Title: Case Study 1, Feature Engineering

Ronit Ganguly: 2487190G:

Introduction

In this case study, we are given a dataset of patients who have suffered from Spinal Cord Injury (SCI). Now, post SCI, patients have a tendency of developing Central Neuropathic Pain. It has been recorded that almost 50% of such cases show symptoms of this pain. What more of a problem is, there is no cure but prevention. Early detection is required to provide with selective treatments because the treatment medication has strong side effects. Therefore, data science can play its role in predicting whether a patient will show such symptoms or not.

We have to build a classifier on the data that have been given. The data have been taken from the patients who have either developed Central Neuropathic Pain or they have not, within 6 months post SCI. The sample size is 18 patients (with 10 readings each) where 10 patients have developed the pain while the remaining 8 patients have not. This method involved taking brain readings of the patients from 48 different locations of the brain(using electrodes) highlighting 9 features. Moreover, the features were categorized into 2 groups : Eye opened and Eye closed. Our prediction algorithm at the end should be able to predict whether the patient will develop the pain.

Objective: To build a classifier without unnecessary features.
The data has more number of features which make it complicated to understand, and increase the chance of "overfitting". Therefore, our classifier, if developed on the raw data, may have poor performance on predicting new cases. Or at least, our data will be less complicated with probably reasonable performance (with fewer features), which explains that some features might be redundant or do not contribute much to the overall prediction. Therefore, this is a task of feature engineering.

Feature engineering is a crucial step in building a model. A model can have m numbers of data (rows) while n number of features (columns). The more the number of features, the more complex our model is. Moreover, having higher number of columns than rows, will likely lead to "overfitting", a condition where the model remembers the data points and doesn't learn anything much [1]. As a consequence, our model may not be able to accurately predict all the new data.

Methods:

There are 3 groups of feature selection: 1) Wrapper 2) Filter and 3) Embedded [2]. A wrapper method is an iterative feature selection process where it trains the model in every iteration with some features and finally gives the best features and a trained model on those features. Filter method, however, doesn't train the model but instead uses certain metrics to evaluate the importance of the features with respect to the true values or correlations amongst them. Embedded methods, on the other hand, lowers or completely removes the influence of some features.

Sub-categories:

1) Wrapper methods: Forward selection, backward selection, recursive feature elimination

2) Filter methods: Chi2 method, ANOVA test, Pearson Correlation, Mutual Information

3) Embedded methods: Ridge Regression, Lasso regression

In this study, I am going to talk about the combination of filter method and wrapper method to attain good accuracy on new unseen data**:**

1. I have used Mutual Information for my first choice of feature selection method.

2. Secondly, I have taken recursive feature elimination method.
3. Thirdly, I took Pearsonr correlation method to remove some remaining highly correlated columns.
4. Finally, classifier used is sklearn.linear_model.LogisticRegression [3]

Firstly, I am splitting the data into training set and test set using a custom function. It's called custom_train_test_split() and it splits the training set and test set "randomly" in such a way that:

- Both sets will have complete patient records i.e. continuous 10 rows of data.
- Test set will have both the classes for measuring the accuracy otherwise, accuracy formula will give inconsistent results and specificity and sensitivity formulas may give zero division error.
- It also uses random.seed() to provide control on the random sampling of train-test splits.
- It returns DataFrame objects which are easier to deal with in row-column operations.

Mutual Information: This method is able to explain not only the linear relationship between two random variables but also it detects and shows the non linear relationship (if it exists). Sklearn has an exact representation of this method under its feature_selection module. It's called sklearn.feature_selection.mutual_info_classif [4].

Secondly, RFE (recursive feature elimination) method which is also given by sklearn [4]. This method recursively eliminates the weakest feature and keeps the optimum ones in the set of features.

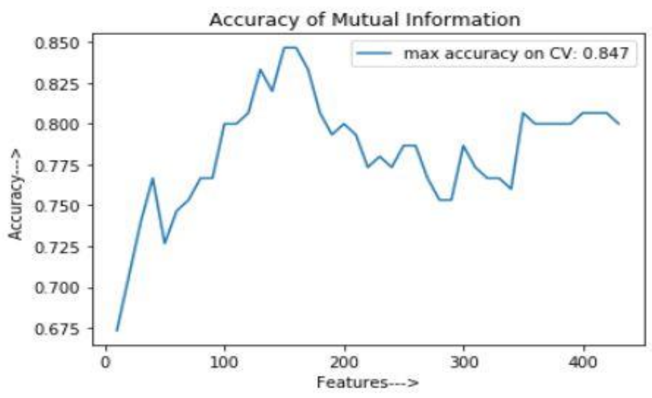
Additionally, pd.DataFrame.corr(method='pearson') will give me a matrix with correlation among columns.

