Kurdistan Journal of Applied Research (KJAR) | Print-ISSN: 2411-7684 – Electronic-ISSN: 2411-7706 | kjar.spu.edu.iq
Volume 2 | Issue 3 | August 2017 | DOI: 10.24017/science.2017.3.18

Successful Data Science Projects: Lessons Learned from Kaggle Competition

Mohammed Zuhair Al-Taie
Faculty of Computing
Universiti Teknologi Malaysia
Johor, Malaysia
mza004@live.aul.edu.lb

Naomie Salim
Faculty of Computing
Universiti Teknologi Malaysia
Johor, Malaysia
naomie@utm.my

Adekunle Isiaka Obasa
Department of Computer Science College of Science and Technology Kaduna Polytechnic
Kaduna, Nigeria
iaobasa2@gmail.com

Abstract: ***The workflow from data understanding to deployment of an analytical model of a data science project begins at framing the problem at hand, a task that is typically business-oriented and requires human-to-human interaction. However, the next three steps: data understanding, feature extraction, and model building that come next in the pipeline are the key to successful data science projects. Failing to fully understand the requirements of each of these three steps can negatively affect the performance of the proposed system. Hence, the current study tries to answer the following question "What are the requirements of a successful data science project?" To answer this question, we will use the solution that we built to measure the relevance of local search results of small online e-businesses and submitted to Kaggle data science platform to shed light on why our solution did not achieve a top position among other competitors. Evaluation of the design that we submitted to the competition is going to be carried out in the spirit of the three winning submissions. Our results revealed that well-performed data preprocessing, well-defined features, and model ensembling are critical for building successful data science projects. Such a clarification provides insight into specific aspects of model design to help others including Kagglers avoid possible mistakes while approaching their data science projects.***

Keywords: Data Science Pipeline, E-businesses, Kaggle Competition, Model Ensembling, Relevance Prediction.

1. INTRODUCTION

Data science (DS) is the art and science of acquiring knowledge through data [1]. In other words, it is about how to obtain data, and use it to extract knowledge and gather insights that we can use to make informed decisions and predictions. As we will see in more detail in the next section, DS pipeline involves several steps: Frame the problem, understand the data, extract features, modeling, and analysis, present results, and finally deploy code.

The large volume of data that is generated every day in recent years, the speed at which data is generated, and heterogeneity of data (the three characteristics of Big Data) can explain why people are interested today in DS more than before. For businesses, DS is able to provide the data information that empowers organizational processes for the sake of optimized efficiency and revenue generation. DS methods are also able to derive results and develop protocols to deal with different scientific goals.

Many programming languages are available today for data scientists to implement their projects. For instance, Python is a general-purpose programming language that is becoming more and more popular to do DS. R is also very popular among data scientists. It was intended to serve statisticians more through the graphics capabilities and large statistical functions that the language is augmented with. Scala is particularly well-suited for data scientists (or data engineers) who are willing to work with Apache Spark and to tackle Big Data applications with ease. MATLAB has achieved in recent years a widespread acceptance among engineers, mathematicians, and scientists. However, it is not well-suited for large software frameworks or string-based data wrangling but rather for numerical computations.

Founded in 2010, Kaggle is a type of crowdsourcing knowledge service that uses different kinds of human knowledge to solve complex and cognitive tasks. It was established as a platform for predictive modeling and analytics competitions. Companies and researchers can post their data while statisticians and data miners can experiment their approaches and compete against each other to achieve the highest competition score. Submissions of participants are scored based on how successful they were meeting the requirements of the given competition. At the end of the competition, hosts give the winners compensation, which can be money, knowledge, or job vacancies in one of the leading business analytics companies like Google, Microsoft, or Cloudera.

The goal of this study is to understand the requirements of a successful data science project by providing an evaluation of the model design that we submitted to Kaggle two years ago to tackle the problem of measuring the relevance of search results, given search query and the corresponding response. The evaluation is going to be addressed in the light of the solutions that came in the first three positions. Upon conducting our evaluation, we will consider only the three most critical stages of DS projects, i.e., data preprocessing, feature engineering, and model building. Such an evaluation can help data scientists (including Kagglers) identify and manage potential issues in their model designs.

The rest of the paper is organized as follows: Section 2 presents the different steps that are involved in a typical data science project. Section 3 discusses the focus of this study, which is Kaggle competition. The competition

aimed to measure the relevance of search engine results. Section 4 describes the solution that we submitted to Kaggle competition some time ago to solve the problem above. Section 5 gives an overview of the three reference models that we are going to compare our solution with. Section 6 provides the results of the study, Section 7 discusses these results, and Section 8 concludes the study.

2. LITERATURE REVIEW

We need to give a high-level overview of the processes of DS by highlighting the different phases that are involved in any DS problem. DS pipeline consists of a number of steps [2] as shown in Figure 1 below. Following these steps means that the data scientist has a clear research plan, a good understanding of the project's aims, and clear deliverables. Such a structured plan can increase the project's success ratio [3].

