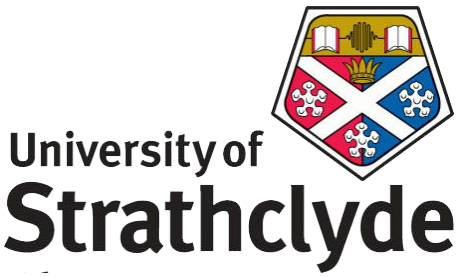

CS801: Quantitative Methods

Main Module Assignment

November 28, 2021

Contents

List of Figures

The aim of this paper is to identify and critically evaluate the quantitative methods used in three Kaggle notebooks, to present alternative approaches and visualisations, and to offer recommendations. For each notebook, the original cells are referenced as [Number of Cell], while the cells containing the student's code are denoted as [Number of Cell MCC].

1 "Do you wanna predict price of car you wanna buy?" by Aishwarya Wuntkal

1.1 Introduction to the Notebook and main objectives

The purpose of the notebook[1] is to implement a supervised learning model to predict car prices. It makes use of data, related to the price of the car, fuel type, transmission type, and owner type, derived from a website called caredkho.com, specialised in the sale of new and used car across Asia, and, in particular, India. As the author specifies, the dataset will be used to make price predictions by means of linear regression. Such a model, which assumes a linear relationship between the dependent and the explanatory - independent- variables, is clearly suitable for the author's aim, as the outcome variable, the price of the car, lies in a continuous range.

1.2 Use of Quantitative Methods, Observations and Recommendations

[6] The author derives the **summary statistics** after having observed the shape of the data-set and having had a first look at the table, and before having checked for **missing values**. The author does not evaluate the findings. For example, the average selling price is 4.66, with a relatively high standard deviations of 5.08 Lakh. The **units of measurements** are not provided by the author, but simply by looking at the website it is easy to see that the currency in which prices are expressed is Indian rupees. In particular, the numbers reported refer to Lakh, where one Lakh corresponds to 100 000 rupees. Reporting this information in the notebook would have been helpful for international readers to understand the magnitude of the values.

The author immediately concludes that there is no **outlier** and that the values are gradually increasing. The author does not explain where they derive this conclusion from and treats outliers superficially. To detect outliers, the author, among the other methods, could have used, a Z-score treatment or boxplots (Appendix: Figure 1) to detect that, indeed, there are outliers. Futhermore, by using the **Interquartile Range (IQR)**, which measures the statistical dispersion of data, we note, for example, that there are 17 outliers for the selling price (See [9 MCC]).

[11] The author makes use of **MissingNo** to check for missing values. MissingNo is a module that provides the tools to easily visualise missing data and have a general overview of the degree of completeness of datasets (Bilogur, 2017). As the **nullity matrix** in the notebook shows, the dataset appears to have no missing values. However, it would have been more appropriate to double check the presence of missing values by, for example, using the **.isnull()** (or .isna() or .info()) function and computing the sum of the missing or null values [2 MCC], as their presence could lead to a lower degree of accuracy in the process of drawing inferences about the data.

[13] The author visualises the data frame using a **seaborn pairplot**, probably with the aim to detect patterns in the data. However, the visualisation is not particularly explanatory, and the author does not evaluate it.

[14] The author does very well by plotting the **correlation heatmap**, which helps visualising the relationship between the elements of the dataset. Such a technique is also useful to detect the correlation between the independent variables themselves, crucial to identify multicollinearity, which does not appear to be present in this case. The author does not evaluate the findings. It is not clear, at this point of the notebook, which **hypothesis** is guiding the author in their analysis.

[15] The author uses a **seaborn bar-plot** to visualise the selling price proposed by dealers against the one offered by individuals. The black lines represent the standard deviation. Using the bar-plot certainly helps to visualise the difference in prices -higher for dealers, as also noted by the author. It is an effective way to show the differences in outcome of categorical variables. Similarly, in **[17]**, they visualise the selling price of manual transmissions cars against the one of cars with automatic transmissions. The graph shows that selling prices are higher for the latter, as also noted by the author. In **[18]**, the author plots a bar-plot to compare the selling prices of cars on the basis of their fuel type, showing that diesel cars have the highest prices.