Also, I am using sklearn's LeaveOneGroupOut [5] function to perform leave one group out cross validation on the training data set. This function runs folds and keeps aside one group for testing on each iteration. These groups will have to be provided explicitly.

Finally, while performing RFE, I have used it in a pipeline to make it simpler.

Results:

- Firstly, I split the data using my custom_train_test_split() function into 15 groups (for training) and (3 groups) for testing.
- random_state taken in my custom_train_test_split() is 123
- I got the optimum number of features from Mutual Information, and RFE using LeaveOneGroupOut cross validation.
- LeaveOneGroupOut CV accuracy of Mutual Information:



It shows that with features 150, the CV accuracy is 0.847.

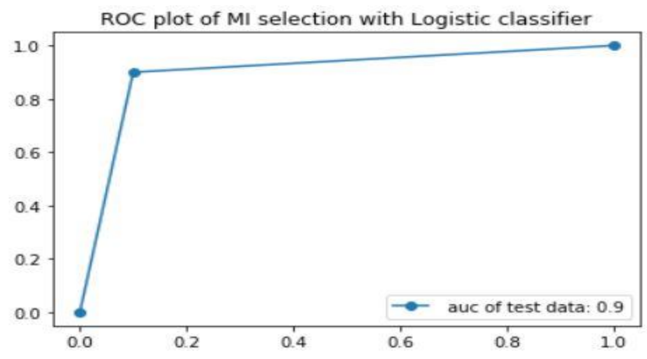
- Trained the model from scratch on the optimum new 150 features and found the accuracy on the unseen 3 groups

- Now accuracy check result on unseen 3 groups dataset:
-

Training Accuracy	0.94
Testing Accuracy	0.9
Specificity	0.9
Sensitivity	0.9
AUC	0.9

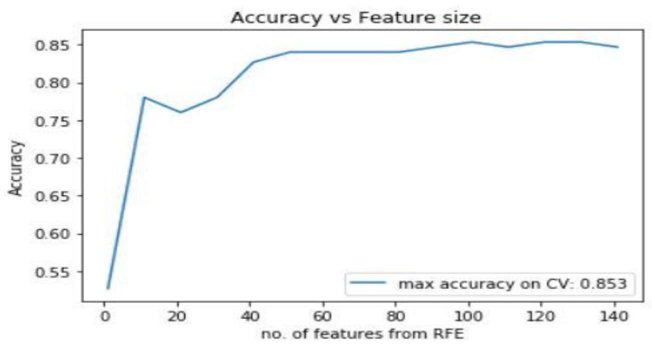
- ROC-AUC plot of 150 features selected by Mutual Information on unseen 3 groups:

```
<matplotlib.legend.Legend at 0x6ea093ec88>
```



-
- This shows that the 150 features yielded 90% accuracy on the test set.
- Now, I feed the 150 features to RFE- Logistic pipeline and find the LeaveOneGroupOut cross validation accuracy:

```
<matplotlib.legend.Legend at 0x6ea0e60b70>
```

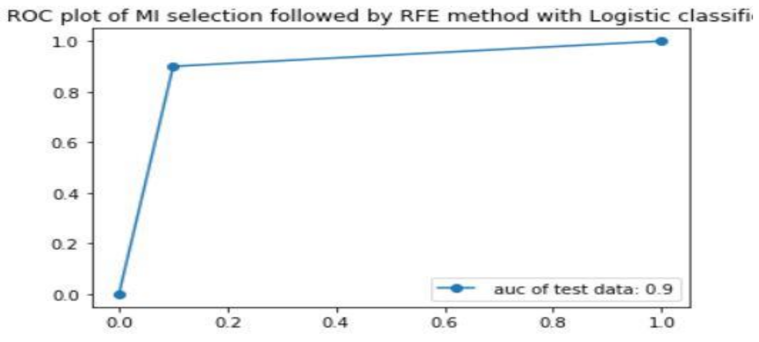


-
- This shows that RFE has selected 101 features from the LeaveOneGroupOut cv and cv accuracy is 0.853.
- Trained the model from scratch on the optimum new 101 features and found the accuracy on the unseen 3 groups.
- Accuracy check result on unseen 3 groups:
-

Training Accuracy	0.933
Testing Accuracy	0.9
Specificity	0.9
Sensitivity	0.9
AUC	0.9

- ROC-AUC plot of 101 features selected by RFE on unseen 3 groups:

```
<matplotlib.legend.Legend at 0x6ea0ecadd8>
```