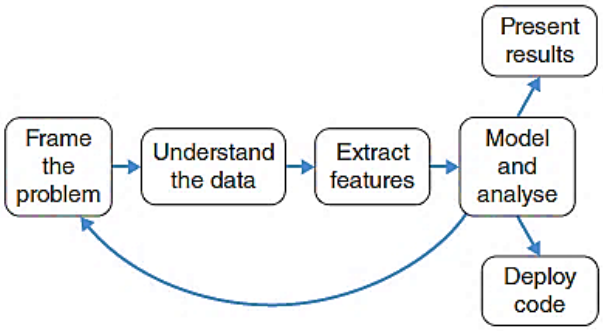

Figure 1 Data science pipeline [2]

1. **Framing the problem** means understanding the context in which the problem occurs. The data scientist in this phase should answer questions that are related to the project like what is the primary goal of the project, what is the benefit of the project, what data and resources are needed, how long is the project going to take, is it part of a bigger strategic plan or one small project, and what are the deliverables [3]. Answering these questions would allow stakeholders to understand *what*, *how*, and *why* an individual project should be undertaken. It would tell everyone else what to do and what is the best course of action.
2. **Understanding the data** has to deal with three tasks. First, the data scientist should be able to answer some preliminary questions about the data before going into analyzing it like is it readily available or should be provided by a third party, how big is it, is it sufficient to represent all the population, does it have missing values, noise, outliers or other inconsistencies. Second, the data scientist needs to prepare the data and get it into a standard format for use in subsequent steps. This task has three requirements [4]: (1) *data cleaning*, which aims to remove noise, fill missing fields, and correct inconsistencies. (2) *Data integration*, which seeks to combine data from varied and different sources into coherent data storage. (3) *Data transformation* helps to convert the data into a usable and understandable form. Third, and to gain a deeper understanding of the data, the data scientist needs to do Exploratory Data Analysis (EDA). This task includes trying different methods to analyze the data to understand how variables (i.e., features) are interacting with each other, the distribution of data, and whether there are outliers or not. Data visualization is also an important part of data analysis. Scatterplots, histograms, line graphs, bar charts, and box plots can be helpful for discovering existing patterns, correlations, and deviations [5].
3. **Extracting features** from raw data is among dimensionality reduction techniques. It aims at finding the most compact and informative features for a given problem [6]. This task can be decomposed into two steps: (1) Feature construction, which helps to convert raw data into a set of useful features and can be considered a preprocessing transformation step. (2) Feature selection which aims at reducing the initial set of features to those that retain enough information for obtaining good results. However, which features will end up being used for the project is an open question in DS and machine learning.
4. **Modeling** (also referred to as model building) is the next phase after features have been extracted. In this phase, the data scientist uses models, domain knowledge, and insights learned from the previous steps to answer the research questions. The techniques used here are borrowed from machine learning, data mining or statistics. A machine learning algorithm can be run, for example, to measure customer loyalty, predict stock prices, or segment customers into different categories based on certain criterion. Typically, building a model is an iterative process that involves selecting variables, executing the model, and model diagnostics to see if the results are achieving the project's goals and satisfying the customer's needs or not.
5. **Presenting the project's results** to the customer is the next step after modeling. This step includes describing the work that has been done and the results that have been achieved. This can be done in many ways ranging from presentations to research reports.
6. **Deploying the project's code** (using GitHub or personal websites for example) would allow it to be run by other people in the future. Code deployment typically includes preparing some documents that show how the code works, and some way to test that the code operates successfully.

The previous description of DS pipeline gives a wrong intuition that these processes are linear but in fact, as shown in Figure 1, this process is highly iterative, and it is common to go back and redo some steps [7]. It is also

worth noting here that this set of processes is intended for a DS project with a limited number of modules. A project with millions of real-time processes would need a different approach that lies out of the scope of this study.

In the next section, we will take a look at the details of Kaggle's competition for finding the relevance of search results, and why this type of competition is important for small online businesses.

3. KAGGLE COMPETITION

Kaggle's competition *Crowdflower Search Results Relevance*, the focus of this study, ran for around two months in 2015 (started on June 29, 2015, and closed on July 6, 2015). The goal was to create an open-source model that can automatically measure the relevance of search results of an e-commerce site. Such type of competition will offer small business owners a model to test against and to match the experience provided by more resource-rich companies. Prizes of the competition were as follows: 1st place - $10,000, 2nd place – $6,000, and 3rd place - $4,000.

The dataset used for the competition was provided by CrowdFlower, a data mining, data enriching and crowdsourcing company. CrowdFlower has had their crowd evaluate searches from a handful of e-commerce websites (Figure 2). Given an example query like 'tennis shoes' and an ensuing result ('Adidas running shoes'), the goal was to score the result on relevance, from 1 (least relevant) to 4 (most relevant). CrowdFlower generated 261 search terms for this purpose, and the list of products and their corresponding search terms were put together.

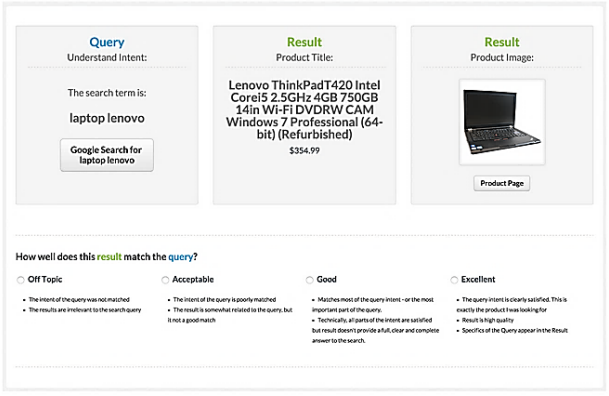

Figure 2 Data collection of Kaggle competition

The anonymized dataset was medium sized and consisted of six attributes: id, query text, product title, product description, median relevance (target variable) and relevance variance of the query. Both the training data and test data are available from CrowdFlower. The data provided by CrowdFlower team contains about 10k samples (i.e., training data with target responses) labeled manually by CrowdFlower using the classes above, and about 20k samples with unknown labels (i.e., test data with no target responses). Solutions were evaluated by using *quadratic weighted kappa* metric. However, two difficulties were present to contestants during the competition: (1) small amount of training data, and (2) non-standard evaluation metric.