[1] Available at: https://www.kaggle.com/aishu2218/do-you-wanna-predict-price-of-car-you-wanna-buy

[19] The author uses **seaborn.regplot** to observe the relationship between the selling price and the current price of the cars. This method is used to visualise data and a linear regression model fit. Interestingly, it is not possible to access the statistical values used to generate this plot, as, Michael Waskom, the lead developer of seaborn rejected access to these features [2]. The graph shows a positive, obvious [3], relationship between the two variables. The choice to put the present price on the vertical axis contradicts the purpose of the author to look at how the selling price of the cars changes with respect of other variables, as this method requires the x to be specified before the y variable. The graph has no title. The author claimed that there was no outlier in the data, but, as also seen above, this graph reports the presence of outliers. Seaborn.regplot can make use of the specification "robust" to deweight outliers [4]. By re-plotting the graph with selling price on the vertical axis (Appendix: Figure 2; Code: [23 MCC]) and using the **robust** method, we observe a moderate positive relationship between the two prices.

[20] As in **[19]**, the author inverts the x and y variable in the **regression**, when looking at the effect of kilometers driven on the selling price. Again, the graph has no title. The author interprets an upward sloping regression line as a negative relationship between the variables. By re-plotting the graph with selling price on the vertical axis (Appendix: Figure 3; Code: [25 MCC]) and using the robust method, we observe a weak and positive relationship between kilometers driven and selling price. This result is counter-intuitive, as we would expect a significant negative correlation, as more used and older cars tend to be sold at a lower price. However, this finding could be explained by the fact that kilometers driven alone are not enough to determine the price of a car, as other factors related to wear and tear and usage could play an important role. Indeed, there is a high variance in the outcome that is not captured by this regression. Furthermore, by looking at the data points on the scattered part of the graph, we notice that the highest prices are observed at lower numbers of kilometers driven.

[26] The author does the same as in cells **[15], [17], and [18]** to compare the selling price of cars with respect to the number of owners the car had. In **[27]**, they use a bar-plot to visualise how the price of the car changes with respect to the age of the car. The author does not expand in the analysis of the graph, despite some interesting points to notice. For example, even if the price decreases with an increase of the age of the car, we observe that for cars of 10 years of age, the price is considerably higher than for cars between 6 and 9 years and 11 and 17 years of age.

[29] The author generates **dummy variables** to better deal with **categorical variables** in preparation for the linear regression they aim to conduct. They convert the columns Fuel_Type, Transmission and Seller_Type. They then drop the column corresponding to the the third fuel type, cng, in cell **[30]**, clearly to avoid multi-collinearity[5]. Converting Fuel_Type into a binary variable does not make much sense. The author does not explore whether there are hybrid cars in the dataset. Therefore, dropping this column may result in a loss of information.

[31] The author uses the **pandas.concat.()** function to add the new columns to the original dataset, and, in cell **[33]**, they drop the columns related to the original categorical variables. In cell [34] they drop the column "Car_Name". Since the names of the car seem sparse and there is no clear reference to the brand, this seems like a justified choice. Anyways, in real-world terms and intuitively, we would expect the name of a car to influence its perceived value.

[36] The author now proceeds to split the data into test and train in preparation of the linear regression. They arbitrarily choose a conventional **30/70 split**, as the dataset has a limited number of observations. Cross-validation would have been a better alternative. After the split, the author standardises the numerical variables **[37-39]** in the training set, by processing the dataset such that it behaves more or less according to a Gaussian distribution, by means of the **.StandardScaler()** function [6].

[38] The function .fit_transform is correctly applied to scale the numerical training data and get each column standardised independently, while .transform() is used on the numerical test data.

[2] See https://github.com/mwaskom/seaborn/issues/655 for a discussion between Michael Waskom and the seaborn users.

[3] The choice of using the current price as a predictor of selling price is in itself questionable given how close the two variables are.

[4] https://seaborn.pydata.org/generated/seaborn.regplot.html

[5] O'Brien (2016) makes a case against dropping highly collinear variables from regression models.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[40] - [41] The author splits the training dataset into Features and Target. The .pop() function, used to generate the y_train set, returns the element "popped" from the list, in this case Selling_Prices. The author uses the module **sklearn.feature_selection with RFE**, i.e. recursive feature elimination, to improve the accuracy of the estimators. The estimator is initally trained on the entire set of explanatory variables, and then their importance is derived. The least important are pruned from the x_train set. These steps are repeated until a satisfactory number of independent variables is obtained [7].

[46] The author adds a constant to the regression model. This guarantees that the residuals of the regression will have a mean of zero, and determine the intercept of the regression line with the y-axis - i.e. the selling price when all the explanatory variables are set to zero.

