Home Work

Adil Bin Bhutto

November 10, 2020

Contents

1 Exploratory data analysis

1.1 Data Description

Data is imported from the given flat files into pandas data frame. Figure 1 gives an overall idea about the raw data. More on individual features, correlation and missing values regarding which to be discussed can be found here: *https://abbhutto.com/eda01*.

Total **21755** observations with **9** features (namely A1, A2, ..., A9) and one target Class is present in our data. It is clearly mentioned in the comment section of flat files that target Class has total 7 distinct values starting from 1 to 7, but description shows Class has min value 1 and max value 5. To make sure the validity and continuity, uniqueness of target Class is checked and it turns out that it has only 5 distinct values (1, 2, 3, 4 and 5).

	A1	A2	A3	A4	A5	A6	A7	A8	A9	Class
count	19885.000000	19985.000000	19645.000000	19670.000000	19795.000000	19705.000000	19860.000000	19855.000000	19735.000000	21750.000000
mean	48.178024	0.203453	85.626572	1.152211	34.554281	4.123776	37.361027	51.161420	13.789714	1.698851
std	12.528780	35.498976	9.244756	59.042317	22.544475	147.752574	13.645467	23.003543	26.417671	1.351700
min	36.000000	-908.000000	71.000000	-587.000000	-46.000000	-898.000000	-18.000000	-353.000000	-356.000000	1.000000
25%	38.000000	0.000000	78.000000	0.000000	26.000000	-4.000000	31.000000	37.000000	0.000000	1.000000
50%	45.000000	0.000000	83.000000	0.000000	42.000000	0.000000	39.000000	44.000000	2.000000	1.000000
75%	55.000000	0.000000	90.000000	0.000000	46.000000	5.000000	42.000000	62.000000	14.000000	1.000000
max	123.000000	1409.000000	113.000000	2565.000000	436.000000	6339.000000	72.000000	130.000000	126.000000	5.000000

Figure 1: Description of shuttle data after importing

It's clear from the Figure 2, count of each feature and target class is off by some numbers due to the fact that our data contains massive amount of **null** values. How to deal with these null values is discussed in the data imputation section.

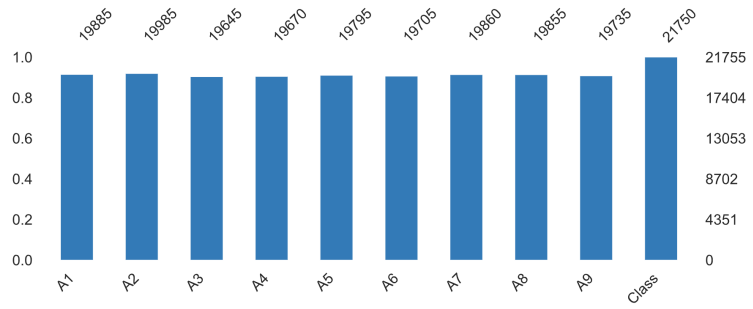


Figure 2: Missing values per feature and target class

Target Class has **5** missing values and all the corresponding observations possess similar null value for each feature. It's shown in the Figure 3. These observations can be dropped directly form the data-set.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	Class
21729	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21732	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21734	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21738	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21745	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 3: Observation with all the null values

Significant amount of duplicate rows are found as shown in the Figure 4. These **10373** rows can be dropped to clean the data.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	Class
1183	37.0	0.0	76.0	0.0	26.0	NaN	40.0	50.0	10.0	1.0
2175	41.0	1.0	86.0	3.0	42.0	6.0	45.0	45.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...
21753	44.0	-4.0	77.0	0.0	44.0	21.0	33.0	33.0	0.0	1.0
21754	54.0	0.0	108.0	5.0	54.0	0.0	54.0	54.0	0.0	1.0