3.1. Relevance of Search Results

Each online retailer should have its search engine tailored to the store. These information retrieval mechanisms are typically connected to a database to search for details of a particular product [8]. Hence, the search facility has always been important for website navigation [9] because they enable customers to find the item that they are looking for both easily and efficiently and without physically visiting multiple locations.

For small online businesses, measuring the relevance of search results can help understand whether the search facilities that they are using are efficient enough to provide the information that a customer needs [10].

For e-businesses, evaluating the capacity of the search engine is critical because users were found not willing to invest more time or more effort to improve their searching strategies [11]. They often settle on using simple keywords for searching and viewing only the first few pages of results by not going beyond 20 results. Too many irrelevant results returned by search engines annoy users, cause information overflow and can increase user dissatisfaction [12].

However, to satisfy user requirements, the information returned by the search engine as a result of the user query must be both objective and subjective. It is only the user who issued the query can decide whether the query result satisfies her need or not, which entails that the user should have the ability and knowledge to determine whether the retrieved item is relevant or not [13].

The next section will give an overview of the different components of the solution that we submitted to Kaggle two years ago to tackle this task.

4. CASE STUDY – PREDICTING THE RELEVANCE OF SEARCH RESULTS

The case study (CS) that we are addressing in this study is an open-source model that we built to estimate the relevance of search results, given search query and corresponding response [14]. This case study can be used as an example to show how a DS project may not give the expected results if the requirements of the three main phases (Figure 3): data preprocessing, feature extraction, and modeling are not fully met.

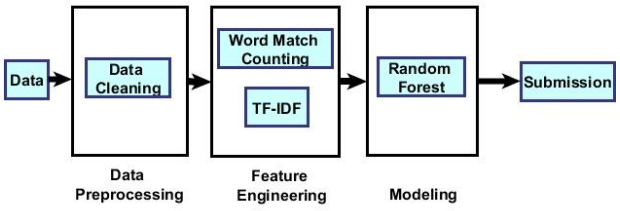

Figure 3 Block diagram of CS [14]

4.1. Data Preprocessing

Because the data is raw and contains information that is irrelevant to the product, we had to remove undesired content. The aim was to improve the quality of data and make it acceptable for mining and analysis.

First, we extracted word tokens from the text by splitting text using spaces. Second, we removed non-alphanumeric characters from text. Irrelevant words, such as stop words, articles, prepositions and pronouns, were also ignored.

4.2. Feature Extraction

The data that we have has just some raw tuples of text queries and product descriptions. Therefore, we had to preprocess it to extract some value, and then transform it to provide regular input data for machine learning algorithms. For this sake, the first task we had to execute is feature extraction. This was done by transforming the raw text search attributes into valuable features for running over machine learning algorithms. We describe, below, how we used two methods for extracting important features from the dataset.

1. **Word Match Counting:** In this step, we wanted to find how many words in each text search that match the product title and product description. This will help to extract two numerical features from the data. For this purpose, we used two Python data manipulation packages: NumPy [15] and SciPy [16]. Using the preprocessed data, we transformed each list of words (from search text attributes) to an array and then used NumPy methods to intersect the search word vector with product title and product description vectors, respectively. This means that we merely counted how many words are in each intersection.
2. **TF-IDF:** The importance of TF-IDF is that instead of just using numeric values to represent the number of word intersections, we can also take into account word weights. For this, we used the TfidfVectorizer class from another Python package, which is scikit-learn. We can provide the text as input and then use this class to calculate weights. The new features that we extracted are much more expressive now because they take into account term weights. Large weight values refer to rare terms while small weight values refer to highly occurring terms across all documents.

4.3. Modeling

After producing the two most important features, as described in the previous subsection, we are now ready to apply machine-learning algorithms. Because the primary goal of the CrowdFlower data is to predict the relevance of search queries, the label attribute for this study will be the median relevance of the search. To this end, we applied two methods:

1. We first started by predicting the values of the median relevance from the test set data by using Support Vector Machine (SVM) [17]. SVM is a supervised machine learning method that uses the concept of decision hyper plane to define decision boundaries. The algorithm takes labeled training data and outputs an optimal hyper plane categorizing new (text) examples.
2. After that, we used a more sophisticated algorithm, which is the Random Forests, an ensemble largely-used machine learning method developed by Breiman and Cutler [18]. Random Forests combine bootstrap aggregating and random selection of features to build a set of decision trees with controlled variance. In other words, each decision tree is constructed by using a random subset of the training data. In this way, a random forest fits a set of decision tree classifiers on multiple sub-samples of the data and then uses averaging to control overfitting and enhance accuracy.

