CS F320 Foundation of Data Science

Assignment-2 Report

Parth Gupta - 2020B4A72235H
Suraj Nair - 2020A7PS0051H
Swapnil Sharma - 2020B4PS2259H

Description of the model:

With the help of provided code we are able to do feature reduction using 4 different techniques - Pearson correlation,Principal component analysis,Greedy forward and backward algorithm. The principal idea behind all these models is to reduce the dimensionality as visualizing data in higher dimensions is not feasible. Hence we use these techniques to scale down the dimensions by discarding the weak features , i.e , features that can be omitted . So now after deciding upon the dimension , we can compare which of the above 4 techniques yielded the optimal model.

We implemented the algorithms in the following way:

1. Initially, we read the data from the CSV file provided using the pandas module present in python. Then we normalize the data using appropriate methods.
2. Next, we perform a 60-20-20 train-test-validation set split for training the model and testing it later.

Pearson correlation technique

- First of all we compute the correlation matrix (DxD matrix) using appropriate libraries.
- Then we defined a function called correlation_feature_selection
- that takes in the dataset and a threshold value for the correlation. Any feature having correlation value w.r.t target attribute less than the threshold is omitted.
- Now from among these selected features we check which features are highly correlated and drop one of the features . This is because of the fact that independent features are uncorrelated.
- This process is repeated with different threshold values to get different feature subsets.
- Training and testing errors are printed for each model.

Threshold	selected set	Rms Train Error:	Rms Val Error:
0.0	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]	0.9071130767605796	0.9536468778836134
0.0001	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]	0.9071784409679464	0.9536225868647628
0.0046	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23]	0.9104401927087323	0.9554484600666352
0.0049	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 23]	0.9106452085247536	0.9557967585697402
0.01	[1, 2, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21]	0.9106483294980505	0.9558855873564374
0.0113	[0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21]	0.9112272734670485	0.9573865982780677
0.0125	[0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21]	0.9127117841897157	0.9568072959603628
0.0136	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21]	0.9135848178291398	0.9582420712139179
0.0201	[0, 1, 2, 3, 4, 5, 6, 10, 11, 13, 14, 15, 17, 18, 19, 20, 21]	0.9165515746712013	0.959215397800891
0.0268	[0, 1, 2, 3, 4, 6, 10, 11, 13, 14, 15, 17, 18, 19, 20, 21]	0.9183361060852462	0.9637287006429869
0.0298	[0, 1, 2, 3, 4, 6, 10, 11, 13, 14, 15, 17, 18, 20, 21]	0.9185664536001322	0.9640822224804217
0.0326	[0, 1, 2, 3, 4, 6, 10, 11, 13, 15, 17, 18, 20, 21]	0.918625518317844	0.9641375173006529
0.0357	[0, 1, 2, 3, 4, 10, 11, 13, 15, 17, 18, 20, 21]	0.919689500052614	0.9651628989204953
0.0477	[1, 2, 3, 4, 10, 11, 13, 15, 17, 18, 20, 21]	0.92047225978144	0.96392548730783
0.0513	[1, 2, 3, 4, 15, 11, 13, 15, 18, 20, 21]	0.9205922821144515	0.9644730096631439

0.0593	[1, 2, 3, 4, 10, 11, 15, 18, 20, 21]	0.9210406923002188	0.9650240235133826
0.0663	[1, 2, 4, 10, 11, 15, 18, 20, 21]	0.9359920189849461	0.9793237200103867
0.0768	[2, 4, 10, 11, 15, 18, 20, 21]	0.9486674613923171	0.9924367607399521
0.0809	[2, 10, 11, 15, 18, 20, 21]	0.9568601839540388	1.0008308854476715
0.0831	[2, 10, 15, 18, 21, 22]	0.9618363005043196	1.003670707240463
0.0901	[2, 10, 15, 18, 20]	0.964736793823915	1.0075250245599467
0.0911	[18, 2, 10, 20]	0.9647419347722301	1.0075869827528734
0.099	[2, 10, 20]	0.9672095155812279	1.01020404625606
0.1096	[10, 20]	0.9672815450630281	1.0104831907032081
0.1158	[20]	0.9677450579295785	1.0102368794250867

Principal component analysis

- First of all we define a function called PCA that computes the eigen vectors and stores them in an empty list.
- Next we compute the eigen values of the corresponding eigen vectors in the previous step (which are by default arranged in descending order).
- A capture function is implemented to calculate the total variance captured by the corresponding model.

