CSC 4780/6780
Fall 2022
Homework 10

October 25, 2022

This homework is due at 11:59 pm on Sunday, Nov 6. It must be uploaded to iCollege by then. No credit will be given for late submissions. A solution will be released by noon on Monday, Nov 7

it is always a good idea to get this done and turned in early. You can turn it in as many times as you like – iCollege will only keep the last submission. If, for some reason, you are unable to upload your solution, email it to me before the deadline.

Incidentally, I rarely check my iCollege mail, but I check my `dhillegass@gsu.edu` email all the time. Send messages there.

Be sure to rename your solution directory to match your name.

1 AutoML for Regression

I once worked with an old engineer who would quietly listen to younger engineers arguing over what each thought was the best solutions to a problem. Eventually, he would say, "There is no point in arguing about things that can be tested." And then he would go and do an experiment that ended the argument.

As we get better and better at working with these models, we can begin to guess which will be best. However, a lot of the time we can just try all of them.

In this exercise, you will get a data set for regression and you will use pycaret to find the best candidates and test them against each other.

1.1 Training and Comparing

`train_concrete.csv` and `test_concrete.csv` contain data about the compressive strength of several different concrete mixes: `https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength`

You will write a program called `concrete_train.py` that will use pycaret's `compare_models` (no turbo!) to try a large variety of regression algorithms on `train_concrete.csv`.

It will pick the best six (based on R^2) and it will tune (using at least 24 different parameter combinations) and finalize each before saving the finalized model to a pickle file. Thus six `.pkl` files will be created.

Run the program and save the output to `train.txt`.