10373 rows × 10 columns

Figure 4: Duplicate rows in the data set

After removing all the duplicate and null value rows, data-set is left with **11381** observations. Figure 5 describes current data. Still a lot of missing data are left to be addressed in all the features. Next step will be to fill these missing cells with some appropriate values.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	Class
count	9883.000000	9997.000000	9736.000000	9768.000000	9820.000000	9829.000000	9900.000000	9892.000000	9786.000000	11381.000000
mean	48.312152	0.308693	85.647186	0.939701	34.512627	4.338997	37.219697	51.249090	13.949928	1.707056
std	12.802052	42.783687	9.255452	53.115921	22.480552	149.193657	13.751379	23.080156	26.450759	1.359834
min	36.000000	-908.000000	71.000000	-587.000000	-46.000000	-898.000000	-18.000000	-353.000000	-356.000000	1.000000
25%	38.000000	0.000000	78.000000	0.000000	26.000000	-4.000000	31.000000	37.000000	0.000000	1.000000
50%	45.000000	0.000000	83.000000	0.000000	42.000000	0.000000	39.000000	44.000000	2.000000	1.000000
75%	55.000000	0.000000	90.000000	0.000000	46.000000	5.000000	42.000000	63.000000	16.000000	1.000000
max	123.000000	1409.000000	113.000000	2565.000000	436.000000	6339.000000	72.000000	130.000000	126.000000	5.000000

Figure 5: Description of the data after removing redundant observations

1.2 Data Imputation

K-Nearest Neighbor Imputation is used to maintain the value and variability of the data-set unlike average values such as mean, median, mode etc. In this method, k neighbors

are chosen based on euclidean distance and their average is used as an imputation estimate. In distance value calculation, missing feature has no role to play. In this case k = 5 is used, which is a default value in Scikit-Learn's [3] (library) KNN Imputer. Also it is safe to say that there is no significant change in the mean and standard deviation of the data-set after the imputation.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	Class
count	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000	11381.000000
mean	48.315210	0.382128	85.548212	0.800018	34.479501	3.965275	37.242281	51.176926	14.058273	1.707056
std	12.800529	44.703967	9.106756	49.657031	22.543932	139.071294	13.618867	22.674259	26.498791	1.359834
min	36.000000	-908.000000	71.000000	-587.000000	-46.000000	-898.000000	-18.000000	-353.000000	-356.000000	1.000000
25%	38.000000	0.000000	78.000000	0.000000	26.000000	-4.000000	31.800000	37.000000	0.000000	1.000000
50%	45.000000	0.000000	83.000000	0.000000	42.000000	0.000000	39.000000	44.000000	2.000000	1.000000
75%	55.000000	0.000000	89.400000	0.000000	46.000000	5.000000	42.000000	62.000000	16.000000	1.000000
max	123.000000	1409.000000	113.000000	2565.000000	436.000000	6339.000000	72.000000	130.000000	126.000000	5.000000

Figure 6: Description of the data after Imputation

1.3 Feature Selection

Linear correlation between two variables is measured using Pearson's correlation coefficient (r) here. It's value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. It's clear from the Equation 1 that r is obtained by dividing co-variance of X and Y with product of their standard deviations.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

After analysing the heat map shown in the Figure 7, it seems:

- **A5** is highly correlated with **A8**.
- **A2**, **A3**, **A4** and **A6** don't have much significant correlation with other features.
- **A1**, **A5**, **A7**, **A8** and **A9** have a good correlation with target Class feature. These features can be very useful for our model. But again out of **A5** and **A8**, only one can be considered due to their high correlation.

To select relevant features, F1 score obtained from a decision tree vs. best k features selected based on **ANOVA** f-test is to be plotted [4] (Shown in Figure 8). ANOVA is an acronym for Analysis of Variance and is a parametric statistical hypothesis test for determining whether the means from two or more samples of data come from the same distribution or not. F-test, is a class of statistical tests that calculate the ratio between variances values, such as the variance from two different samples or the explained and

unexplained variance by a statistical test, like ANOVA. The ANOVA method is a type of F-statistic referred to here as an ANOVA f-test.