No of pc	**RMS Train Error**	**RMS Test Error**	**variance captured**
1	0.9774151094499796	1.0198238150138903	35.7943%
2	0.9772103695480466	1.0190223706784114	63.2049%
3	0.9770261895610534	1.0183763544843145	70.8804%
4	0.9656829631159032	1.0198238150138903	77.7757%

5	0.9652166121643306	1.0190223706784114	81.8851%
6	0.9650153960007782	1.0183763544843145	85.607%
7	0.9647249744104737	1.0177718965333573	89.2662%
8	0.9647249744104737	1.017837518002754	91.3941%
9	0.9644128354480103	1.0173884187260456	93.3765%
10	0.9518898490672398	1.0067574803914126	94.8248%
11	0.9415174269173228	0.9910088594843237	95.8313%
12	0.9409400343096458	0.9912754365064659	96.5309%
13	0.9369846985738641	0.992024542004359	97.0932%
14	0.9344629159975517	0.9922422880838173	97.6156%
15	0.9321578710156099	0.9866792177666982	98.065%
16	0.9321547907241743	0.9865677058509936	98.5266%
17	0.9296710501804802	0.9878880488363856	98.8842%
18	0.9253887476044154	0.9828724308776738	99.1624%
19	0.9253700042551983	0.9829066890327206	99.4135%
20	0.9202959956564507	0.9736929707739788	99.5804%
21	0.918304587154033	0.9743542944287986	99.7394%
22	0.9148052314003494	0.9708336681288877	99.791%
23	0.9121439940535454	0.9661266755994968	99.8997%
24	0.9074080442882896	0.9591605452578492	99.9853%
25	0.9071130767605797	0.9592240975470254	100.0%
26	0.9071130767605796	0.9592240975470253	100.0%

Greedy Forward algorithm

- Here we have defined a function called greedy_forward in which we take in the dataset and an empty Feature set.
- We recursively proceed ahead to add selected features to the Feature set and compute the training and testing error. Here the testing error is done on the **validation set.**
- In every iteration we select that feature which on including in the feature set gives minimal training error and we keep doing this recursively.

Selected Feature Set	**Rms error on training set**	**Rms error on validation set**
[20]	0.9677450579295785	1.0102368794250867
[20, 1]	0.9597520835621702	1.0045082979993192
[20, 1, 13]	0.9483566259431007	0.9915796491421796
[20, 1, 13, 3]	0.9417394052858028	0.9835231077605336
[20, 1, 13, 3, 15]	0.939131026927505	0.9812926850487654
[20, 1, 13, 3, 15, 5]	0.9372643131302054	0.9777978718723715
[20, 1, 13, 3, 15, 5, 21]	0.9356502318665331	0.9757791973372938
[20, 1, 13, 3, 15, 5, 21, 4]	0.9341454054304765	0.9727802412383466
[20, 1, 13, 3, 15, 5, 21, 4, 16]	0.9184671365957269	0.9626081743856978
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2]	0.9148683369446169	0.9587026915660296
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7]	0.9137953685278691	0.9575769458468606
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9]	0.9136177970208559	0.9566542355877022
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10]	0.9128442767006457	0.9557536502747487
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18]	0.9124770084186007	0.95485223695966

[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14]	0.9089431503510044	0.9537930639344476
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22]	0.9088006256324471	0.9535795044685403
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17]	0.9087919762229608	0.9533915895285424
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11]	0.9081952021940989	0.9533256380234056
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23]	0.9081433531780388	0.9528904010549382
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6]	0.9075884867532843	0.9528254557800214
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12]	0.907581747770789	0.95279851047578
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12, 19]	0.9075817477408673	0.9527984389682738
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12, 19, 24]	0.9075170973313453	0.9528246423510919
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12, 19, 24, 25]	0.9075170973313454	0.9528246423510917
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12, 19, 24, 25, 8]	0.9072365911425667	0.9530991791044892
[20, 1, 13, 3, 15, 5, 21, 4, 16, 2, 7, 9, 10, 18, 14, 22, 17, 11, 23, 6, 12, 19, 24, 25, 8, 0]	0.9071130767605796	0.9536468778836136

Greedy Backward algorithm

- Here we have defined a function called greedy_backward in which we take in the dataset and a Feature set containing all the 26 features.
- We recursively proceed ahead to remove selected features from the Feature set and compute the training and testing error. Here the testing error is done on the **validation set.**
- In each step we drop that feature which on removal gives the least rmse error because this implies that dropping that feature does not really contribute to error and we can reduce the dimensionality too.