[47] The author estimates an **Ordinary Least Squares (OLS)** regression model. The coefficients have signs as expected. For example, an increase in one unit in Kms_Driven is indeed associated with a decrease of 0.0412 Lakh in the selling price, and a increase of 1 Lakh in the present price is linked with an increase in the selling price of 0.7424 Lakh. The author does not analyse the results of the regression, and only resolves to drop the variable Petrol due to its coefficient being insignificant (p-value > 0.05). However, at the 5% significance level, the coefficients on Diesel, Kms_Driven and Owner are also not statistically different from zero. Since no further justification for this choice is offered, it is not clear why the author decides to drop Petrol and not the other variables as well. Dropping a variable solely on the grounds of its coefficient being statistically insignificant is a poor choice. Moreover, as outlined by Hastie et al. (2008: 60), "smart stepwise procedures [...] will add or drop whole groups [**of dummy variables relating to the same category**[8]] at a time, taking proper account of their degrees-of-freedom". The **R-squared** suggests that 89.6% of the variance in the selling price is explained by the independent variables. While this might appear as a commendable result, such a high R-squared is by (mathematical) definition **upwardly biased** (Verbeek, 2017). The same model, if applied to a new dataset, would likely lead to poorer estimates and fits. Finally, the **Durbin-Watson** test statistic, being close to the critical level of 2, shows that there is **autocorrelation** in the model, which the author does not account for.

[48 - 53] The author iteratively deletes variables with statistically insignificant coefficients and re-estimates the model following a process of Backward Feature Elimination for dimensionality reduction[9]. The coefficients of the remaining variables change. It is likely that the omitted and included variables are correlated.

[55] The author makes use of the module **statsmodels.stats.outliers _ influence .variance _inflation_factor (VIF)**[10] . This is a tool to measure how the variance of the estimates changes if an additional variable is included to the linear regression. If the VIF is greater than 5, then the explanatory variable is collinear with the others. This is not the case here.

[57] The author performs an analysis of the **regression residuals**, i.e. they evaluate the predicted values of y with its observed values to determine the appropriateness of the regression. The visualisation does not appear to be suitable. Figure 4 (Appendix; Code: [58 MCC]) provides an example of an alternative visualisation and shows that, since the points are randomly scattered around 0, the linear regression is a suitable model to apply to the dataset.

[62-END] The author compares the values of the y_test set and the predicted values of the selling price in y_pred to evaluate the goodness of fit of the model. The visualisation is sufficiently explanatory. As also shown in **[63]** by the difference between actual and predicted price, the model does **not** predict the selling price with a high degree of accuracy, which could be improved by adding more data, for example.

The balance of this review is generally positive, but there a few points of improvement. Across the entire notebook, the author does not evaluate their findings, and does not offer guidance to the reader in respect to where the analysis is going. They do not offer methodological notes [11] and do not justify their choices, and fail in detecting both outliers and the presence of auto-correlation (clearly shown by the OLS output). Instead, at the end of their notebook, even though the author reaches the stated goal of finding a method to predict the selling price, they just repeat what already stated during the EDA phase. The reader is **left to wonder** about the findings and any possible inference to be drawn from them.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_s election.RFE.html$

[8] Addition of the student.

[9] **Why not a Principal Component Analysis?**

[10] https://www.statsmodels.org/stable/generated/statsmodels. stats.outliers_influence.variance _inflation_factor.html

[11] These could be useful given the audience the Kaggle notebook is directed at to push forward discussion.

2 "IEEE Fraud Detection - First Look and EDA" by Rob Mulla

2.1 Introduction to the Notebook and main objectives

The purpose of this notebook[12] is to conduct an EDA on the IEEE Fraud Detection Dataset. The notebook was produced to take part in a competition launched by the IEEE Computational Intelligence Society in partnership with Vesta Corporation, the world's leading payment service company, in an effort to find solutions to detect and prevent fraudulent activities. The dataset, provided by Vesta, contains data related to real-life e-commerce transactions. It offers information on device type, credit card type, e-mail domains, time references, transaction amounts in USD dollars, addresses of the payees, identity, and other features not better specified probably for the sake of confidentiality. The author's stated aim is to predict the probability that an online transaction is fraudulent. The target variable is a binary variable, 'isFraud', for which a logistic regression model would have been suitable if it were not outside the scope of the analysis.

2.2 Use of Quantitative Methods, Observations and Recommendations

[5] The author makes use of the **numpy.unique()** function to find the sum of the unique elements in both the train and the test sets for transactions and identities. They then find the percentage of TransactionIDs (24% in the train set, and 28% in the test set) that have an an associated train_identity.

