RESEARCHERID
THOMSON REUTERS

[ICRTCET-2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

A METHODICAL APPROACH TO OBTAIN HIGH QUALITY CLASSIFIERS FROM UNPROCESSED DATASETS

Anurag S[*1] & Dr. Minal Moharir[2]

[*1&2]Dept. of Computer Science and Engineering, R.V. College of Engineering, Bangalore – 560059, India

ABSTRACT

Over the last few years, there has been an explosion in the data generated by corporate entities. These entities retain data and analyze it in order to gain valuable insights. More often than not, the information derived from the data propels corporate firms towards formulating important decisions affecting the firm. As a result, data scientists are highly sought after professionals. Fundamentally, data scientists are engaged in processing raw data and compiling useful information out of it. In order to do this, the common practice is to adhere to a structured paradigm, which aid these professionals in producing high quality models which concur with the objectives set by their respective organizations.

In this paper, the various steps involved in this paradigm are realized. The Titanic dataset was used to exemplify the flow of the paradigm and logistic regression was used to solve the classification problem. Towards the end of the paradigm, certain miniscule yet effective operations for enhancing the accuracy of the model were applied, which enhanced the model further. The application of all the steps of the paradigm on the dataset yielded a model with a score of 0.84. The overall objective of the paper is to provide a framework that can be used for producing high quality predictive models for data science applications

Keywords: *Logistic Regression, Data Munging, Feature Engineering, Data Science Lifecycle*.

I. INTRODUCTION

Data Science refers to an emerging area of work concerned with the collection, preparation, analysis, visualization, management, and preservation of large collections of information. The main emphasis of the field is to be able to process a given raw dataset by applying a series of operations on it. Conventionally, the term 'Data Science Lifecycle' is used to refer to the series of operations, a part of which may be applied iteratively in order to obtain high quality datasets. These datasets in turn are used to train predictive models of exceptional quality, so that the best possible model accuracy is realized. However, this is easier said than done. There are a number of factors which influence the complexity involved in pre-processing a dataset. They include overall size of the dataset, number of missing elements in the dataset, outliers present in the data and so on. The purpose of this endeavor is to provide a reliable method which can help tackle these complexities.

In order to obtain the aforementioned high quality datasets, a number of operations are applied in order. This involves data extraction, exploratory data analysis, data munging, feature engineering, visualization, model building and lastly, model optimizations [1]. The purpose of this endeavor is to deal with the complexities faced in pre-processing structured datasets, and enhance the performance metrics of the trained model through the step-wise application of the above operations.

II. OVERVIEW

A. Overview of Data Science Life Cycle

The data science lifecycle is described as a series of operations, one of more of which are applied iteratively (in order), to refine a given raw dataset into its high quality equivalent and build a model that records a high accuracy score. These operations are applied repetitively; as the understanding of data gets clearer, more tweaks can be performed on the model. The expected accuracy of the final model depends on the use-case. The series of steps

applied are not fixed, and varies by application. However, certain operations are elementary, in the sense that almost all data science projects involve them. Figure 1 depicts a graphical representation of the data science lifecycle.

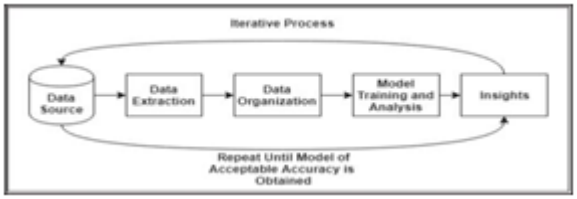


Figure 1. Data Science Lifecycle

The lifecycle process begins with Data Extraction phase. Data extraction is a process of extracting unstructured, semi-structured, and structured data from the user requirement [2]. The data can be obtained from various data sources such as a website like Kaggle or systems that produce data. The term is used to refer to aggregation of raw data, from any data source.

The Data Extraction phase is succeeded by Data Organization. This is the most exhaustive step in the data science lifecycle. It is estimated that around 50% to 80% of the time spent on the data science lifecycle is for accomplishing this step alone [3]. It is the most crucial step in the entire lifecycle, and contributes the most towards the enhancement of the model. Traditionally, this phase of the lifecycle consists of multiple sub-operations. Figure 2 illustrates the common operations applied during Data Organization.