Applying both SVM and Random Forests is a straightforward process in Python using scikit-learn as both algorithms share a common named functions for training. Using the training data from CrowdFlower that we now have as a Python dataframe, and the features that we already extracted, we will apply three simple steps for algorithm learning: initializing the model, fitting it to the training data, and predicting the new values. Initializing and fitting the train data will allow us to predict the label attributes (search terms median relevance).

4.4. Learning Models Benchmark

We applied four types of method combination: SVM with word match counting based features, SVM with TF-IDF based features, Random Forests with word match counting based features and, Random Forests with TF-IDF based features. The goal was to predict the median relevance values (label attributes). The results have shown that Random Forests with TF-IDF achieved the best results (highest score), which is 0.59211. Other results are as follows. Random forests with match counting: 0.53834, SVM with TF-IDF: 0.57654 and SVM with match counting: 0.51241.

As we can see, while CS model followed the main points of a typical DS project, it missed out some important details. For example, data preprocessing was limited to data cleaning and ignored some other critical tasks such as word correction/replacement, stemming, lemmatization, and dimensionality reduction. It used only two types of features: word count and word weight, for training and testing, and single modeling (based on Random Forests) to train the model and give results.

The next section will provide an overview of the different components included in the three submissions that occupied the top positions in the competition.

5. REFERENCE MODELS

In this section, we are going to address the models that took the first three positions in Kaggle 'Search Results Relevance' competition (Figure 4). The first winner (RF1) was Chenglong Chen, a Ph.D. graduate from Guangzhou, Guangdong, China. The second winner (RF2) was a team formed by Mikhail Trofimov, Stanislav Semenov, and Dmitry Altukhov. The third winner (RF3) was 'Quartet' team, formed by Maher Harb, Roman Khomenko, Sergio Gamez, and Alejandro Simkievich. They were from Canada, Ukraine, Spain, and Brazil, respectively. Maher Harb is a Physicist and data scientist, Roman Khomenko is a senior software developer and security researcher, Sergio Gámez is computer vision researcher, and Alejandro Simkievich is a technology entrepreneur and CEO at Statec.

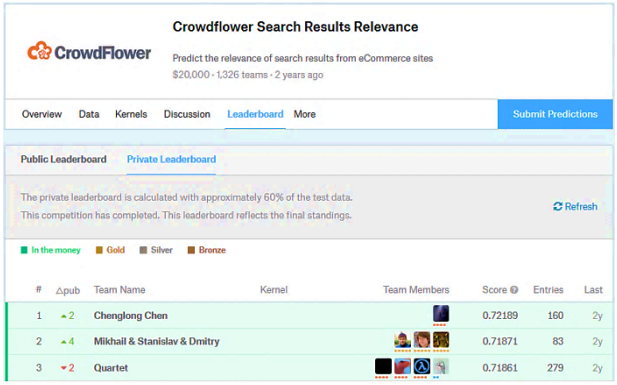

Figure 4 First three winners of Kaggle competition

5.1. Reference Model One

This solution [19], which won the first prize of the competition, consisted of three major parts: preprocessing, feature extraction/selection, and modeling techniques and training. The details of RF1 are shown in Figure 5.

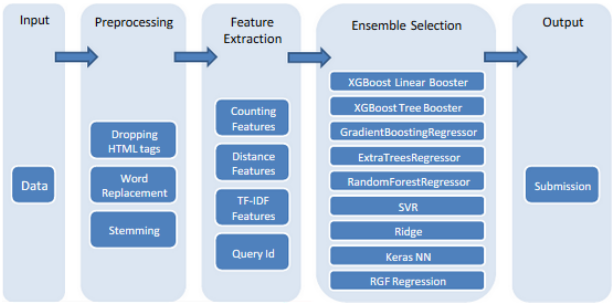

Figure 5 Flowchart of RF1

5.1.1. Preprocessing

For preprocessing, the design used some steps to clean up the text as follows:

1. **Dropping HTML tags**: the author used bs4 library to clean up HTML tags in the production description field of the data.
2. **Word Replacement**: the authors performed word replacement/assignments for example spelling correction and synonym replacement to align those words with the same or similar meaning. For example, refrigirator → refrigerator and bicycle, bike → bike. In addition to these two types of word replacement, the author also applied other replacements for better data processing such as hard disk → hard drive and mac book → macbook.
3. **Stemming:** the author also used stemming before generating features like counting features and BOW/TF-IDF features. For this sake, he used Porter stemmer and Snowball stemmer from NLTK.

5.1.2. Feature Extraction/Selection

For this purpose, the contestant has developed three major types of features: counting features, distance features, and TF-IDF features.

1. **Counting Features:** for this kind of features, they used some features including (1) Basic Count Features (i.e. the count of n-gram, count & ratio of digit, count & ratio of unique n-gram, and description missing indicators). (2) Intersect Counting Features (i.e. count & ratio of *a's n*-gram in *b's n*–gram). (3) Intersect Position Features (i.e. statistics of positions of *a's n*-gram in *b's n*-gram, and statistics of normalized positions of *a's n*-gram in *b's n*–gram).
2. **Distance Features:** using Jaccard coefficient and Dice distance, the contestant computed two types of features: Basic Distance Features and Statistical Distance Features.
3. **TF-IDF Based Features:** the contestant has extracted various TF-IDF features (i.e. Basic TF-IDF Features and Co-occurrence TF-IDF Features) and the corresponding dimensionality reduction version via SVD (i.e., LSA). He also computed the (basic) cosine similarity and statistical cosine similarity.
4. **Other Features:** this included Query Id (one-hot encoding of the query (generated via genFeat id feat.py).