`train.txt` should look like this:

```
*** Setting up session***
              Description             Value
0              Session id              8371
1                  Target             csMPa
2             Target type        Regression
...
18                    USI              6171
*** Set up: 1.89 seconds
                                    Model      MAE       MSE     RMSE  \
catboost                CatBoost Regressor   2.8945   18.6345   4.2833
lightgbm    Light Gradient Boosting Machine   3.4278   24.0534   4.8647
et                   Extra Trees Regressor   3.5456   26.9685   5.1605
rf                 Random Forest Regressor   3.8756   27.8363   5.2477
gbr            Gradient Boosting Regressor   3.9316   28.5122   5.3121
mlp                          MLP Regressor   5.1949   46.8561   6.8208
dt                 Decision Tree Regressor   5.0301   56.0932   7.3678
ada                     AdaBoost Regressor   6.3680   61.0570   7.7998
knn                  K Neighbors Regressor   7.3850   96.5281   9.7726
br                          Bayesian Ridge   8.1946  108.8475  10.4094
kr                            Kernel Ridge   8.2325  108.9195  10.4127
en                             Elastic Net   8.2139  109.0079  10.4165
ridge                     Ridge Regression   8.2163  109.0042  10.4161
lr                       Linear Regression   8.2163  109.0043  10.4161
lasso                     Lasso Regression   8.2147  109.0795  10.4200
ard      Automatic Relevance Determination   8.2609  109.4030  10.4368
huber                      Huber Regressor   8.1080  116.0969  10.6962
par             Passive Aggressive Regressor   9.6928  149.4162  12.0777
lar                  Least Angle Regression   9.9147  163.0199  12.6530
omp            Orthogonal Matching Pursuit  12.0965  216.9526  14.6893
svm               Support Vector Regression  12.0595  227.7054  15.0594
tr                      TheilSen Regressor   9.0357  232.6367  14.6477
llar            Lasso Least Angle Regression  13.6897  286.5223  16.8957
dummy                      Dummy Regressor  13.6897  286.5223  16.8957
ransac              Random Sample Consensus  10.4945  352.2832  17.8668

              R2   RMSLE    MAPE  TT (Sec)
catboost  0.9337  0.1358  0.1003     0.227
```

```
lightgbm  0.9143  0.1576  0.1191     0.016
et        0.9043  0.1622  0.1235     0.032
rf        0.9010  0.1772  0.1404     0.035
gbr       0.8986  0.1760  0.1383     0.016
mlp       0.8327  0.2217  0.1769     0.090
dt        0.8006  0.2311  0.1713     0.007
ada       0.7840  0.2828  0.2631     0.017
knn       0.6541  0.3188  0.2843     0.007
br        0.6097  0.3320  0.3135     0.007
kr        0.6092  0.3307  0.3135     0.009
en        0.6091  0.3315  0.3134     0.090
ridge     0.6091  0.3312  0.3131     0.089
lr        0.6091  0.3312  0.3131     0.219
lasso     0.6088  0.3318  0.3137     0.095
ard       0.6073  0.3314  0.3149     0.007
huber     0.5809  0.3235  0.3038     0.010
par       0.4758  0.3902  0.3636     0.007
lar       0.4182  0.4281  0.3720     0.007
omp       0.2359  0.4757  0.5022     0.007
svm       0.1996  0.4822  0.5051     0.009
tr        0.1558  0.3313  0.3066     0.171
llar     -0.0068  0.5397  0.6003     0.007
dummy    -0.0068  0.5397  0.6003     0.006
ransac   -0.2651  0.3615  0.3362     0.017
*** compare_models: 16.59 seconds
*** Best:
    CatBoostRegressor
    LGBMRegressor
    ExtraTreesRegressor
    RandomForestRegressor
    GradientBoostingRegressor
    MLPRegressor


*** 0 - CatBoostRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
0     3.4779  28.4514  5.3340  0.9139  0.1772  0.1307
...
9     2.5817  15.6806  3.9599  0.9387  0.1492  0.1040
Mean  3.0409  19.2967  4.3551  0.9313  0.1484  0.1077
Std   0.3354   5.1192  0.5742  0.0193  0.0199  0.0141

*** 1 - LGBMRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
```

```
0     3.5398  29.0514  5.3899  0.9121  0.1635  0.1275
...
9     2.9727  19.1409  4.3750  0.9252  0.1640  0.1160
Mean  3.1203  21.8118  4.6207  0.9222  0.1522  0.1100
Std   0.3422   6.4237  0.6787  0.0236  0.0230  0.0142

*** 2 - ExtraTreesRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
0     5.3227  53.8647  7.3393  0.8370  0.2351  0.2000
1...
9     5.0418  38.2233  6.1825  0.8506  0.2439  0.2172
Mean  4.9365  41.6118  6.4389  0.8523  0.2119  0.1816
Std   0.2409   5.1511  0.3898  0.0196  0.0268  0.0253

*** 3 - RandomForestRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
0     4.6211  46.6803  6.8323  0.8588  0.2290  0.1846
...
9     4.6621  32.9120  5.7369  0.8714  0.2293  0.2000
Mean  4.5181  35.5683  5.9522  0.8745  0.2030  0.1707
Std   0.1097   4.5604  0.3733  0.0121  0.0318  0.0272

*** 4 - GradientBoostingRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
0     3.3277  25.1146  5.0114  0.9240  0.1740  0.1271
...
9     2.9214  19.6030  4.4275  0.9234  0.1694  0.1215
Mean  3.1014  20.4411  4.4966  0.9272  0.1542  0.1114
Std   0.2730   4.1364  0.4706  0.0171  0.0203  0.0145

*** 5 - MLPRegressor ***
Fitting 10 folds for each of 24 candidates, totalling 240 fits
         MAE      MSE    RMSE      R2   RMSLE    MAPE
Fold
0     5.6160  56.9794  7.5485  0.8276  0.2771  0.1929
...
9     5.1128  37.7620  6.1451  0.8524  0.2474  0.2167
Mean  5.1949  46.8561  6.8208  0.8327  0.2217  0.1769
Std   0.4478   7.8475  0.5772  0.0343  0.0291  0.0239
Transformation Pipeline and Model Successfully Saved
*** Tuning and finalizing: 165.12 seconds
*** Total time: 183.60 seconds
```

(Yes, depending on the versions of the libraries that you have installed, there may be some warnings from this process. I'm not showing those here.)