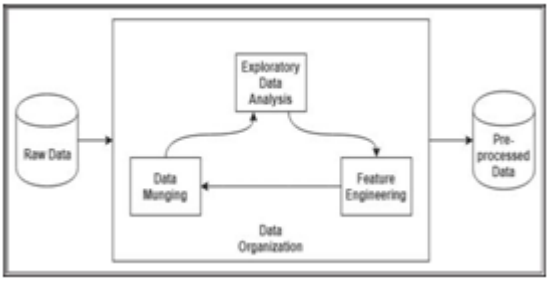


Figure 2. Data Organization

Exploratory Data Analysis (EDA) is the art of exploring data without any clear a priori ideas of what to look for [4]. It includes using observable mathematical entities such as plots and graphs in order to gain high level insights about the data. This step is used to explore the basic structure, and obtain a terse catalogue of the dataset.

Data Munging is the task of refining the dataset, so as to reduce the number of discrepancies in it. The discrepancies in the dataset include missing values, out-of-place values, erroneous values and outliers [5]. It is a very common real world problem, and is a challenging task for data scientists across the globe. There are multiple reasons for the existence of such values. These include system errors, human data entry errors and non-applicability of data for a given real world entity. Data Munging is carried out by deletion and/or imputation. Deletion involves deleting the incorrect data, and replacing it with a default value for the field. Imputation involves replacing missing values with a certain computed value, such as a measure of central tendency. Imputation is the preferred method, since its application usually does not create new outlier values in the dataset. Finally, outlier values need to be considered as well, since they cause a shift in the central tendency measures. Outlier manipulations are performed by imputation, deletion, transformation or binning (grouping). For training models which output accurate results, it is essential to mutate the dataset and homogenize it.

Data Munging is followed by the Feature Engineering process. Feature engineering is the process of taking raw data objects as input and outputting feature vectors suitable for a machine learning algorithm [6]. This step of the data

ISSN 2348 – 8034
Impact Factor- 5.070

organization phase is the single, most important step towards obtaining high accuracy models. To exemplify, consider the classic problem of accommodating categorical character based features in a model. Since most learning algorithms process only numeric data, it is imperative that these characters be converted into numeral forms, which describe the characters in a distinguishable manner. Methods to perform this conversion include binary encoding (for binary fields), label encoding and one-hot encoding. In addition to this, new features which describe the data better may also need to be created. So all-in- all, the Feature Engineering Process is a cut-and-try process. Cutting edge research in the field of data science focuses on creating new tools and methods which help in automating the Feature Engineering process.

After the completion of the Data Organization phase, we obtain a model of high quality, preferably containing no outliers, no missing values, encoded categorical features and new features which add more depth to the dataset. From this point on, the Data Modeling phase begins. At a very high level, this phase produces a high accuracy model from the dataset obtained from the previous phase. Various classifiers such as support vector machines (SVMs), Decision Table, Multilayer Perceptron, Naïve Bayes, Logistic Regression, etc. are used for this purpose [7]. For the purpose of demonstrating the data science life-cycle, a simple algorithm was sufficient. Hence, in this process, logistic regression was used to perform binary classification on the Titanic dataset.

Before applying a classifier to the dataset, one must establish a baseline in order to evaluate if the model which would be trained is useful. For this purpose, a baseline model is trained. This model simply outputs the most common value (mode) for the output feature. Any model that is trained must have accuracy higher than that of the baseline model, otherwise it isn't worth keeping. For the Titanic dataset obtained straight from Kaggle, the baseline model was computed. Keeping this in mind, logistic regression was applied to the pre-processed Titanic dataset. The specifics of the Titanic dataset are discussed in the next sub-section of this chapter.

B. Abbreviations Used

Following abbreviations have been used in further sections:

TN: True Negative Count
FN: False Negative Count
TP: True Positive Count
FP: False Positive Count

C. Model Performance Metrics

Accuracy describes the fraction of correct predictions made among all the predictions made by the model.

Accuracy = Correct Predictions / Total Predictions (1)

Precision describes the fraction of positive predictions which are correct.