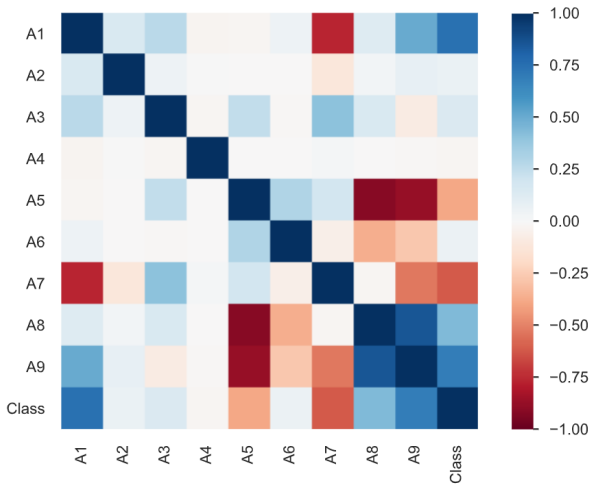
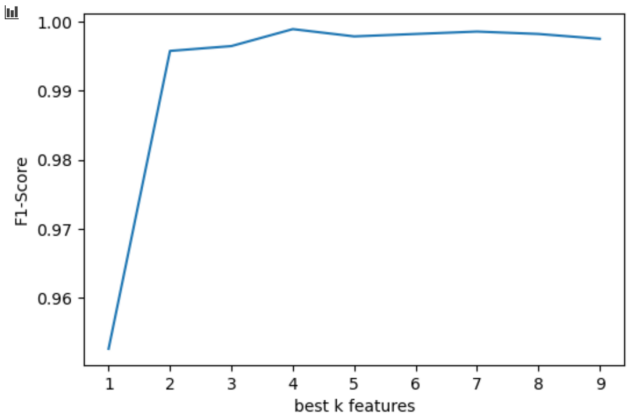


Figure 7: Duplicate rows in the data set

Figure 8 clearly indicates that 4 features namely **A1**, **A7**, **A8** and **A9** can be picked for the final data-set. F1 score increases till 4 features and remains almost constant till selection of all the features. This inside is also similar to the heat-map analysis done on Pearson's correlation coefficient. After dropping irrelevant features, **8104** observations became duplicate and data-set is left with **3277** observations. (**3277 + 8104 = 11381**)




```
Best  1  Feature:  [1]
Best  2  Feature:  [1 9]
Best  3  Feature:  [1 7 9]
Best  4  Feature:  [1 7 8 9]
Best  5  Feature:  [1 5 7 8 9]
Best  6  Feature:  [1 3 5 7 8 9]
Best  7  Feature:  [1 2 3 5 7 8 9]
Best  8  Feature:  [1 2 3 5 6 7 8 9]
Best  9  Feature:  [1 2 3 4 5 6 7 8 9]
```

Figure 8: Duplicate rows in the data set

Detailed profiling of entire data after doing all these pre-processing can be found in the given link with much more visual aids *https://abbhutto.com/eda02*.

2 Choosing a classifier

Labeled data set gives us the liberty to choose a supervised learning model for our classification problem. Figure 9 shows that there are outliers and lot of overlapping between different Classes, which suggests KNN or tree based model should be preferred rather than logistic regression model. Due to the fact that our data-set is not so large (**3277** observations) it's better to choose eXtreme Gradient Boosting classifier [2] (**XGBoost**). XGBoost is an ensemble method which uses multiple decision trees and optimisation of loss function under the hood. XGBoost is a library of gradient boosting algorithms