Feature selection followed feature extraction. Feature selection can be helpful in identifying some possible well-performed feature set to train the model with and thus reducing the computation burden. The contestant adopted the idea of "untuned modeling" used by Marios Michailidis and Gert Jacobusse to solve Microsoft Malware Classification Challenge on Kaggle. For features of high dimension (e.g. feature set including raw TF-IDF features), he used XGBoost with linear booster (MSE objective). For features of low dimension, he used ExtraTreesRegressor from scikit-learn. Using ensemble selection, it is possible to train a model library with various feature sets (as shown next) and to pick out the best ensemble within the model library.

5.1.3. Modeling

This solution used ensembling for model building. Ensembling (or ensemble learning) is the process of combining more than one predictive model to produce a new model that is expected be more accurate than any other individual model. The solution consisted of two main steps: (1) training the model library using different models, various parameter settings, and different subsets of features. (2) Generating an ensemble submission from the model library predictions using bagged ensemble selection. For this phase the following techniques were utilized:

1. **Cross Validation Methodology**. The performance was estimated using cross-validation within the training set.
2. **Model Objective and Decoding Method**. In this competition, submissions are scored based on the *quadratic weighted kappa*, which measures the agreement between two ratings. This metric typically varies from 0 (random agreement between raters) to 1 (complete agreement between raters).
3. **Sample Weighting.** The variance of the relevance scores, included in the data, was used as a measure of the confidence of the ratings and to weight each sample.
4. **Ensemble Selection**. For a supervised learning method, the contestant used ensemble selection to generate an ensemble from a model library. Bagged ensemble selection [20] was used for ensemble selection. Ensemble selection is able to build an ensemble that is optimized to an arbitrary metric (such as quadratic weighted kappa used in this competition). In addition, the contestant has applied a number of modifications to the original algorithm. (1) The model library is built with parameters of each model guided by a parameter searching algorithm. (2) Model weight optimization is allowed in the procedure of ensemble selection. (3) Random weight was used for ensembling model similar to ExtraTreesRegressor.

Without any stacking or ensembling, the best (Public LB) single model they have obtained during the competition was an XGBoost model with linear booster. It is with Public LB score: 0.69322 and Private LB score: 0.70768. Apart from the counting features and distance features, it used raw basic TF-IDF and raw co-occurrence TF-IDF.

This solution is based on the use of Python version 2.7.8 and some dependencies including numpy, scipy, scikit-learn, pandas, NLTK, bs4, hyperpot, keras, XGBoost, and ml_metircs. In addition to Python packages, the solution also used rgf and libfm. In particular, Pandas was helpful for feature engineering, NumPy for data manipulation, TfidfVectorizer and SVD from Sklearn for extracting text features, and XGBoost, Sklearn, keras and rgf for model training.

5.2. Reference Model Two

The second solution [21], which came in position #2 of the competition, was based on three key ideas: query expansion, model stacking, and custom class separator. Their solution consisted of the following steps: text preprocessing, query expansion, feature extraction, model training and ensembling, and optimizing class separators.

5.2.1. Text Preprocessing

For text preprocessing, the team applied first some word correction/replacement in both query and title such as hardisk → hard drive, and extenal → external. Stemming was applied next to all queries and titles using nltk package. Finally, lemmatization was applied to the query, title and description using nltk.stem.wordnet.WordNetLemmatizer().

5.2.2. Query Expansion

The team used query expansion techniques to deal with the issue of short search queries. To this end, information from related titles was used to make queries longer, as follows: (1) each query was concatenated to all respective product titles having label = 4. (2) Top *n* (10 to 15) most frequent words were extracted from the product title in descending order. Top *n* words are called "expanded query."

5.2.3. Feature Extraction

Five groups of features were extracted for this competition:

1. **Group 1**: built using some features such as the *number of words in query*, the *number of words in title*, etc.
2. **Group 2**: built using expanded query based on top 10 words. Example features include *number of words from expanded query that are present in title*, *compression distance between expanded query and title*, etc.
3. **Group 3**: same as Group 2 but based on top 15 words.
4. **Group 5**: this group used letter frequencies in query and title using the following ratios as features: *(FreqQuery + A) / (FreqTitle + B) / (query length)*, where *A* and *B* are constants that maximize local score (obtained from cross-validation).
5. **Group 5**: used features such as *word2vec similarities between title and query*, and *vector between mean of words in title and words in query* (using word2vec pertained embedding).

5.2.4. Model Training and Ensembling

This solution used average predictions of four different models. Each model used a different set of features.

1. **Support Vector Regressor 1**: This model used the following feature groups: Group 1, Group 2, Group 4, and Group 5. The team concatenated expanded query (on top 10 words) and title, used TF-IDF transformations on 'char' n-grams from 1 to 5, and selected 300 components of SVD decomposition. All features were scaled to [0, 1] range and the model was trained with 1/(1+variance) weights.
2. **Support Vector Regressor 2**: this model using the following feature groups: Group 1, Group3, Group 4, and Group 5. The team concatenated expanded query (on top 15 words) and title, used TF-IDF transformations on 'char' n-grams from 1 to 5, and selected 400 components of SVD decomposition. All features were scaled to [0, 1] range and the model was trained with 1/(1+variance) weights.
3. **Stacked model 1**: This model was built on original (not expanded) queries. This model was built based on some small per-query models that do not use the whole dataset for training. They were built only on BoW extracted from train+test union.
4. **Stacked model 2**: This model is the same as Stacked model 1 except that this one is built on expanded queries based on top 10 words.

