, the better the model, this model does not fit the train dataset well. This model has the R-Squared value of 0.097, which means 9.7% of the variability observed in the target variable is explained by the regression model.

2 a.2 Decision Tree model

[5]:
```
model_tree = DecisionTreeRegressor(max_depth=5, min_samples_split=2)
model_tree.fit(X_train, Y_train.reshape(-1, 1))
```

[6]:
```
y_hat_tree = model_tree.predict(X_train)
print('Decision tree Regression - Root Mean Squared error:',␣
  ↪round(mean_squared_error(Y_train, y_hat_tree, False), 4))
print('Decision tree Regression - R-Squared:',␣
  ↪round(r2_score(Y_train,y_hat_tree), 4))
```

```
Decision tree Regression - Root Mean Squared error: 98.8429
Decision tree Regression - R-Squared: 0.073
```

The Root Mean Squared error of Decision tree model is 98.84. This model does not fit the train dataset well. This model has the R-Squared value of 0.073, which means 7.3% of the variability observed in the target variable is explained by the regression model.

3 a.3 Random forest model

[7]:
```
model_RF = RandomForestRegressor(n_trees = 5, max_depth = 6, min_samples_split␣
  ↪= 5)
model_RF.fit(X_train, Y_train)
```

[8]:
```
y_hat_RF = model_RF.predict(X_train)
print('Random Forest Regression - Root Mean Squared error:',␣
  ↪round(mean_squared_error(Y_train, y_hat_RF, False), 4))
print('Random Forest Regression - R-Squared:', round(r2_score(Y_train,␣
  ↪y_hat_RF), 4))
```

```
Random Forest Regression - Root Mean Squared error: 93.0082
Random Forest Regression - R-Squared: 0.1792
```

The Root Mean Squared error of Random Forest model is 93.0082. This model does not fit the train dataset well. This model has the R-Squared value of 0.1792, which means 17.92% of the

variability observed in the target variable is explained by the regression model.

4 a.4 GradientBoost model

[9]:
```
model_GradiantBoost = GradientBoostTreeRegressor(n_elements=10, learning_rate=0.
↪01)

#fit the model
model_GradiantBoost.fit(np.array(X_train), Y_train)
```

[10]:
```
y_hat_GB = model_GradiantBoost.predict(X_train)
print('Gradient Boost Regression - Root Mean Squared error:',␣
↪round(mean_squared_error(Y_train,y_hat_GB, False),4))
print('Gradient Boost Regression - R-Squared:', round(r2_score(Y_train,␣
↪y_hat_GB),4))
```

```
Gradient Boost Regression - Root Mean Squared error: 134.8229
Gradient Boost Regression - R-Squared: -0.7247
```

The Root Mean Squared error of GradientBoost model is 134.8229. This model does not fit the train dataset well. This model has the R-Squared value of -0.7247, which this model fits the dataset poorly.

4.1 Comparison

The Random Forest model has the lowest RMSE, which means it predicts the training data best. And it also has the highest R-square value which means it explains the most variability of the training dataset.

5 b.1 Lasso regression model

[11]:
```
y_hat_test_LassoReg = model_LassoRegression.predict(X_test_transformed)
print('Lasso Regression - Root Mean Squared error:',
      round(mean_squared_error(Y_test, y_hat_test_LassoReg, False), 4))
print('Lasso Regression - R-Squared:',
      round(r2_score(Y_test, y_hat_test_LassoReg), 4))
```

```
Lasso Regression - Root Mean Squared error: 97.1062
Lasso Regression - R-Squared: 0.093
```

1. Lasso regression has RMSE of 97.1062, which is similiar to the RMSE of training dataset.
2. The model has R-Squared value of 0.0927. 9.27% of the variability observed in the target variable is explained by the regression model.
3. Compared to training set, the RMSE is similiar between train and test set. Therefore the model is not overfitting.
4. R-Squared is very close to zero, which means the model does not fit the dataset well and it is underfitting.

6 b.2 Decision Tree model

[12]:
```
y_hat_test_tree = model_tree.predict(X_test)
print('Decision tree Regression - Root Mean Squared error:',␣
 ↪round(mean_squared_error(Y_test, y_hat_test_tree, False), 4))
print('Decision tree Regression - R-Squared:', round(r2_score(Y_test,␣
 ↪y_hat_test_tree), 4))
```

```
Decision tree Regression - Root Mean Squared error: 100.3224
Decision tree Regression - R-Squared: 0.0319
```

1. Decision tree has RMSE of 100.3224.
2. R-Squared value of 0.0319. And 3.2% of the variability observed in the target variable is explained by the model.
3. The RMSE of the test set is similiar to the train set, which means the model is not overfitting.
4. R-Squared is close to 0 which means the model is underfitting.