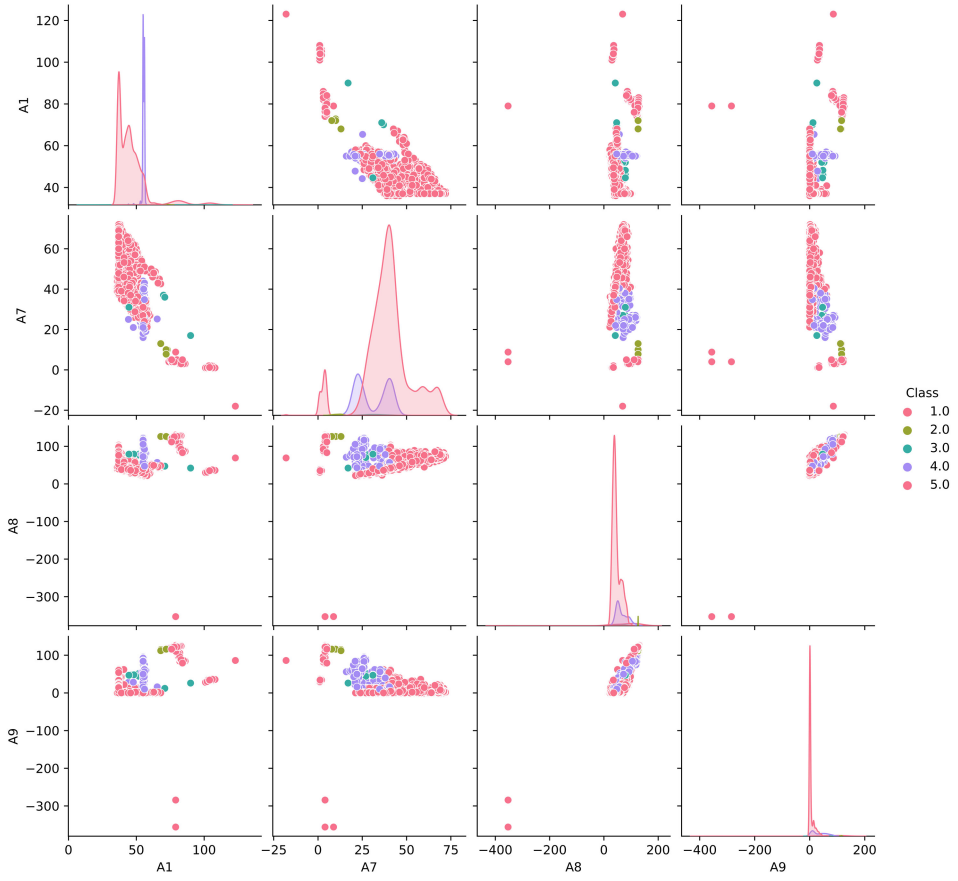


Figure 9: 16 Pair plots with Target Class distinction

optimized for modern data science problems and tools. It leverages the techniques mentioned with boosting and comes wrapped in an easy to use library. Some of the major

benefits of XGBoost are that its highly scalable/parallelizable, quick to execute, and typically out performs other algorithms.

3 Problems encountered

1. Obtained data-set is highly imbalanced with almost 80% observations only from Class **1**. In contrast Class **2** has only 0.2% of total observations. Frequency and count of each target Class is shown in the Figure 10.

Value	Count	Frequency (%)
1	2619	79.9%
4	501	15.3%
5	141	4.3%
3	10	0.3%
2	6	0.2%

Figure 10: Imbalance data set

2. Use of tree based method increases the probability of over fitting.
3. Dividing this relatively small data-set into train, validation and test data is quite challenging.

4 How to address the problems

1. To acknowledge the imbalance in data-set, weight is assigned to each target Class. Lesser the frequency of a target class more weight is assigned to it, which acts as a penalty while calculating loss function by the model. XGBoost is trained to minimize a loss function and the 'gradient' in the gradient boosting refers to the steepness of this loss function. A small gradient means a small error and, in turn, a small change to the model to correct the error. A large error gradient during training in turn results in large correction. Class weight value is used to scale the gradient respective class.
2. Hyper-parameter tuning is used to get the maximum depth of trees. Deeper trees would result in fewer trees being required in the model, and the inverse where simpler trees require many more trees to achieve similar results. The increase complexity provided by deeper individual trees would result in greater over-fitting of the training data. Having more trees also have the same effect. Therefore finding a middle ground where depth and tree number is optimum can solve the problem.

