ITM760 Big Data Analytics

Course Project

Dr. Morteza Zihayat Kermani

Monday April 19, 2021

Team Lead: Arif Shash (500827924)

Sukhkaran Grewal (500844346)

Harman Pawar (500901587)

Rafa Mohammed (500909564)

Introduction

Given the two datasets (online new popularity & online shoppers purchasing intention) we used the following algorithms SVM, k-Nearest Neighbor and Decision Tree in order to model the training set and classify the examples in the test set. For Dataset 1 (online new popularity), the target attribute was to predict the number of shares in social networks. This was done, through using a decision tree and then using the KNN algorithm in order to estimate how good of a fit the values were. After this the SVM algorithm was used to determine the following: classification, regression and outliers detection. Moreover, in regards to Dataset 2 (online shoppers purchasing intention), the goal was to use all the cases in order to figure out which user will buy something from the portal. The same steps were used that were used for Dataset 1 were repeated for Dataset 2. In addition there were 4 learning methods that were used in this analysis, Non-Predictive Attributes, SKlearn SelectBest Function and Pandas.Cut for the first data set ('Online_news_popularity') and only 1 learning method for the dataset ('Online_shoppers_internion') was used, Convert to Categorical to Numerical.

Data Exploration

During the processing of the datasets, multiple functions and algorithms were used. For Dataset 1 (online new popularity) the target attribute was to find the number of shares in social networks. The first step is to import data which is where we imported the data and created a dataframe object ex, import pandas as pd, import matplotlib.pyplot as plt. Additionally, it is important to describe data, we used the function newsdata.describe() in order to see what data ranges the values are in and newsdata.isnull().sum() in order to detect any invalid or missing values within

the dataframe. After using the newsdata.describe() function we discovered the data ranges as the following: mean, max, min and count.

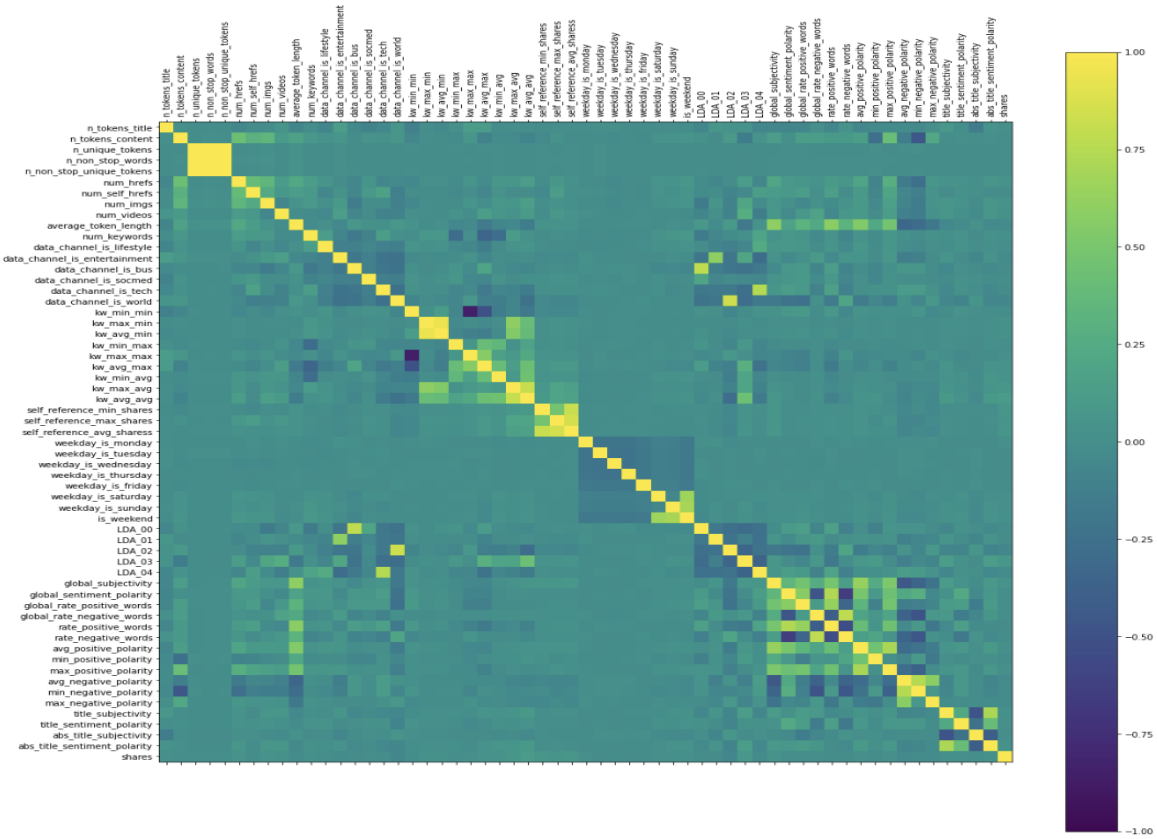
	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens
count	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000
mean	354.530471	10.398749	546.514731	0.548216	0.996469	0.689175
std	214.163767	2.114037	471.107508	3.520708	5.231231	3.264816
min	8.000000	2.000000	0.000000	0.000000	0.000000	0.000000
25%	164.000000	9.000000	246.000000	0.470870	1.000000	0.625739
50%	339.000000	10.000000	409.000000	0.539226	1.000000	0.690476
75%	542.000000	12.000000	716.000000	0.608696	1.000000	0.754630
max	731.000000	23.000000	8474.000000	701.000000	1042.000000	650.000000