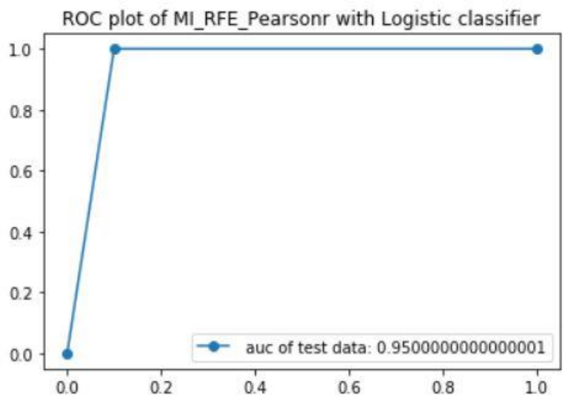


- Applying Pearson correlation function of Pandas: pd.DataFrame.corr('pearson') to check if highly correlated columns are still present and if we remove them, should the accuracy increase?
- Found 29 correlated columns above abs(threshold=0.9). After removing them we are left with 72 columns:
-

Training Accuracy	0.90
Testing Accuracy	0.933
Specificity	0.9
Sensitivity	1.0
AUC	0.95

- ROC-AUC curve after removing some highly correlated columns:

```
<matplotlib.legend.Legend at 0x346e17ce48>
```



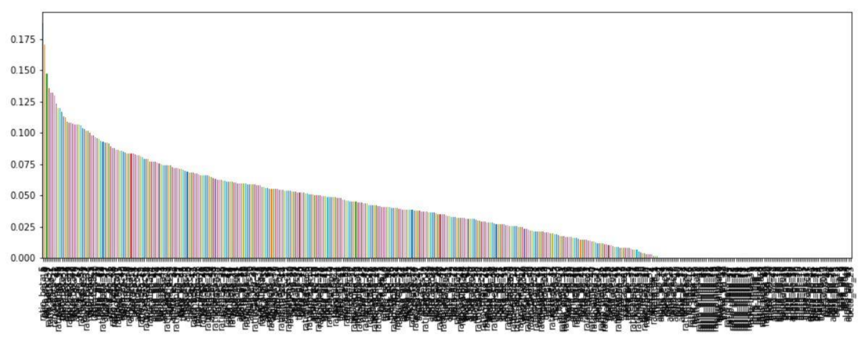
-
- Final Table after taking Mutual Information --> RFE ---> Pearson correlation:

	Mutual Information	RFE	Pearson correlation
Training Accuracy	0.94	0.933	0.906
Testing Accuracy	0.9	0.9	0.933
Specificity	0.9	0.9	0.9
Sensitivity	0.9	0.9	1.0
AUC	0.9	0.9	0.95

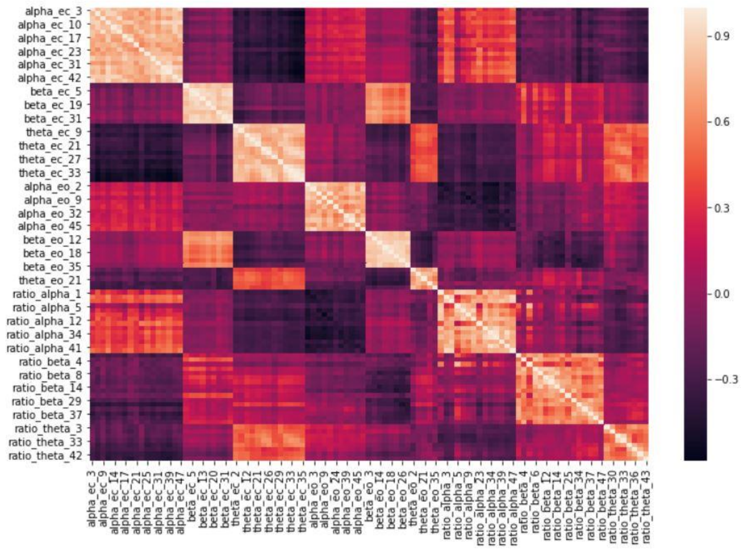
Discussion:

- I started with filter method because the number of features are high and wrapper method would take much time to compute.
- I have used mutual information because it takes care of the non-linear dependencies as well between variables, unlike Chi2 or ANOVA test which explain linear dependencies [7]. Chi2 better explains categorical variables [8]. Also, an initial run of ANOVA test yielded 350 features which proves that there are non-linear dependencies because with mutual information, I got 150 features. At this moment, after ANOVA, I did not choose any wrapper method because it would take too much time. So, with all these points, I proceeded with mutual information as my filter method.

```
<matplotlib.axes._subplots.AxesSubplot at 0x6e9c355e80>
```