The author notes that the temporal gap between the train and the test sets will have an impact on choosing the more appropriate cross validation technique. Indeed, splitting the dataset in this situation is a non-trivial task that cannot be conducted randomly, as, in rough terms, it would not make sense to use data from the future to predict the past (Hermann-Saffar, 2020).

The temporal gap is also shown in **[6]**, where author plots an histogram to compare the distribution of the TransactionDT, the timedelta[13] from a given reference datetime. The graph is well made, with a legend, a title, and different colours to identify the train and the test set. In **[7]** the author plots two scatterplots to show the transaction amount in test and train across time. By means of the **.loc()** function, the author is able to access the fraudulent transactions, which they then add to the plot. The author does not evaluate the graph, which shows a higher concentration of fraudulent activities in the train set (left-side). In **[9]**, the author, using an horizontal bar chart, shows the amount of fraudulent transactions against the total number of transactions in the train set.

[10] The author plots the distribution of the logarithm of the transactions amount. They loglinearise the data in order to deal with the possibility of large transactions to skew the distribution. The logarithmic transformation of the data helps to decrease the variability of data and makes it closer to a standard Gaussian distribution with mean zero and unit-variance (Feng et al., 2014). The author notes that any values between 0 and 1 appears to be negative. As also outlined by Kaggle user "abccba" in a comment to the notebook[14], even if the bin is in the negative area of the graph, the corresponding transaction amount in absolute value is less than one and non-negative.

[13-14] The author does well in plotting bar graphs to show the number of transactions and the percentage of fraudulent transactions by category of ProductCD. Albeit the interpretative limitations given by the lack of information about what the categories actually refer to[15].

[15] The author proceeds by looking at the columns referring to card types. They note that the data were presented as being **categorical**, but some of them, card 1, 2, 3, 5, appear to be numeric. The output of cell [15] shows the presence of missing values, which are left untreated and apparently unnoticed. We also observe a lack of exploration in terms of datatypes, summary statistics, and detection of outliers.

[16-17] The author plots 8 histograms to visualise the distribution of transactions for both non numerical and numerical[16] card type data. It is not clear why, in **[17]**, the labels on the y-axis for the Count of card6 non-fraud are more than in the graph for the Count of card6 fraud. The graphs show that the credit card with the highest amount of transactions and frauds is VISA, followed by MASTERCARD, and that frauds are more frequent for debit cards. It would be interesting, to get a better idea about which type of card is more likely to be associated with fraudulent transactions, to have data on the distribution and the adoption of the different

[12] Available at: https://www.kaggle.com/robikscube/ieee-fraud-detection-first-look-and-eda
[13] The difference in times. Units are not specified in this case.
[14] https://www.kaggle.com/robikscube/ieee-fraud-detection-first-look-and-eda/comments
[15] We could assume that W stands for wholesale, S for services, R for retail, C for consumer goods.
[16] The graph for card1 is surprisingly similar to the Glasgow skyline.

cards in the whole population, so that a more accurate proportion could be established. The reason why the author does not explore the number of frauds for or the correlation between this and the other card types is unknown.

[18] The author correctly checks for missing values in the addr1 and addr2 columns using the **.isna()** and **.sum()** functions. After detecting the missing values, the author does not act in removing them or in dealing with them by means, for instance, of a **linear interpolation** of the pandas dataframe or of the module sklearn.impute.SimpleImputer[17] with a suitable method of filling the cells. Arguably, keeping the missing could be seen as a prudent choice, as the pattern of missing values, present in a high number in the columns under consideration, might carry valuable information and be somehow foretelling. Nonetheless, the author plots data without taking into consideration the high proportion of missing values, resulting in a less than ideal level of accuracy.

[22] The author plots a pairplot of the columns from C1 to C14, in order to gain intuition about potential feature interactions. Questionably, it is difficult to gain intuition from a visualisation that is not *legible*. The author uses the method hue='isFraud' to visualise the fraudulent transactions associated with the columns. It is not clear which colour identifies them (we assume that red does). Same for **[24]**.

[26] The author plots graphs showing the number of T, F, and missing values by M. No specific information are provided by Vesta about these variables. These graphs offer no particular insight, apart that in column M4 are present different values, i.e. MO, M1, M2.

[32] The author makes use of the pandas .merge() function, a join operation, to add the 'isFraud' column to the data-frame to conduct analysis.

[33] The author uses the methods **.groupby(), .mean(), and .sort_values()** to plot the percentage of fraud by device type, showing that this is higher for mobile devices.