Precision = TP / (TP + FP) (2)

- Recall describes the fraction of positive cases among all positive cases

Recall = TP / (TP + FN) (3)

D. Confusion Matrix

Table 1 depicts a confusion matrix. The matrix records the different kind of predictions made by the model under scrutiny in a tabular format. These values are obtained by running a trained model on the test dataset.

TABLE 1. CONFUSION MATRIX

ISSN 2348 – 8034
Impact Factor- 5.070

	Negative Prediction	Positive Prediction
Genuinely Negative	TN	FP
Genuinely Positive	FN	TP

The advantage of recording values in such a format is that the above performance metrics can be evaluated easily by referencing the table.

III. DATASET DESCRIPTION

The Titanic dataset is available on the Kaggle platform. The data has been split into test and train sets by the creators of the dataset. The training set is used to train the classifier, while the test set is used to test the effectiveness of the trained model when it operates on unseen data [8]. The training dataset consists of 891 rows and 12 columns, while the test dataset contains 418 rows and 11 columns (No output feature). Table 2 summarizes the features present in the dataset.

TABLE 2. FEATURE DESCRIPTION OF TITANIC DATASET

Field Name	Description	Key
PassengerId	Index Column	-
Survived	Survival	0/1 = Yes/No
Pclass	Ticket Class	1/2/3 = 1st/2nd/3rd
Name	Passenger Name	-
Sex	Gender	-
Age	Age in Years	-
SibSp	No. of siblings/spouses	-
Parch	No. of parents/children	-
Ticket	Ticket ID	-
Fare	Ticket Fare	-
Cabin	Cabin Number	-

Embarked	Port of Boarding	C/Q/S = Cherbourg/ Queenstown /Southampton

The survival feature is a binary feature which needs to be predicted by the model (output feature). The 11 remaining features are the input features (predictors) for model training. For analysis, the PassengerId column is used as an index for accessing random tuples.

IV. IMPLEMENTATION

This chapter is meant to elaborate on the details to implement the phases and operations of the data science lifecycle. It includes template selection, data organization and the model training phases of the lifecycle. Towards the end, a short coverage on optimizing the trained model is also included.

A. Template Selection

Various files such as config files, source code and other resource files in a data science project need to be organized in order to increase overall productivity. Hence, most data science projects are usually not started from the rock bottom. Rather, a commonly accepted project template is used in order to keep files organized. For our analysis, the cookiecutter data science template was used. Source code was stored in the src directory, while datasets were stored in the data directory. The.env file was used to store authentication details. Jupyter notebooks were updated in the notebooks directory.

B. Data Extraction

The data extraction phase involved importing the required datasets from a data source. In the context of the Titanic dataset, the data source was equivalent to the Kaggle dataset repository. The various options available to extract the datasets were approaches such as simply downloading the Titanic datasets from the Kaggle challenge website, importing tuples from a database, employing APIs in order to import data, web scraping and so on. By trying all alternatives, it was found that the fastest and most efficient way to extract datasets was by utilizing APIs.

The Python requests module was employed in order to extract the Titanic dataset. However, before doing so, a mandatory Kaggle authentication step was to be completed. For this, the dotenv module was used to manipulate the .env file, which stored login credentials in environment variables. Using these variables, a dictionary was created. This dictionary was given as an input to a requests object, which created a session for downloading the datasets through a URL.

C. Exploratory Data Analysis

Once the raw dataset was available, the next step was to check the structure of the dataset and discover its abnormalities. For data analysis, the Python (3.6.3) pandas library (v0.22.0) was used. The pandas module contains a plethora of functions which greatly aid data scientists in carrying out the various operations of the data science life cycle.

First, the extracted raw data was brought into the code by using the csv helper functions that were supplied by pandas. Once brought into program, the train and test datasets were joined to produce a single DataFrame object, for a combined analysis. After this, the df.info() method was used to understand the key properties of the raw dataset, such as the number of tuples and columns, the data-types of each column, number of non-empty tuples and so on. These newly discovered facts about the raw data were analyzed and utilized in further steps of the Data Organization phase to refactor the dataset.

D. Data Munging

After the discrepancies in the data had been exposed, Data Munging operations had to be applied on the raw dataset. This was again achieved by using the pandas module. As an example, consider the Embarked feature of the test Titanic dataset. During the Exploratory Data Analysis step, it was found that 1307 out of 1309 tuples in the dataset contained non-null values. This meant two rows had missing data under the Embarked feature.