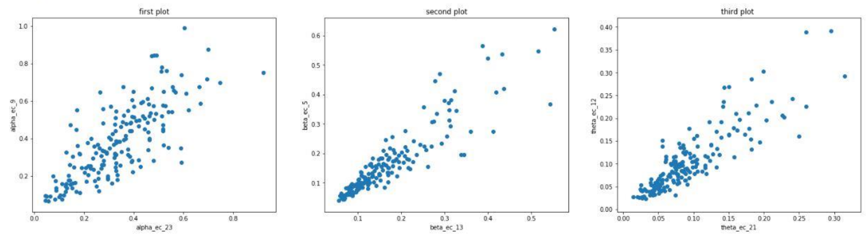


-
- The above bar graph shows the mutual information score of each feature with respect to the target variable. Higher the score, higher the importance of the feature. Also, we see that there is an exponential decrease in the importance of the features, where some don't even have a score at all.
- Second step was a wrapper method: RFE (Recursive Feature Elimination) which is able to remove weak features which do not improve the accuracy.
- Up to this moment, there is no change in accuracy(on unseen groups) observed, but Mutual information yielded 150 best features whereas, from those 150 features, RFE is able to bring it down to 101. Moreover, both the processes have same accuracy on the validation set. This shows that 101 features can explain the data as much as 150 features. This means our model is simpler now.
- Next, I checked if there is any highly correlated column left.:

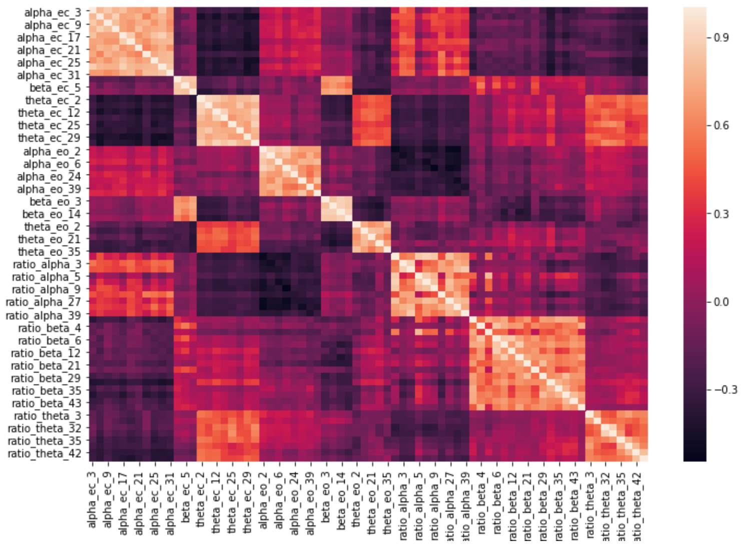


-

```
Text(0,0.5,'theta_ec_12')
```



- 29 columns were found to have high correlations and dropping them increased the accuracy(mentioned in the results section above)



- Correlation above 0.9 is now not present apart from the diagonal elements, and a more distinguishable diagonal has come up. On the diagonal, the correlation of a feature is with itself, so there is nothing to worry about it.

- In conclusion, selected mutual information because it removed non-linear dependencies along with linear ones, and made the job easier for wrapper method RFE. Also, at first with just Mutual Information method, the training accuracy was 0.94, reduced to 0.933, and finally stood at 0.906. This proves that with more and more feature reduction, bias is added. This also means that it is going away from high variance problem. To prove that this is generalizing, we can check the final accuracy on the unseen data(validation set). And in this case, it has increased to 0.933. This proves that our model is not overfitting but now generalised.

References:

[1] Jorge R. Vergara, Pablo A. Este´vez ***' A review of feature selection methods based on mutual information'*** Springer London pp. 1 [online] Available at < https://link-springer-com.ezproxy.lib.gla.ac.uk/article/10.1007%2Fs00521-013-1368-0> (Accessed on: 19 November 2019)

[2] . Guyon I, Elisseeff A (2003) ***'An introduction to variable and feature selection***.' J Mach Learn Res 3:1157–1182

[3] sklearn.linear_model.LogisticRegression Available at < https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html >

[4] sklearn.feature_selection.mutual_info_classif Available at < https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html>

[5] sklearn. .feature_selection.RFE Available at < https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html>

[6] Tutorial: Pearson's Chi-square Test for Independence Available at <https://www.ling.upenn.edu/~clight/chisquared.htm> (Accessed on: 19 November 2019)

[7] Difference between Mutual Information and F_test, Available at <https://scikit-learn.org/stable/auto_examples/feature_selection/plot_f_test_vs_mi.html>

[8] Chi-Square Test for Independence, Available at <https://stattrek.com/chi-square-test/independence.aspx?Tutorial=AP>