5.2.5. Optimizing Class Separators

The most important part is the transition from raw outputs to class labels. Straightforward rounding from predictions to integers gives very poor results. Therefore, and for each cross-validation iteration, optimal class borders were found via the following routine: scale model outputs to [0, 1] range. Perform exhaustive search over all reasonable borders with 0.01 step size. Custom class separators were found very valuable as they allowed the team to take advantage of evaluation metric.
All single models scores were in the range [0.703, 0.707] depending on random see and appropriate class separators. Models {1, 2} and {3, 4} produced very similar results. Therefore, it is enough to ensemble Model 1 and Model 4 to get a decent score.
The code was run on Ubuntu machine and the following software: python 2.7.6, numpy, pandas, scikit-learn, scipy, nltk, BeautifulSoup, tsne, gensim, and backports.lzma.

5.3. Reference Model Three

The third solution [22], which came in position #3 of the competition, consisted of the following steps: text preprocessing, dimensionality reduction, feature engineering, modeling, rounding results, and ensembling individual models. The details of RF3 are shown in Figure 6.

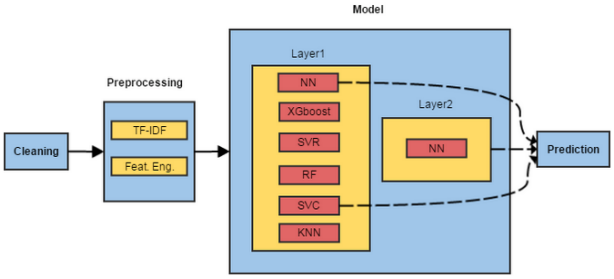

Figure 6 Block diagram of RF3

5.3.1. Data Preprocessing

Because the original dataset was raw text, a number of tasks were performed to convert the raw data to one on which machine learning algorithms can be trained. Data preprocessing included:

1. Removing html tags and other non-text elements. These elements were present since, presumably, the datasets were generated programmatically by parsing the web.
2. Removing stop words such as common articles, prepositions, etc. (e.g. ‘the,' ‘a,' ‘with,' ’on’).
3. Using word-stemmers so that similar terms such as ‘root’ and ‘roots,' or ‘game’ and ‘gaming’ are converted to the same base word.
4. Generating basic features were by applying the TF-IDF operation on the resulting text. The basic functionality of TF-IDF algorithm is to assign a value (between 0 and 1) to every word in a given document, which in turn is part of an existing corpus. The more frequent a term is in the document and less common in the corpus, the higher the value for such term.

5.3.2. Dimensionality Reduction

Because TF-IDF creates a feature for every token in the corpus, dimensionality reduction was applied to the result of data preprocessing to work with 225-250 features instead of thousands of features. For this purpose, truncated singular value decomposition (SVD) was used by the team.

5.3.3. Feature Engineering

The following sets of features were used during the training and testing of the models:

1. **N-gram similarity features**: 14 features were generated through the evaluation of similarity between unigrams, bigrams, and trigrams extracted from the query and the product title & description strings. Some of those features are boolean while others are similarity scores between the n-gram and the search string.
2. **Query-product name similarity features:** A tailored algorithm was developed by the team to extract the main noun from the product title and then performed a similarity measure between the query and the extracted noun. This was done with the help of a set of rule-based regular expressions that were carefully crafted to remove all

unnecessary descriptions, prefixes, suffixes, etc. For example any phrase following the words "for," "by," and "with" was removed; sizes & colors were removed, and so on. Finally, the last two words in the cleaned title were regarded as the main product.

3. **Alternative query similarity features:** For each of the 261 unique queries, the team created a corpus by combining all titles associated with that query, extracted the top trigram, and used it as an alternative query. The idea behind this approach was to capture the correct product implicated when the query did not have similarity with the product title, which could be an issue when the search query is a brand name.
4. **Intra-title similarity features:** The idea behind this set of features was to use the group of titles with the same relevance and within the same query as a reference point and measure similarity of all individual titles within that query to that benchmark. For example, if a particular query has 40 labeled titles (10 per relevance class) then various similarity measures between each title (both labeled and unlabeled) and the four groups are constructed.
5. **Antonym feature:** Because some queries performed particularly bad when compared to hand-labeled predictions in some particular cases, the team developed some additional rules to address this issue. One such case was when an antonym of a noun in the query was in the product title or description (e.g., query: "*men shoes*," product title: "*beautiful women shoes*"). A boolean variable indicating whether an antonym is present (1) or not (0) was created. This was done only for the most common antonyms.

5.3.4. Modeling

A variety of models were used for supervised learning including SVM, ANN, gradient boosted regressors, kNN, and random forests. For some of the models, both the classification and regression versions (e.g., SVM classifiers and SVM regressors) were created. The solution also used alternative implementations of the same model (e.g. using both XGBOOST package and H2O to build gradient boosted regressors). The reason for having a wealth of different models is to use them in ensembling and building of second-layer models.

5.3.5. Rounding Regression Results

This phase included rounding the regression predictions to integers using specially developed thresholds that optimize the kappa score. The team used experiments on the cross-validation folds to determine the optimal rounding limits for rounding predictions obtained through regression.