In order to fill in the missing data, we used imputation. Imputation may be done on the basis of different measures in the dataset. The median is preferred to impute missing data, since it is a measure of central tendency which does not get affected by outliers. Hence, in our analysis, we computed the median Fare for a grouping of PClass and Embarked features. The median data was compared against the two tuples for which Embarked was null in the dataset. Since both the tuples contained 1st Class and a Fare of 80, the median Fare closest to 80 was picked, and the corresponding Embarked value was imputed (in this case, it was C). Similar analysis was performed for the Age and Fare fields, and missing values were plugged in.

Outliers were also dealt with during this step. The Fare column was one column which contained many extreme values. To deal with them, binning was used. Four bins were created: very_low, low, high and very_high. Using the qcut() method in pandas, these 4 bins were automatically created for the Fare feature.

E. Feature Engineering

After the Data Munging step was completed, we obtained a DataFrame object which represented the Titanic dataset. This representation was free from any missing values. At this stage, the Feature Engineering process was applied to the DataFrame object.

As an example, consider the Parch and SibSP fields of the DF object. Instead of having two fields, it is possible to combine them and create a single new field which describes the overall family size of a passenger. To facilitate this new field creation, pandas is used as:

```
df['FamilySize'] = df.Parch + df.SibSp
+ 1                                    (4)
```

The result of combining the two fields was unknown at the time of applying this operation. However, the data science lifecycle is iterative in nature. Hence, different combinations were tried to try and maximize the accuracy gains from the model through Feature Engineering.

As another example, the Sex field of the DF was taken. It contained one of two values: Male or Female. However, this field needed to be converted into numeric form for the purpose of training a classifier. Hence, a categorical feature encoding operation was performed. The one-hot encoding scheme was followed, and two new fields (isMale and isFemale) were created. Each of these was a binary feature, which was set to 1 if the passenger's gender was the same as that which the feature indicated. As a part of the Feature Engineering process, features were created to replace the remaining character based features as well.

From the Data Munging phase, binned values were available to address outliers. Another feature Fare_Bin was added in order to record the bin each passenger got categorized into. Finally, one-hot encoding was applied to the Fare_Bin feature to convert the labels into binary features, and the Fare_Bin feature was dropped. At the end of the feature engineering step, quite a few derived features were introduced into the DataFrame. The character based features were dropped, since they had all been refactored into other features. Lastly, the Kaggle test and train sets were separated and stored into separate csv files.

F. Model Training

For training the Logistic Regression model, the Python module scikit-learn (v0.18.1) was used [9]. This library provided support for splitting the training data into test and training sets (which is meant for us to check how accurate the model is before running the model over the Kaggle test dataset). In addition, various performance metrics were also inbuilt into this Python library.

To begin with, the training dataset was split into test and train portions in a 20: 80 ratio. The train split was used to train the classifiers, while the test split was used to rate the trained model. A baseline model was constructed using the DummyClassifier module provided by scikit-learn. The most_frequent strategy was applied, so that the baseline model only classified the modal output feature value. Performance metrics were computed for this model for future analysis.

Next, the actual classifier was trained using the LogisticRegression module of the scikit-learn library. Inbuilt performance metrics such as accuracy, precision, recall and the confusion matrix were used in order to analyze the performance of the trained classifier.

G. Enhancement of Trained Classifier

By tweaking the parameters that were passed to the LogisticRegression methods and employing other modules in the library, it was possible to enhance the accuracy of the trained classifier. First, the MinMaxScaler module was used to normalize the features in the dataset. This converted all the values in the each feature of the dataset, such that they lie between 0 and 1.

The normalized DF was used to train the classifier. To obtain the best model for the given data, the regularization parameter in the LogisticRegression constructor was tested for different values. This was implemented using the GridSearchCV module, with the cross validation parameter set to 3, which configures a 3-fold cross validation process during training.

V. RESULTS AND ANALYSIS

This chapter elucidates the results obtained by applying the phases of the data science lifecycle. It covers the changes made to the dataset ensuing the phases, and also looks at the performance metrics for the trained classifier. At the end, the boost in performance by tuning the model is also covered.