Using the newsdata.isnull().sum() we discovered that there were no missing values within the dataframe.

```
url                                0
timedelta                          0
n_tokens_title                     0
n_tokens_content                   0
n_unique_tokens                    0
                                  ..
title_subjectivity                 0
title_sentiment_polarity           0
abs_title_subjectivity             0
abs_title_sentiment_polarity       0
shares                             0
Length: 61, dtype: int64
```

Also using the newsdata.shape function we were able to deduce the shape and size of the dataset which was (39644, 61). In order to visualize the data we used histograms and boxplots. This was done using the following functions, Histogram: newsdata.drop(['url', 'timedelta'], axis=1).hist(bins=30, layout=(20,3), figsize=(50,80)) & Boxplot: newsdata.drop(['url', 'timedelta'], axis=1).plot(kind='box', subplots=True, layout=(15,4), sharex=False, sharey=False, figsize=(30,200), title='Boxplot for each input variable'). This allowed us to visualize the dataset to detect any irregularities that could have occurred such as, any outliers within the dataset.

Lastly, a correlation analysis was done to determine the correlation between the different attributes within the dataset. This helped to identify the relationships between 2 attributes in order to deduce whether they had a strong correlation meaning they were related or weak correlation meaning they were not related.



For Dataset 2 (online new popularity), the goal was to use all the cases in order to figure out which user will buy something from the portal. The first step is to import data which is where we imported the data and created a dataframe object ex, import pandas as pd, import matplotlib.pyplot as plt.Additionally, it is important to describe data, we used the function shopsdata.describe() in order to see what data ranges the values are in and shopsdata.isnull().sum() in order to detect any invalid or missing values within the dataframe. After using the newsdata.describe() function we discovered the data ranges as the following:

mean, max, min and count.