5.3.6. Ensembling

This phase included ensembling individual models and building second-layer models. The winning model was an ensemble of single-layer ANN, a single-layer SVM, and a second-layer ANN (with features derived from over 20 single-layer models). The details of this two-step ensemble model are as follows:

1. **Step 1:** Ensemble of first-layer ANN (30% weight) and second-layer ANN (70% weight). The averaging is applied to the unrounded predictions, and the optimal rounding thresholds were applied after averaging.
2. **Step 2:** Ensemble of the predictions obtained from step 1 and SVM classifier, according to the following expression: floor (0.8*step1 + 0.35*SVM).

SVM was built using scikit learn (hyperlink), artificial neural networks were built using keras (which in turn outsources some tasks to theano) on Python, while GBM models were built using the XGBOOST and H2O R packages.

We can see that the three reference models discussed above show a strong commitment to the major critical tasks included in the DS pipeline. Various data preprocessing steps (e.g., cleaning, word replacement/correction, stemming, lemmatization, etc.) were applied. Four types of features (i.e., matching, similarity, meta and attribute features) were extracted and model ensembling for the model building was adopted.

The next section will show the differences between the four models regarding the three major phases of a typical DS project: data preprocessing, feature engineering and model building.

6. RESULTS

The study has addressed the different phases involved in the building of the case study model (i.e. CS) as well as the three reference models (i.e., RF1, RF2, and RF3). The three main steps (i.e., preprocessing, featuring engineering, and modeling) of each model were discussed in a rational order that shows how each step contributes to the next step. Table 1 below shows which model is involved in which data preprocessing activity.

Table 1: Preprocessing steps used by each solution

	Data Cleaning	Word Replacement /Correction	Stem ming	Lemmatiz ation
CS	√	×	×	×
RF1	√	√	√	×
RF2	√	√	√	√
RF3	√	×	√	×

Table 2 below shows types of the features used by each of the four models.

Table 2: Types of the features used by each solution

	Matching Features	Similarity Features	Meta Features	Attribute Features
CS	√	×	×	×
RF1	√	√	√	×
RF2	√	√	√	×
RF3	√	√	√	√

Table 3 below shows whether a model has applied single modeling or ensemble modeling and the techniques adopted by each model.

Table 3: Type of modeling used by each solution

	Single Model	Ensemble Model	Techniques
CS	√	×	Random forest
RF1	×	√	XGBoost, regressors, SVR, ridge, keras NN, RGF regression
RF2	×	√	Regressors, random forest, linear SVC, linear SVR
RF3	×	√	SVM, ANN, gradient boosted regressors, kNN, random forests

Implementation wise, all the four models used different Python packages. Also, RF3 used some R packages, while RF1 used rgf and libfm. All the four models were built using Windows platform except RF2 which used Ubuntu operating system for its experiments.
The next section will provide an interpretation of the results that we have obtained in this section.

7. DISCUSSION

The previous section has shown how the four models are different from each other regarding the three major parts: data processing, feature engineering, and modeling. Obviously, there is a large gap between CS model and the other three models while this gap is minimized between any of the reference models and the other two models.
Data preprocessing is concerned with removing undesired content from raw data and improving data quality. The reason that CrowdFlower provided noisy data is that it wants to mimic a real life scenario in which noisy HTML snippets and unnecessary information are present. In our model, data preprocessing was limited to data cleaning while ignoring other critical processes such as word correction/replacement, stemming, and lemmatization. For example, word replacement/correction is very important to align words with the same or similar meaning. While using data cleaning, word correction/replacement and stemming was critical, the use of word lemmatization seems to have no significant effect on the final result of this particular competition.
All contestants were interested in predicting the feature "Median Relevance." Finding how the query is related to product titles and descriptions would help to find certain values for this feature. Feature engineering was an important factor for winning an advanced position in this competition because the quality of the models was enhanced by creating ad-hoc features that allow machine learning models to make better decisions. For the three reference models, the contestants were able to select the most significant features from a pool of features. Features that are based on similarity (using for example cosine similarity, Jaccard similarity or Word2Vec Distance) were important. On the other hand, CS model used only two features for training and testing: word count features and weight features. These two features were used for calculating the correlation or distance between query and product title/description.
Regarding system modeling, our submission was limited to using an individual model for providing results. However, all the other three submissions used model ensembling to give results. One of the reasons that the other models gave better results than ours is that the product of model ensembling is in general better than any single model because the latter approach can correct, at least in part, errors resulting from variance, bias, and irreducible errors associated with the performance of almost every individual model.
The next section will give a brief conclusion about the most important points/discoveries discussed in this article.

8. CONCLUSION

The goal of this study is to understand the requirements of a successful data science project. This is done through providing an evaluation of the model design that we submitted to Kaggle to measure the relevance of search results of small online businesses. The study sheds light on specific aspects of model design to help others including Kagglers avoid possible mistakes while approaching their data science projects.
In order to build a reliable system that works efficiently under different circumstances, we need to test different features, different models, different machine learning tasks and different ensembling and rounding strategies. Nevertheless, testing so many different ideas can take a lot of time and much effort.
Working as a team with different perspectives and outlooks yet complementary skills can create a diversity of approaches and views on how to solve the same problem. Business wise, more companies rely on hiring a team of specialists (if the budget was big enough) instead of relying on a single expert person. The reason is that when a project is divided into smaller modules, each member is assigned a specific area and particular set of functions. However, exceptional data scientists can win competitions with no work team, and this particular competition is one of them.
Python programming language is absolutely an inevitable solution for all data scientists. Besides that the language is simple to learn and has an active online community, it has pre-built DS modules that can be used by users at all levels of experience. Compared to its main competitor (which is R), when it comes to free tools, Python is the best option to handle a DS project that requires a bundle of statistics, numerical computation, and web parsing capabilities.