[35-END] The author visualises the identity info and the numeric features of the train and test set as a function of time by using the **DataFrame.set_index()** method with TransactionDT as the index. The graphs are not explanatory in themselves, and the author ends the notebook without evaluating them or making any comment about the findings of their exploratory data analysis, which forgets to take into account other variables, such as the e-mail domain associated with transactions and frauds.

After inspecting this notebook, it appears that choosing to visualise the variables in broader categories (e.g. C to C14) was not the best approach to get insights from such a vague and broad dataset. The author, for instance, could have analysed the variables (columns) one by one to be able to derive more valuable information. Instead of just visualising the numerical variables across time in absolute terms, the author could have explored their correlation with the time variable, and also expanded on the correlation between the variables. The author ultimately does not achieve their stated goal of identifying all the variables that are associated with an higher likelihood of fraud.

3 "Disaster Tweets – Word2Vec and tf-idf" by Maxim Knyazev

3.1 Introduction to the Notebook and main objectives

The purpose of this notebook[18]is to discern between tweets that are about a real disaster and tweets that are not, by means of two word embedding techniques, of Word2Vec, a two-layer neural net that computes word embedding via word vectorisation, and TF-IDF, a weighting factor whose value increases proportionally to the number of times a word appears in a collection and measures how important a word is (Goldberg Levy, 2014; Wu, 2008). The dataset, provided by the company Figure-Eight, is publicly available on the 'Data For Everyone' website. It contains 10,000 hand-classified tweets. There are 5 columns: an id that uniquely identifies the single tweets, the text of the tweet, the location the tweet was sent from, a keyword from the tweet, and a binary variable (target) that denotes the realness of the tweet (1 if the tweet is about a real disaster, 0 otherwise)[19].

[17]https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html
[18]Available at: https://www.kaggle.com/maximknyazev/disaster-tweets-word2vec-tf-idf-81-acc
[19]See: https://www.kaggle.com/c/nlp-getting-started/data

3.2 Use of Quantitative Methods, Observations and Recommendations

[11- 15] The author does well in making use of the **SpaCy library** to detect the stopwords present in the tweets. They then define functions to return the word count, the character count, the average word lenght, the hashtag count, the digit count and the upper case counts of the tweet. This gives a deeper overview of the tweets.

[16] The author starts the EDA by using a seaborn.displot() function to visualise the **univariate distribution** of the characters count. In [17] the author uses a kernel density estimate (KDE) plot to compare the distribution of character counts in real and fake disaster tweets, showing that, on average, when the target is equal to zero, the tweets tend to have less characters. These findings are supported by the violin plot in **[18]** that beautifully and effectively re-proposes the aforementioned comparison. The EDA stops here.

3.2.1 BAG OF WORDS: IF-TDF

[19] The author proceeds to **clean the data**. As the IF-TDF model results in each word of the vocabulary becoming a **feature**, a correct and thorough preprocessing of the data is crucial in reducing this **dimensionality problem**. The author defines a series of functions to remove the username, to transpose the words in lower cases, to fix the contractions, to generate strings, to remove the URL, the email, the retweets, the HTML texts, the dots, any special character, to **tokenise** the tweets in the dataframe, to apply **spelling corrections**, to return the base form of the word[20] (**stemming**), and to replace all pattern occurrences in the target strings[21].The author does not seem to include a **lemmatisation**, crucial for a better contextualisation of the words, in their data pre-processing.

[26] The stopwords are removed from the text data using the Gensim remove_stopwords module.

[30-34] The author generates two word clouds to visualise respectively the most common words in the real disaster tweets and the fake disaster tweets. This is a simple visualisation technique. It would have been interesting to see a wordnet[22] to check the concurrences of words in the tweets. A careful inspection of the output in **[32]** that shows that data cleaning could have been conducted more accurately, as there are accents, random spare characters, some words in other languages and nonsensical words (e.g. viva argentina, ûïall, òåêcnbc, jhmnye, clipuri, auth, ghe).

[36] The author **vectorises** the text into n-gram integer vectors to apply the **TF-IDF statistical measure to the dataframe**. To do so, they make use of the Scikit Learn function .TfidfVectorizer()[23]. TF-IDF multiplies two metrics, one related to the frequency of appearance of a word in a text document, and one measuring how rare or common the word is in the document.

The author now proceeds to implement six supervised learning models.

[40-42] First, the author applies a **Linear Support Vector Classifier (SVC)** to the training data. This is a binary classifier that minimises squared hinged loss, using the OWLQN optimizer, and penalises the size of the bias (L2 regularisation)[24]. The precision in telling real from fake tweets is respectively 0.79 and 0.78; the accuracy is 79%, which is a good enough result.