feature_set	**Rms error on training set**	**Rms error on validation set**
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]	0.9072365911425667	0.9530991791044894
[1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]	0.9075170973313453	0.9528246423510917
[1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25]	0.9075170973313454	0.9528246423510915
[1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]	0.9075817477408673	0.952798438968274
[1, 2, 3, 4, 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]	0.9081004447349058	0.952791460606085
[1, 2, 3, 4, 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23]	0.908109561400084	0.952828999777177
[1, 2, 3, 4, 5, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22,	0.9081433531780388	0.9528904010549382

23]		
[1, 2, 3, 4, 5, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 23]	0.9082823817371993	0.9531345415543692
[1, 2, 3, 4, 5, 7, 9, 10, 11, 13, 14, 15, 16, 18, 20, 21, 23]	0.9083099989794857	0.9534580689312686
[1, 2, 3, 4, 5, 7, 9, 10, 13, 14, 15, 16, 18, 20, 21, 23]	0.9089429063295197	0.9537635964855874
[1, 2, 3, 4, 5, 7, 9, 10, 13, 14, 15, 16, 18, 21, 23]	0.9089429289093781	0.9537714782193671
[1, 2, 3, 4, 5, 7, 9, 10, 13, 14, 15, 16, 18, 21]	0.9089438707938511	0.9538003006037813
[1, 2, 3, 4, 5, 7, 10, 13, 14, 15, 16, 18, 21]	0.9090270243369919	0.9544790582852751
[1, 2, 3, 4, 5, 7, 10, 14, 15, 16, 18, 21]	0.9094870296034431	0.9555161268268422
[1, 2, 3, 4, 5, 10, 14, 15, 16, 18, 21]	0.9099135063683741	0.9561115494188908
[1, 2, 3, 4, 10, 14, 15, 16, 18, 21]	0.910487746984965	0.9578854415223292
[1, 2, 3, 4, 10, 15, 16, 18, 21]	0.9146850112524448	0.9596519416418627
[1, 2, 3, 4, 10, 15, 16, 21]	0.9151668670649231	0.9606897247029885
[1, 2, 3, 4, 10, 15, 16, 21]	0.9156548845610392	0.9604745492346012
[1, 2, 3, 4, 15, 16, 21]	0.9156548845610392	0.9165112374359143
[1, 2, 3, 4, 15, 16]	0.9165112374359143	0.9604745492346012
[1, 3, 4, 15, 16]	0.9291693854447325	0.9763396265281576
[1, 4, 15, 16]	0.9291693854447325	0.9763396265281576
[4, 15, 16]	0.9489700456161633	0.9928293400859011
[4, 16]	0.9614836982979293	1.0109645244822458
[4]	0.9761240373799284	1.0228309700651779

Comparative analysis :

From the above 4 tables the highlighted values are the best models in the corresponding techniques. Now we proceed to check which among these models is the best model by using the last 20% testing dataset.

METHOD	rms train error	rms test error
Using PEARSON_CORRELATION Feature Reduction	0.9127117841897157	0.9626434365256136
feature selected: ['T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed']		
Using GREEDY_FORWARD Feature Reduction	0.9075817477408673	0.9585112680703396
feature selected: ['RH_out', 'RH_1', 'RH_7', 'RH_2', 'RH_8', 'RH_3', 'Windspeed', 'T3', 'T9', 'T2', 'RH_4', 'RH_5', 'T6', 'T_out', 'T8', 'Visibility', 'RH_9', 'RH_6', 'Tdewpoint', 'T4', 'T7', 'Press_mm_hg']		
Using GREEDY_BACKWARD Feature Reduction	0.9081004447349058	0.960065353765553
feature selected: ['RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'RH_4', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint']		
Using PCA Feature Reduction	0.9074080442882896	0.9591605452578492
No of principal Components = 24 Eigen values =[1.34888084e-02 2.23600328e-02 2.83831479e-02 4.15383202e-02 4.35872890e-02 6.56048176e-02 7.26852631e-02 9.34011634e-02 1.17399472e-01 1.20575154e-01 1.36473367e-01 1.46902977e-01 1.82742278e-01 2.62929677e-01		

3.78344963e-01 5.17875287e-01 5.55881603e-01 9.55876112e-01 9.72282308e-01 1.07351140e+00 1.80128315e+00 2.00506153e+00 7.16053566e+00 9.35059987e+00]		
Using No Feature Reduction and using closed form	0.9071130767605796	0.9592240975470253

From the above table we observe that the Greedy forward feature selection model containing 22 features yields the optimal model with minimal testing error compared to remaining models.

OBSERVATION :

The above result suggests that the optimal model will have 22 features (tabulated in greedy forward table) . But if a person has computation constraints and wants a model which has lesser dimensionality , it can be done by compromising on the test error ,i.e, features can be reduced but we might have more testing error .