When I run this, I end up with a pickle file for the top six models:

- `LGBMRegressor.pkl`
- `CatBoostRegressor.pkl`
- `MLPRegressor.pkl`
- `ExtraTreesRegressor.pkl`
- `RandomForestRegressor.pkl`
- `GradientBoostingRegressor.pkl`

1.2 Testing

You will write a program called `concrete_test.py` that will scan the current directory for `.pkl` files. It will use pycaret to load those in one at time.

Each model will be tested on `concrete_test.py`. The program will print the time that inference required and the R^2 value.

Run the program and save the output to `test.txt`

My `test.txt` looks like this:

```
GradientBoostingRegressor:
    Inference: 0.0095 seconds
    R2 on test data = 0.9110
RandomForestRegressor:
    Inference: 0.0319 seconds
    R2 on test data = 0.8815
AdaBoostRegressor:
    Inference: 0.0147 seconds
    R2 on test data = 0.7239
ExtraTreesRegressor:
    Inference: 0.0170 seconds
    R2 on test data = 0.9007
MLPRegressor:
    Inference: 0.0052 seconds
    R2 on test data = 0.7861
CatBoostRegressor:
    Inference: 0.0030 seconds
    R2 on test data = 0.9079
LGBMRegressor:
```

```
Inference: 0.0071 seconds
R2 on test data = 0.9101
```

Which would you use if accuracy was most important? What if speed was also really important?

2 X^2 testing for independence between categorical variables

Some times we will look at two categorical variables and try to figure out if they are related. Does knowing that the mouse has a particular gene tell us anything about the probability that it will get cancer?

You are given a csv with the results of this sort of experiment called `mice.csv`. Write a program `check_mice.py` that does the analysis. Put the analysis into a LaTeX file. (`mice.tex`) Convert that to a PDF `mice.pdf`). Include bot files in your zip file.

For example, you should start out with a contingency table: (I did these examples with different data.)

Gene	No Cancer	Has Cancer	
R	34	2	**36**
J	4	45	**49**
K	17	18	**35**
	55	**65**	**120**

Then show conditional proportions:

Gene	No Cancer	Has Cancer	
R	94.4%	5.6%	**30.0%**
J	8.2%	91.8%	**40.8%**
K	48.6%	51.4%	**29.2%**
	45.8%	**54.2%**	

Then show the expected counts if the gene and cancer were independent:

Gene	No Cancer	Has Cancer	
R	16.5	19.5	**36**
J	22.5	26.5	**49**
K	16.0	19.0	**35**
	45.8%	**54.2%**	

Use the two tables to find X^2:

$$X^2 = 62.379$$

Note the degrees of freedom. (It is 2.)

And do a p-test:

$$p = 2.853273173286652 \times 10^{-14}$$

And then give proclamation: "It seems very, very unlikely that we would have seen these numbers if the gene and cancer were independent."

3 Criteria for success

If your name is Fred Jones, you will turn in a zip file called `HW10_Jones_Fred.zip` of a directory called `HW10_Jones_Fred`. It will contain:

- `concrete_train.py`
- `concrete_test.py`
- `check_mice.py`
- `test.txt`
- `train.txt`
- `mice.tex`
- `mice.pdf`
- `train_concrete.csv`
- `test_concrete.csv`
- `mice.csv`

Be sure to format your python code with black before you submit it.

We would run your code like this:

```
cd HW10_Jones_Fred
python3 concrete_train.py
python3 concrete_test.py
python3 check_mice.py
```

Do this work by yourself. Stackoverflow is OK. A hint from another student is OK. Looking at another student's code is *not* OK.

The template files for the python programs have import statements. Do not use any frameworks not in those import statements.