[43-45] Second, the author performs a **logistic regression**, which is suitable as the target is a binary variable. As showed in **[45]**, the precision of this model to predict real disaster tweets is 0.86, 0.08 points higher than for the Linear SVC.

[46-48] Third, the author uses the non-parametric **K-Nearest Neighbours** classification method. They use the .KNeighborsClassifier() function[25]. By default, the number of neighbours used is 5 and weights used in prediction are set as uniform (ibid). The accuracy, and the precision for both values of the target variable are lower than for the previous models. The weights could have been set as 'distance', a method that gives more importance to closer neighbours, and, in cross-validation, the author could have have compared the results

[20]The author uses the Token.lemma_ function, which returns "the base form of the token, with no inflectional suffixes". Source: https://spacy.io/api/token
[21]https://pynative.com/python-regex-replace-re-sub/
[22]https://pypi.org/project/wordnet/
[23]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[24]https://docs.w3cub.com/scikit _learn/modules/generated/sklearn.svm.svcsklearn.svm.SVC
[25]https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

and choose the method with the lowest cross-validation loss (Clements, 2021). Indeed, [MCC 51-52] show, for instance, that when distance is used as a weight and the number of neighbours is reduced to 3, the precision increases for fake tweets up to 80% (but decreases for real tweets from 90 to 73%), and the overall accuracy increases from 0.67 to 0.77.

[49-51] Fourth, the author uses a **Multinomial Naive Bayes classifier**, which is suitable for this problem of text classification with fractional counts with TF-IDF[26]. This model seems to propose a good balance between the precision in detecting fake and real tweets, and has an accuracy of 89%, the highest observed up to this point.

[52-54] Fifth, the author uses a **Decision Tree Classifier**, a non-parametric supervised classification method. The predictions obtained by using it are neither smooth nor continuous, but segmented approximations. They are, thus, not suitable for forecasts, but adapt to the current problem. The accuracy and the precision are lower than for previous models. The output of [55 MCC] provides a visualisation that is somewhat helpful to visualise the functioning of the model (Appendix: Figure 5).

[58-62] Lastly, the author uses a **Random Forest Classifier**. This is a meta estimator that applies different decision trees (by default, 100) on sub-samples of the dataset and computes averages to foster accuracy and control for overfitting[27].

[63-65] The author computes and compares the **scores** of the models. In [65], the author visualises the comparison. The graph is superfluous as the scores are really similar, so that it is not easy to identify the best one by simply looking at the graph. The MNB has the highest score of 0.79, followed by the LR score of 0.78.

3.2.2 WORD EMBEDDING: WORD2VEC

[66-77] This time, the author vectorises the text by means of the **Word2Vec technique**. One of the main features that distinguishes Word2Vec from TF-IDF, is that this scans the entire corpus and the vectors are created after determining the words that most frequently are concurrent with the target, i.e. the words that appear more frequently in real disaster tweets. This method, therefore, also reveals, mathematically, how close the words on a semantic basis (Akdogan, 2021). In **[71]** the author uses .to_numpy() to **convert the Pandas series into a numpy array**, and then they **resize its elements** by means of the the .reshape() function. In **[73]**, they use the **numpy.concatenate()** function to join the arrays of the same shape along the horizontal axis. In **[76]**, the author, via **sklearn.normalise()**[28], scales the input data set (train) on a scale from 0 to 1 (**Unit norm**).

[78-84] The author trains with W2V the same models used above with TF-IDF. The output in **[81]** shows the results. We can observe that the precision of these models is higher than for their counterparts described above. The author could have fit the data on the corpus using Gensim (as done above), and determining the maximum distance between the words, and the training algorithm (e.g. skip-grams) (as done, for example, in Di Pietro, 2020).

3.2.3 HYPERPARAMETER TUNING

[82-106] The author conducts an hyperparameter tuning for the logistic regression model. It is not clear, and the author does not provide the reasons why they choose this model over the MNaive Bayesan model, as this performs marginally better both with TF-IDF and Word2Vec. In **[120]**, the author codes a dictionary for the parameter grid and searches over specific parameter values, to **find an estimator**, by means of the Grid Search CV method [29]. Once the **best parameter**, which, in this case, give a level of accuracy of 80.09%, is found (penalty = 'l2'), the author incorporates these findings in a final run of the logistic regression model (with Word2Vec). Applying an L2 penalty tends to result in small but non-zero regression coefficients (Goeman, 2010).