3. If the original data-set is big enough, it can be split into three subsets: training, testing, and validation. The validation set is about the same size as the testing set, and it is used for evaluating the model after training. The testing set is then used for final evaluation once the model is done training and tuning. However, partitioning the original data-set into three distinct sets will cut into the size of the training set. This can reduce the performance of the model if our original data-set is not large enough. A solution to this problem is cross-validation (CV). Cross-validation creates synthetic validation sets by partitioning the training set into multiple smaller subsets. One of the most common algorithms for cross-validation, K-Fold CV, partitions the training set into k approximately equal sized subsets (referred to as folds). There are k "rounds" of the algorithm, and each "round" chooses one of the k subsets for the validation set (a different subset is chosen each round), while the remaining k - 1 subsets are aggregated into the round's training set and used to train the model.

5 Classifier validation

As mentioned earlier K-Fold validation technique [1] is used here. Here value of K is taken as **4** considering size of data-set and Class distribution. Each round of the K-Fold algorithm, the model is trained on that round's training set (the combined training folds) and then evaluated on the single validation fold. The evaluation metric depends on the model. For classification models, this is classification accuracy on the validation set. Figure 11 shows mean and standard test score for different max depth of trees. Here depth of **4** shows better result compared to others (rank = 1).

	params	rank_test_score	mean_test_score	std_test_score
0	{'max_depth': 2}	3	0.995116	0.000003
1	{'max_depth': 3}	2	0.995117	0.001148
2	{'max_depth': 4}	1	0.995524	0.001346

Figure 11: Test scores of cross validation with different parameter values

	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	{'max_depth': 2}	0.995122	0.995114	0.995114	0.995114
1	{'max_depth': 3}	0.993496	0.996743	0.995114	0.995114
2	{'max_depth': 4}	0.993496	0.996743	0.996743	0.995114

Figure 12: Test scores of all the subset with different parameter values

All the cross validation metrics look fair and maximum depth for trees is set to **4** for the final model.

6 Performance of classical validation metric

Data-set is split into two subsets, one is training(75%) and another one is testing(25%) set.

Classification Report of Training Set:

	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	1964
2.0	1.00	1.00	1.00	4
3.0	1.00	0.86	0.92	7
4.0	1.00	1.00	1.00	376
5.0	1.00	1.00	1.00	106
accuracy			1.00	2457
macro avg	1.00	0.97	0.98	2457
weighted avg	1.00	1.00	1.00	2457

```
[1964    0    0    0    0]
[   0    4    0    0    0]
[   1    0    6    0    0]
[   0    0    0  376    0]
[   0    0    0    0  106]
```

Figure 13: Validation Metrics for training set with confusion matrix on the right

Classification Report of Test Set:

	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	655
2.0	1.00	0.50	0.67	2
3.0	1.00	1.00	1.00	3
4.0	0.98	0.99	0.99	125
5.0	0.97	1.00	0.99	35
accuracy			1.00	820
macro avg	0.99	0.90	0.93	820
weighted avg	1.00	1.00	0.99	820

```
[653   0   0   2   0]
[  0   1   0   0   1]
[  0   0   3   0   0]
[  1   0   0 124   0]
[  0   0   0   0  35]
```

Figure 14: Validation Metrics for test set with confusion matrix on the right

It's clear from Figure 13 and Figure 14 that classifier struggles a bit in case of recall due to highly imbalanced nature of the data-set. In turn f1-score (harmonic mean of precision and recall) gets affected. Here precision is just a hoax and it doesn't reflect classifier's performance in it's true essence due to biasness towards Class **1**. Despite all the drawbacks, it's worth mentioning that overall this classifier performs well considering the confusion-matrix and classification report.

References

[1] Adaptilab. *Machine Learning for Software Engineers*. URL: https://www.educative.io/courses/machine-learning-for-software-engineers. accessed: 01.11.2020.

[2] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Neko Yan. *Machine Learning for Software Engineers*. URL: https://www.educative.io/courses/hands-on-machine-learning-with-scikit-learn. accessed: 02.11.2020.