7 b.3 Random forest model

[13]:
```
y_hat_test_RF = model_RF.predict(X_test)
print('Random Forest Regression - Root Mean Squared error:',␣
 ↪round(mean_squared_error(Y_test, y_hat_test_RF.reshape(-1), False), 4))
print('Random Forest Regression - R-Squared:', round(r2_score(Y_test,␣
 ↪y_hat_test_RF.reshape(-1)), 4))
```

```
Random Forest Regression - Root Mean Squared error: 93.7816
Random Forest Regression - R-Squared: 0.154
```

1. Random Forest model has RMSE of 93.7816.
2. The model has R-Squared value of 0.154. This model predict well on the test dataset. And 15.4% of the variability observed in the target variable is explained by the regression model.
3. The R-Squared value of this model is the highest among the four models. It is the best model for this dataset.
4. The RMSE of the test set is similiar to the train set, which means the model is not overfitting.

8 b.4 GradientBoost model

[14]:
```
y_hat_test_GB = model_GradiantBoost.predict(X_test)
print('Gradient Boost Regression - Root Mean Squared error:',␣
 ↪round(mean_squared_error(Y_test, y_hat_test_GB, False), 4))
print('Gradient Boost Regression - R-Squared:', round(r2_score(Y_test, ␣
 ↪y_hat_test_GB),4))
```

```
Gradient Boost Regression - Root Mean Squared error: 133.8199
Gradient Boost Regression - R-Squared: -0.7225
```

1. GradientBoost model has RMSE of 133.8199

2. The model has R-Squared value of -0.7225. This means it does not predict the test dataset well.
3. Negative R-Squared value suggest this model is underfitting.
4. The RMSE of the test set is similiar to the train set, which means the model is not overfitting.

9 c

Do you see any bias and variance issues? How do you interpret each model output? (4x3=12 points)

9.1 c.1 Lasso Regression

The R-Squared value of this model is close to 0 which indicates it has high bias and the test RMSE does not increase which suggest the model has low variance.

9.2 c.2 Decision Tree Regression

The R-Squared value of this model is close to 0 which indicates it has high bias and the test RMSE does not increase which suggest the model has low variance.

9.3 c.3 Random Forest

The R-Squared value of this model is close to 1 which indicates it has low bias. And the test RMSE does not increase which suggest the model has low variance.

9.4 c.4 Gradient Boost

This model has high bias because the R-Squared value is negative and it does not make enough assumptions to the dataset. And the similiar RMSE values between train and test sets suggests it have low variance.

10 Important features

The following functions calculate the importance of features by mutating each feature and calculate the change of r square score. If a feature changes and the r square value also changes significiantly, then the feature is important to the model.

[15]:
```
def get_score_after_permutation(model, X, y, col_idx):
    X_mutated = X.copy()
    X_mutated[:, col_idx] = np.random.permutation(
        X_mutated[:, col_idx])
    permutated_score = r2_score(model.predict(X_mutated), y.reshape(-1))
    return permutated_score


def get_feature_importance(model, X, y, col_idx):
    baseline_score_train = r2_score(model.predict(X), y.reshape(-1))
    permutated_score_train = get_score_after_permutation(model, X, y, col_idx)
```

```
    feature_importance = baseline_score_train - permutated_score_train
    return feature_importance


def calculate_All_Feature_Importance(model, X, y):
    list_Feature_Importance = []
    for col_Index in range(X.shape[1]):
        list_Feature_Importance.append(
            get_feature_importance(model, X, y, col_Index))
    return list_Feature_Importance
```

11 d.1 Lasso Regression

[16]:
```
importances = calculate_All_Feature_Importance(model_LassoRegression, X_train,␣
 ↪Y_train)
for label, score in zip(data.columns, importances):
    print(f"{label} have the importance of {score}")
```

```
Appliances have the importance of -1.2566896409026782
lights have the importance of -0.2852172387786993
T1 have the importance of -8.368445184303553
RH_1 have the importance of 3.2194094972822995
T2 have the importance of 2.868195260783665
RH_2 have the importance of 2.4599037037742733
T3 have the importance of -4.197751688079762
RH_3 have the importance of -1.439936815446032
T4 have the importance of -0.9256331992126974
RH_4 have the importance of -1.3862833843381281
T5 have the importance of -0.3401224462060526
RH_5 have the importance of 7.960023511015507
T6 have the importance of 13.518239450763673
RH_6 have the importance of -1.2784814639617252
T7 have the importance of 4.1195217652591225
RH_7 have the importance of 0.35759985375270276
T8 have the importance of 9.121771550969882
RH_8 have the importance of -1.6388840009409904
T9 have the importance of 3.0933022151537415
RH_9 have the importance of 2.2962267386232043
T_out have the importance of -0.3375256462123133
Press_mm_hg have the importance of 28.744125224373548
RH_out have the importance of 0.9940699437162834
Windspeed have the importance of -0.45844399329220664
Visibility have the importance of -0.41679633636975666
Tdewpoint have the importance of 0.05884715342986624
rv1 have the importance of 0.10882309023635628
```

Press_mm_hg has the highest score of importance, and therefore it is the most important feature

of the model.