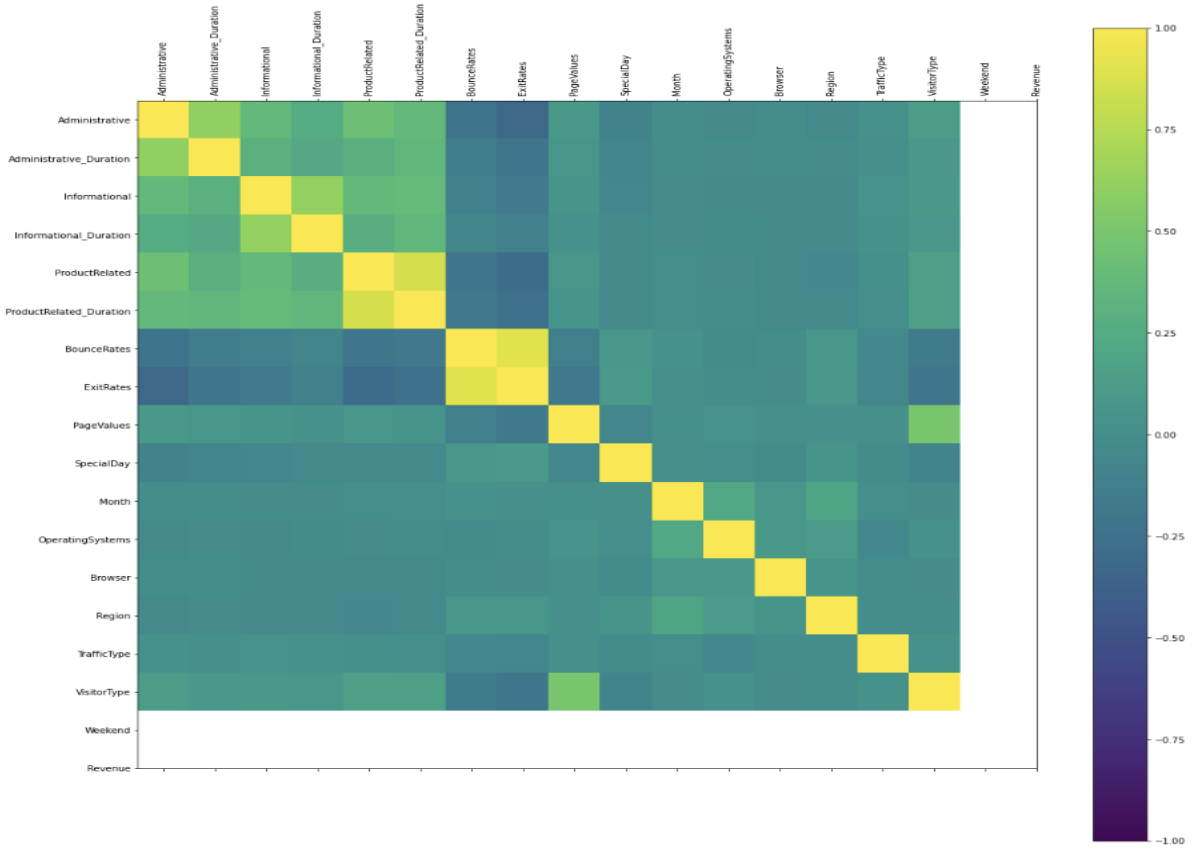
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	I
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	
mean	2.315166	80.818611	0.503569	34.472398	31.731468	
std	3.321784	176.779107	1.270156	140.749294	44.475503	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	7.000000	
50%	1.000000	7.500000	0.000000	0.000000	18.000000	
75%	4.000000	93.256250	0.000000	0.000000	38.000000	
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	

Using the newsdata.isnull().sum() we discovered that there were no missing values within the dataframe.

```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

Also using the shopsdata.shape function we were able to deduce the shape and size of the dataset which was (12330, 18). In order to visualize the data we used histograms and boxplots. This was done using the following functions, Histogram:

shopsdata.drop(['Revenue','Weekend','VisitorType','TrafficType','Region','Browser','OperatingSystems','Month'], axis=1).hist(bins=30, figsize=(12,12)) & Boxplot:shopsdata.drop(['Revenue','Weekend','VisitorType','TrafficType','Region','Browser','OperatingSystems','Month'], axis=1).plot(kind='box', subplots=True, layout=(4,4), sharex=False, sharey=False, figsize=(20,20), title='BoxPlot for each numeric input variable'). This allowed us to visualize the dataset to detect any irregularities that could have occurred such as, any outliers within the dataset. Lastly, like Dataset 1 a correlation analysis was done to determine the correlation between the different attributes within the dataset. This helped to identify the relationships between 2 attributes in order to deduce whether they had a strong correlation meaning they were related or weak correlation meaning they were not related.



Learning Methods

There were 3 learning methods that were utilized within this analysis, Removal of Non-Predictive Attributes, SKlearn SelectBest Function and Pandas. Cut for the first data set (‘Online_news_popularity’) and only 1 learning method for the second dataset (‘Online_shoppers_intention’) was used: Convert to Categorical to Numerical. Each learning method was tested using the Decision Tree Classifier, KNN Classifier and SVM Classifier. The three learning methods were also paired with the utilization of a confusion matrix in order to develop calculations for accuracy, precision, recall and F1-score. The confusion matrix was used to calculate True Positives, True Negatives, False Negatives and False Positives which was then used to calculate the accuracy of each algorithm. Although a confusion matrix was created using the line `print(confusion_matrix(y_test, pred))`, it was not needed as `print(classification_report(y_test, pred))` was run that had the ability to calculate all needed indicators for each learning method.

Method 1: Removal of Non-Predictive Attributes