A. Data Organization

The Data Organization phase included applying Data Munging and Feature Engineering operations to the imported raw dataset. After the conclusion of this phase, the dataset had 33 columns, as opposed to 12. Table 3 summarizes the changes that were made to the dataset. The convention followed is that F_X is a derived feature which was derived from original feature F. If a single row describes multiple features, it is stated explicitly.

TABLE 3 DATASET STATE POST DATA ORGANIZATION

Feature Name	Before Data Organization	After Data Organization
Age	714 non-null entries	891 non-null entries
Deck_X [8 features]	Cabin feature, with 204 non-null entries	One-hot encoded features for Cabin, each with 891 non-null entries
Embarked_X [3 features]	Embarked feature, with 889 non-null entries	One-hot encoded features for Embarked, each with 891 non-null entries.
PClass_X [3 features]	PClass feature, with 891 non-null featuress	One-hot encoded features for PClass, each with 891 non-

		null entries
IsMother	-	Derived feature, if female passenger has Parch>0 and Age>18
IsMale	-	Derived Feature, for male passengers
Title_X [7 features]	-	Derived feature, obtained by extracting title from Name field
Fare_X [4 features]	-	Derived feature, based on the 4-way binning process applied on the Fare feature
AgeState_X [2 features]	-	Derived feature, based on Age feature. Adult if Age>18, else Child.
FamilySize	-	Derived Feature
Fare	891 non-null entries	Unchanged

Survived	891 non-null entries	Unchanged

The summarized change-log shows that all missing values in the raw dataset have been imputed. In addition to this, several derived features based off the original features were also created for training a better model.

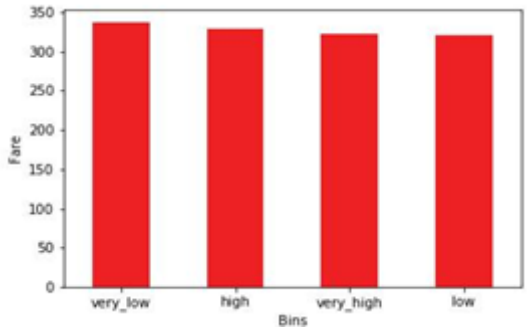
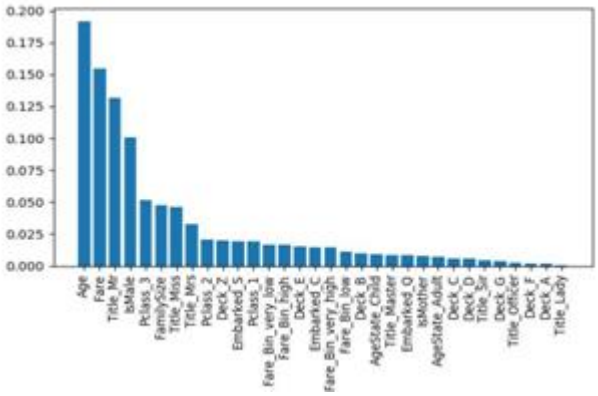


Figure 3. Binning Applied to Fare Column

Figure 3 depicts the re-distribution of number of tuples present in the Fare feature post the quartile binning operation. After this grouping of Fare data, the Fare_X features were created as binary features.

Figure 4 shows the ranking of features by importance, as seen by the ExtraTreesClassifier module of scikit-learn

ISSN 2348 – 8034
Impact Factor- 5.070



According to the bar plots in Figure 4, the Age, Fare and Title_Mr features contributed the most towards predicting the fate of each passenger. This graph would be useful in deciding which features to retain for the next iteration of the lifecycle, so as to get a better, yet less complicated classifier for the Titanic problem. In other words, features like Title_Lady, Deck_A, Deck_F and so on can be dropped without significant consequences.

Performance of Classifier

First, the dataset from the previous phase was split into test and train fragments. Then, a baseline model was created in order to set a bottom benchmark for the model. Finally, the actual model was trained using the LogisticRegression module of scikit-learn. The results are recorded in Table 4.