Overall, the work conducted in this notebook, which is very well structured and presents clear and elegant code, is commendable. The authors ultimately manages to generate fairly accurate predictions about the realness of disaster tweets. From the point of view of a learner, it would have been highly precious to have more

[26] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[27] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest%20classifiersklearn.ensemble.RandomForestClassifier
[28] https://scikit-learn.org/stable/modules/generated/ sklearn.preprocessing.normalize.html
[29] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

comments related to choices made by the author in regards to the different models and methods they made use of. The notebook would have also benefited from more visualisations.

References

Akdogan, A. (21). 'Word Embedding Techniques: Word2Vec and TF-ID Explained'. Towards Data Science. Available at: owardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08

Bilogur, A. (2017). 'Missingno'. GitHub public repository. https://github.com/ ResidentMario/missingno

Di Pietro, Mauro (2020). 'Text Classification with NLP: Tf-IDF vs Word2Vec vs Bert'. Towards Data Science. Available at: https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794.

Feng, C., Wang, H., Lu, N. et al.(2014). 'Log-transformation and its implications for data analysis'. Shanghai Arch Psychiatry. 2014 Apr; 26(2): 105–109. doi: 10.3969/j.issn.1002-0829.2014.02.009

Clements, J. (2021). 'K-Nearest Neighbours (KNN) Explained: the intuition behind a simple, but powerful algorithm. Available at: https://towardsdatascience.com/k-nearest-neighbors-k-nn-explained-8959f97a8632

Goldberg, Y., Levy, O. (2014). 'word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method'. Available at: arXiv:1402.3722.

Goeman, J. J.. (2010) 'L1 penalized estimation in the Cox proportional hazards model'. Biometrical Journal 52(1), 70–84. doi: 10.1002/bimj.200900028.

Hastie, T. (2008). Tibshirani, R. and Friedman, J.(2009): The elements of statistical learning. Data mining, inference, and prediction. Second Edition. Springer. Available at: https://web.stanford.edu/ hastie/Elem-StatLearn//
printings/ESLII_print10.pdf.

O'Brien, R. M. (2017). 'Dropping highly collinear variables from a model: why it typically is not a good idea'. Social Science Quarterly, 98(1), 360-375.

Wu, H. Luk, R., Wong, K., Kwok, K. (2008). 'Interpreting TF-IDF term weights as making relevance decisions'. ACM Transactions on Information Systems, 26 (3), pp. 1-37. Available at: https://www.semanticscholar.org/paper/Interpreting-TF-IDF-term-weights-as-making-Wu-Luk/f6bbbf2cc785cf96019dcd9c41ab1801aad962dd.

Appendix

Figure 1: Outliers in Selling Price (Indian Lakh) (Notebook 1)

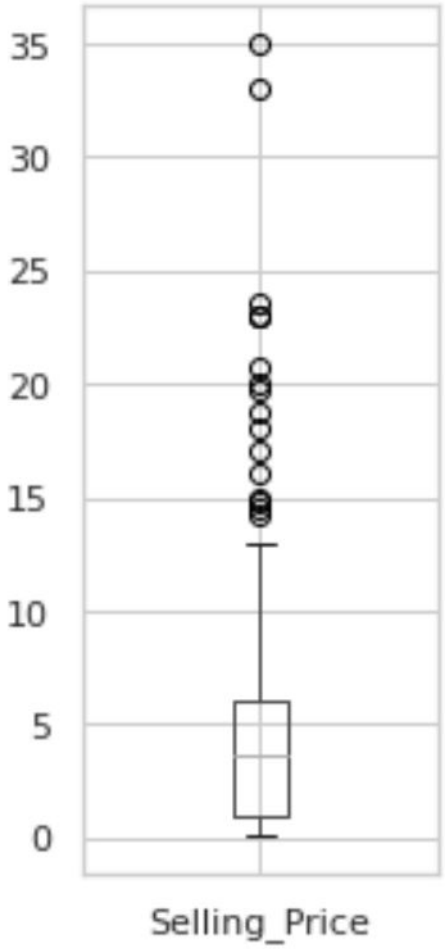

Figure 2: Linear regression of selling price on present price (Notebook 1)