```
from pandas.plotting import scatter_matrix
from matplotlib import cm
attribute_names = ['n_tokens_title', 'n_tokens_content', 'n_unique_tokens', 'n_non_stop_words', 'n_non_stop_unique_tokens', 'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_length', 'num_ke
x = newsdata[attribute_names] #Comparitive attributes without url timedelta and shares
y= newsdata['Sharescat'] #Target attribute is sharescat as it is converted from conitious variable to categorical
```

This method seeked to remove all non-predictive attributes from the data as non-predictive attributes have been known to not be helpful in creating models, such as unique keys: 'url', ' timedelta'. The removal of ‘shares’ was also performed as it was the target attribute and it would

result in a higher accuracy if it was implemented as a comparison attribute within itself, skewing the model.

Method 2: SKlearn SelectBest Function

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif, f_regression, mutual_info_regression, mutual_info_classif

selector = SelectKBest(f_classif, k=5)
selector.fit(x_train, y_train)

cols = selector.get_support(indices=True)
x_train = x_train.iloc[:,cols]
x_test = x_test.iloc[:,cols]
```

This method seeks to take two arrays x and y, and return a pair of arrays (scores, pvalues) or a single array with scores. The k parameter is important if you use selector.fit_transform(), which will return a new array where the feature set has been reduced to the best 'k'. K set to 5 meaning any selection has to be greater than 5 and dropping any attributes below it.

Method 3: Pandas.Cut

```
sharescategory = pd.cut(newsdata.shares,bins=[0,3395,843300],labels=['1','2'])
newsdata.insert(5,'Sharescat',sharescategory) #create new category with the cl
```

This method seeks to create bins for each continuous variable based on the mean and max of the data set. Each share is then placed into a class either 1 or 2, 1 for being below the mean threshold and 2 for being above the mean. The ranges for the bin are from 0 to the mean and from the

mean to the max, with each new share class being placed into a separate category called Sharescat. The new category, Sharescat is then used as the target attribute when splitting and normalizing the data.

Data Set 2

Method 1: Convert to Categorical to Numerical (Data Set 2: Online_Shoppers_Intention)

```
for col in ['Month','VisitorType']:
    shopsdata[col] = shopsdata[col].astype('category')

cat_columns = shopsdata.select_dtypes(['category']).columns
cat_columns

shopsdata[cat_columns] = shopsdata[cat_columns].apply(lambda x: x.cat.codes)
shopsdata['Weekend'] = shopsdata['Weekend'].astype(int)
shopsdata['Revenue'] = shopsdata['Revenue'].astype(int)
```

This method was applied on the second data set as some of the attributes were not in numerical format and thus could not be read by either of the three algorithms. These attributes were Month and VisitoType and needed to be converted into an integer. This was possible by using the variable “category” to select all categories that were not an integer by first applying `select_dtypes` then applying `.cat.codes.` Then converting the existing data under these two categories into integers by using .astype(int). As a side note only method was used for the second dataset as it yielded a high accuracy that is sufficient enough to rely on. This method was

discovered through Stack Overflow linked here:

https://stackoverflow.com/questions/32011359/convert-categorical-data-in-pandas-dataframe

Evaluation

Three methods were attempted for the first data set ('Online_news_popularity') and only 1 method was used in the second data set (Online_shoppers_intention). Measurements were used on the models as a benchmark to determine how effective they were in predicting the target class for each data set, Shares for data set 1 and Revenue for data set 2. The accuracy of each method was calculated using a confusion matrix in order to determine which method resulted in the best prediction. SVM Classifier, KNN Classifier and Decision Tree Classifier were used as predictive algorithms in order to gain accuracy and other indicators.