TABLE 4 PERFORMANCE METRICS FOR MODELS

Performance Metric	Baseline Model	Logistic Regression based Model
Confusion Matrix	[[110 0] [69 0]]	[[95 15] [15 54]]
Accuracy	0.61	0.83
Precision	0.00	0.78
Recall	0.00	0.78

Since the metrics of the trained model were better than that of the baseline model, it was retained for running on unknown data.

C. Model Enhancement

To enhance the dataset further, the feature standardization operation was applied to the dataset. Feature standardization was applied to try and bring the skew-ness of the data close to 0, so that the data is uniformly distributed. The GridSearch module was used to run the training algorithm on the dataset for different regularization constants, and finally return the model with the best score. The accuracy of the best model obtained by applying the operations above was recorded to be 0.84.

Although not a significant increase, this was a welcome result. The importance of tweaking parameters and performing operations like normalization, regularization, standardization and so on isn't felt as much with logistic regression because of the nature of the algorithm. However, it has been shown to have significant impact on other classification algorithms [10].

VI. CONCLUSION

This process demonstrated the various phases in the data science lifecycle. It also discussed about the steps in each phase, and different ways to implement them. The Data Extraction phase is used to extract data from a data source. The Data Organization step which consumes about 80% of the lifecycle time, fixes the shortcomings in the dataset, and adds new features which might help in building a better classifier. The model building phase involves employing a suitable classification algorithm to train the classifier. Finally, an optional model enhancement phase involves tweaking parameters to obtain a higher scoring model. It was concluded that the application of the steps of the paradigm resulted in the creation of a model with an accuracy of 0.84. All-in-all, this paper demonstrated the consistent steps that can be used for any data science project, with a few minor changes at best.

VII. FUTURE WORK

The paradigm can be reapplied to the Titanic dataset, and new better features which result in a higher accuracy model can be created. In addition to this, the paradigm can be applied to check its consistency with other classification algorithms, which are proven to yield better results than logistic regression. Lastly, the paradigm can be applied to other unsolved data science problems, in order to obtain high scoring models.

REFERENCES

1. *Proyag Pal, Triparna Mukherjee and Asoke Nath, "Challenges in Data Science: A Comprehensive Study on Application and Future Trends", International Journal of Advance Research in Computer Science and Management Studies, Volume-3, Issue-8, August 2015 [Online]. Available: . http://www.ijarcsms.com/docs/paper/volume3/issue8/V3I8-0004.pdf*
2. *M. A. B. M. Azir and K. B. Ahmad, "Wrapper approaches for web data extraction : A review," 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), Langkawi, June 2017, pp. 1-6.*
3. *Furche, T, Gottlob, G, Libkin, L, Orsi, G & Paton, NW 2016, Data Wrangling for Big Data: Challenges and Opportunities. in Advances in Database Technology — EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology. Advances in Database Technology, July 10–12, 2017, pp. 473-478*
4. *S. Malinchik, B. Orme, J. A. Rothermich and E. Bonabeau, "Interactive exploratory data analysis", Proceedings of the 2004 Congress on Evolutionary Computation,July 19-23, 2004, pp. 1098-1104, Vol.1.*
5. *Sean Kandel, Jeffrey Heer et al., "Research directions in data wrangling: Visualizations and transformations for usable and credible data", Sage Journals, Volume: 10, issue: 4, page(s): 271-288 October 2011*
6. *M. R. Anderson and M. Cafarella, "Input selection for fast feature engineering," 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, 16-20 May, 2016, pp. 577-588.*
7. *Himani Raina, Omais Shafi, "Analysis Of Supervised Classification Algorithms", 2015 International Journal of Scientific and Technology Research Vol. 4, Issue 09, September 2015, ISSN 2277-8616*
8. *Titanic: Machine Learning from Disaaster (Kaggle Challenge) [Online]. Available: https://www.kaggle.com/c/titanic/data*
9. *Scikit-learn: Machine Learning in Python, Pedregosa et al., Journal of Machine Learning Research 12, pp. 2825-2830,October 2011.*
10. *Andrew Craig,Olivier Cloarec,Elaine Holmes,Jeremy K. Nicholson, and, and John C. Lindon, "Scaling and Normalization Effects in NMR Spectroscopic Metabonomic Data Sets", April 2006, 78 (7): 2262-2267.*