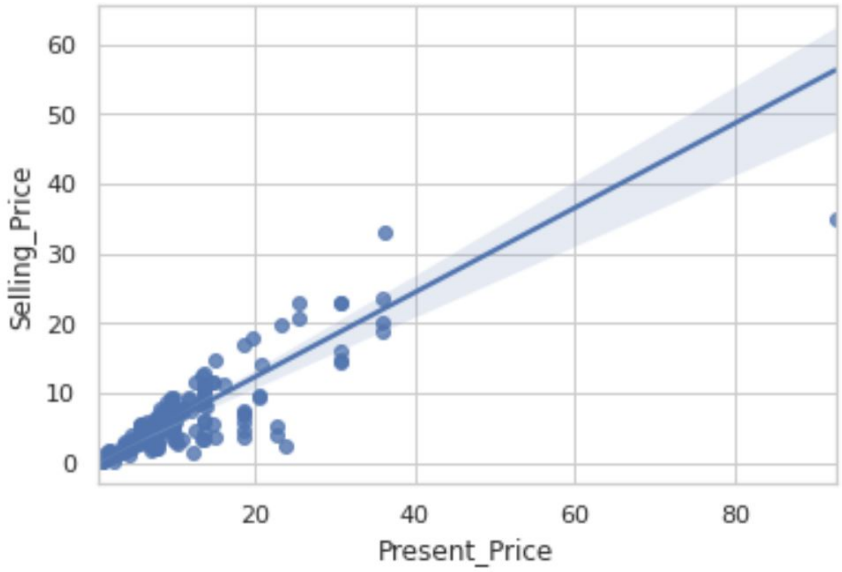

Figure 3: Linear regression of selling price on kilometers driven (Notebook 1)

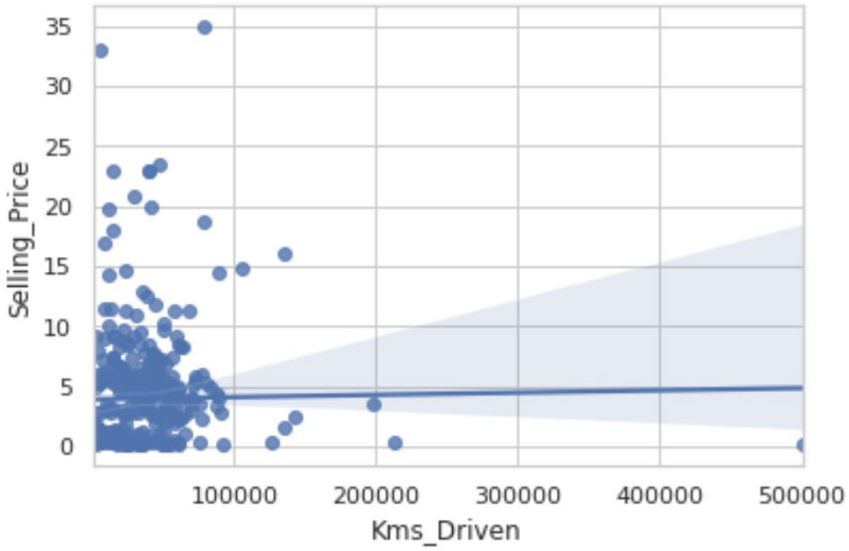

Figure 4: Standard Residuals of the Linear Regression (Notebook 1)

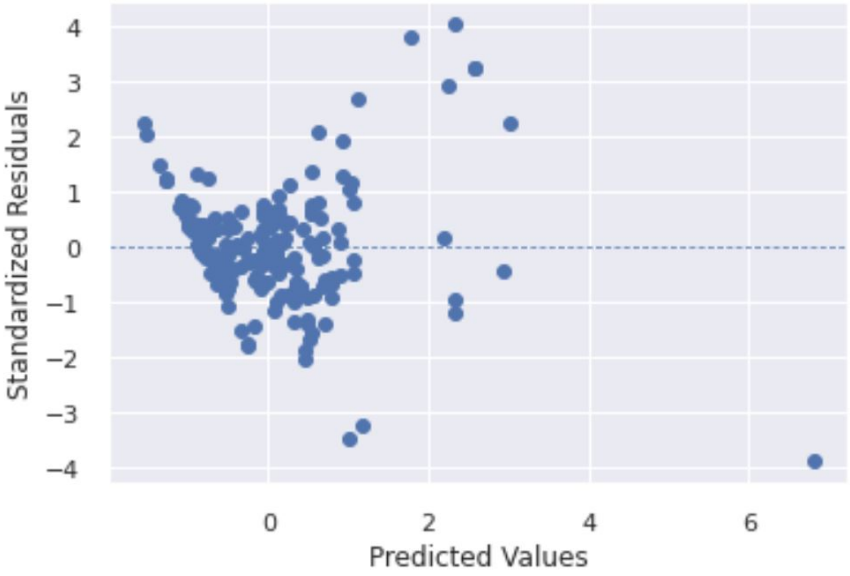

Figure 5: Decision Tree (Notebook 3)