9. REFERENCE

[1] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big Data*, vol. 1, pp. 51-59, 2013.

[2] F. Cady, *The Data Science Handbook*: John Wiley & Sons, 2017.

[3] D. Cielen, M. Ali, and A. Meysman, *Introducing data science: big data, machine learning, and more, using Python tools*: Manning Publ., 2016.
[4] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining* vol. 72: Springer, 2014.
[5] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*: Elsevier, 2011.
[6] A. Destrero, S. Mosci, C. De Mol, A. Verri, and F. Odone, "Feature selection for high-dimensional data," *Computational management science*, vol. 6, pp. 25-40, 2009.
[7] T. Ojeda, S. P. Murphy, B. Bengfort, and A. Dasgupta, *Practical data science cookbook*: Packt Publishing Ltd, 2014.
[8] F. Lazarinis, "Evaluating the searching capabilities of e-commerce web sites in a non-English language: A Greek case study," *Online Information Review*, vol. 31, pp. 881-891, 2007.
[9] A. Lee and M. Chau, "The impact of query suggestion in e-commerce websites," in *Workshop on E-Business*, 2011, pp. 248-254.
[10] R. Palanisamy, "Evaluation of search engines: a conceptual model and research issues," *International Journal of Business and Management*, vol. 8, p. 1, 2013.
[11] P. Schmutz, S. Heinz, Y. Métrailler, and K. Opwis, "Cognitive load in ecommerce applications: measurement and effects on user satisfaction," *Advances in Human-Computer Interaction*, vol. 2009, p. 3, 2009.
[12] M. Markland, "Does the student's love of the search engine mean that high quality online academic resources are being missed?," *Performance measurement and metrics*, vol. 6, pp. 19-31, 2005.
[13] M. Cao, Q. Zhang, and J. Seydel, "B2C e-commerce web site quality: an empirical examination," *Industrial Management & Data Systems*, vol. 105, pp. 645-661, 2005.
[14] M. Z. Al-Taie, S. M. Shamsuddin, and J. P. Lucas, "Predicting the Relevance of Search Results for E-Commerce Systems," *Int. J. Advance Soft Compu. Appl*, vol. 7, 2015.
[15] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, pp. 22-30, 2011.
[16] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: open source scientific tools for {Python}," 2014.
[17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273-297, 1995.
[18] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.
[19] C. Chen. (2017, Accessed: July 7, 2017). *"CrowdFlower Winner's Interview: 1st place"*. Available: http://jikeme.com/crowdflower-winners-interview-1st-place-chenglong-chen
[20] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 18.
[21] M. Trofimov. Kaggle 'Search Results Relevance' 2nd place solution [Online]. Available: https://github.com/geffy/kaggle-crowdflower/blob/master/description.pdf
[22] T. Quartet. (2017, Accessed: July 7, 2017). *CrowdFlower Winners' Interview: 3rd place*. Available: http://blog.kaggle.com/2015/07/22/crowdflower-winners-interview-3rd-place-team-quartet/

Biography

Mohammed Zuhair Al-Taie is currently a Ph.D. candidate at the Universiti Teknologi Malaysia (UTM), Faculty of Computing (FC). He holds master's degree in computer science and communication from the Arts, Science and Technology University (AUL) in Lebanon and bachelor's degree in computer science from Al-Mustansiriya University in Iraq. He has published a number of studies on topics like social networks, e-government, and e-commerce as well as two books on social network analysis. His fields of interest include machine learning, social networks, and advanced web technologies. Previously, Mr. Taie was a university teacher in Iraq where he taught modules at the undergraduate level in subjects related to artificial intelligence, Web technologies, and software engineering. He has been a reviewer for Social Network Analysis and Mining (SNAM) journal for several years. His Google scholar page can be found here https://scholar.google.com/citations?user=JdLRwWsAAAAJ&hl=en.

Naomie Salim is currently a Professor in the Faculty of Computing at the Universiti Teknologi Malaysia. She has a Master's Degree in Computer Science from the Western Michigan University and a Ph.D. in Information Studies (Chemoinformatics) from the University of Sheffield. She has taught at both undergraduate and postgraduate levels in subjects related to databases and information systems, and her research interests include information retrieval and chemoinformatics. Her Google scholar page can be found here https://scholar.google.com/citations?user=PcYJDfoAAAAJ&hl=en.

Obasa Adekunle Isiaka is currently a Chief Lecturer in Department of Computer Science, Kaduna Polytechnic, Kaduna, Nigeria. He obtained a Ph.D. degree in Computer Science from University Technology Malaysia. His Master degree in Computer Science was obtained from Federal University of Technology, Akure while his Bachelor degree in Industrial Mathematics was bagged at University of Benin, Benin City, Nigeria. He has more than 20 years of lecturing and research experience. He is currently an external examiner in Computer Science to some institutions. He has several publications to his credit.