12 d.2 Decision Tree model

[17]:
```
importances = calculate_All_Feature_Importance(model_tree, X_train, Y_train)
for label, score in zip(data.columns, importances):
    print(f"{label} have the importance of {score}")
```

```
Appliances have the importance of 0.0
lights have the importance of 0.0
T1 have the importance of 0.0
RH_1 have the importance of 0.0
T2 have the importance of -10.446803996482123
RH_2 have the importance of 0.0
T3 have the importance of -0.46076493610095426
RH_3 have the importance of 0.0
T4 have the importance of 0.0
RH_4 have the importance of 0.0
T5 have the importance of 0.0
RH_5 have the importance of -0.8233568281804668
T6 have the importance of 1.0103223305481048
RH_6 have the importance of 0.0
T7 have the importance of 3.732086436757019
RH_7 have the importance of 0.6152929075156628
T8 have the importance of 0.14277960515826038
RH_8 have the importance of 0.4288490860232006
T9 have the importance of 1.4553354824045925
RH_9 have the importance of 0.0
T_out have the importance of 0.0
Press_mm_hg have the importance of -3.718543740120552
RH_out have the importance of 0.0
Windspeed have the importance of 0.0
Visibility have the importance of 4.0411236706652325
Tdewpoint have the importance of 0.0
rv1 have the importance of 0.0
```

RH_1 has the highest score followed by Appliances and T7. Therefore RH_1, Appliances and T7 are the most important feature of Decision Tree Regression model.

13 d.3 Random Forest

[18]:
```
importances = calculate_All_Feature_Importance(model_RF, X_train, Y_train)
for label, score in zip(data.columns, importances):
    print(f"{label} have the importance of {score}")
```

```
Appliances have the importance of 1.9308648073433492
lights have the importance of -0.9868054037016663
```

```
T1 have the importance of -0.9895014066068022
RH_1 have the importance of -0.3200427236045851
T2 have the importance of -0.06752775581890447
RH_2 have the importance of 0.05600440751085678
T3 have the importance of 0.24088913229968512
RH_3 have the importance of 0.8062264991777734
T4 have the importance of 0.0011094876277777388
RH_4 have the importance of 0.07393067167042666
T5 have the importance of 0.17285651467916985
RH_5 have the importance of 1.9215667817714728
T6 have the importance of -0.22513077351771216
RH_6 have the importance of 0.10933728401362686
T7 have the importance of 1.1188694862791628
RH_7 have the importance of 0.01125241440978364
T8 have the importance of 0.7741434868563033
RH_8 have the importance of 0.4741611984193952
T9 have the importance of 0.36221104059448983
RH_9 have the importance of 0.0
T_out have the importance of 0.0689157450452278
Press_mm_hg have the importance of 0.5986266086785808
RH_out have the importance of 1.5741549787493252
Windspeed have the importance of 0.06697385846242021
Visibility have the importance of 0.2161607968152115
Tdewpoint have the importance of 0.0
rv1 have the importance of 0.0
```

Appliances has the highest score followed by RH_5, T6 and RH_8. Therefore, Appliances, RH_5, T6 and RH_8 are the most important features for Random Forest model.

14 d.4 GradiantBoost

[19]:
```
importances = calculate_All_Feature_Importance(model_GradiantBoost, X_train,␣
 ↪Y_train)
for label, score in zip(data.columns, importances):
    print(f"{label} have the importance of {score}")
```

```
Appliances have the importance of 278.8108434619976
lights have the importance of -25.190488236717556
T1 have the importance of 14.160145839144207
RH_1 have the importance of 130.5446901852356
T2 have the importance of 85.31293066547232
RH_2 have the importance of 267.9208600334864
T3 have the importance of -190.48000854476186
RH_3 have the importance of 21.110476903028484
T4 have the importance of 19.952185609416574
RH_4 have the importance of 28.698451338666473
T5 have the importance of -23.541078299128458
RH_5 have the importance of 352.78947228324705
```

```
T6 have the importance of 371.7941123152086
RH_6 have the importance of -9.383296690524276
T7 have the importance of 517.981424188004
RH_7 have the importance of 129.94173882121186
T8 have the importance of 224.22134437545492
RH_8 have the importance of -29.281719012983558
T9 have the importance of 111.93121885726896
RH_9 have the importance of 88.36315787177591
T_out have the importance of 77.83386320146155
Press_mm_hg have the importance of 528.5360866856672
RH_out have the importance of 48.37737197972683
Windspeed have the importance of 44.16565111520549
Visibility have the importance of 27.756898795022153
Tdewpoint have the importance of 0.0
rv1 have the importance of -8.585786740232379
```

Here r-square values are negative and will cause very large score of importance after substraction. And therefore the score of rv1 is reasonable. So rv1 is the important feature of GradiantBoost model.