Method 1:

- In the first method ,only non predictive attributes were removed.When we tested the method using the test set it yielded an accuracy of 0.02 for the Decision Tree Classifier, 0.01 for the KNN Classifier and 0.05 for SVM Classifier. The training data set also yielded a low accuracy with it being 0.23 for KNN Classifier and 0.07 for SVM Classifier, with the only algorithm having a 100% training data accuracy being the Decision Tree Classifier. Interestingly, this method also yielded 0 for all categories in the confusion matrix, resulting in precision, recall and F1-score being 0. This method was clearly flawed as all the values of the confusion matrix resulted in 0.The run time for each algorithm was between 0-2 seconds, this seemed off so we have decided not include them individually.

Confusion Matrix for ALL Algorithms(Decision Tree, KNN) for Method 1

Prediction	NO	YES
NO	0	0
YES	0	0

Method 2:

In the second method, SKlearn SelectBest Function was used to create a model. This resulted in an accuracy of 0.00 for KNN, 0.06 for Decision Tree and 0.01 for SVM. The training data set accuracy was also 0.02 for SVM, 0 for KNN and 0.06 DescionTree, extremely low for all algorithms meaning something had to have gone with the method or Google's colab software is having trouble. Precision, F1-Score and Recall all resulted in 0, as the confusion matrix self had 0 for TP,FP,TN and FN. The run time for each algorithm was between 0-2 seconds, this seemed off so we have decided not include them individually.

Confusion Matrix For all Algorithms (DescionTree,KNN and SVM) for Method 2

Prediction	**NO**	**YES**
NO	0	0
YES	0	0

Method 3:

The final method for the first data set, which ultimately resulted in the best all around accuracy, precision, recall and F1-Score was through the use of pd.cut. This resulted in an accuracy of 0.7 for the Decision Tree Classifier, 0.78 for the KNN Classifier and 0.8 for the SVM Classifier using the test data set. The training data set also resulted in 1.00 for the Decision Tree Classifier, 0.83 for the KNN Classifier and 0.8 for the SVM

Classifier. Interestingly, the SVM classifier resulted in a confusion matrix that had 0 False Positives and 0 True Positives. The Decision Tree yielded a precision of 0.28, recall of 0.3 and a F1-Score of 0.29. The KNN Classifier resulted in a higher precision 0.37 but lower recall 0.15 and f1-score 0.21. The SVM classifier resulted in 0 for the tes data set for precision, recall, f1 score. This most likely has to do with Google colab and not the code as all the other algorithms worked and delivered similar accuracies. The KNN Classifier also had a run-time of 26 seconds, the Decision Tree Classifier a 28 second run-time and SVM with a 35 seconds run-time.

Decision Tree Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	9247	17
YES	3083	17

KNN Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	7406	498
YES	1709	298

SVM Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	7904	0
YES	2007	0

Decision Tree Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

	precision	recall	f1-score	support
1	0.82	0.80	0.81	7904
2	0.28	0.30	0.29	2007
accuracy			0.70	9911
macro avg	0.55	0.55	0.55	9911
weighted avg	0.71	0.70	0.70	9911

KNN Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

	precision	recall	f1-score	support
1	0.81	0.94	0.87	7904
2	0.37	0.15	0.21	2007
accuracy			0.78	9911
macro avg	0.59	0.54	0.54	9911
weighted avg	0.72	0.78	0.74	9911

SVM Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

1	0.80	1.00	0.89	7904
2	0.00	0.00	0.00	2007
accuracy			0.80	9911
macro avg	0.40	0.50	0.44	9911
weighted avg	0.64	0.80	0.71	9911

Data Set 2

Method 1:

This method is the only method used for the second dataset which is to Convert Categorical to Numerical using the `select_dtypes` then applying `.cat.codes`functions as it resulted in a high accuracy and good precision, recall and f-score indicators. Using this method it resulted in an accuracy of 0.86 for the Decision Tree Classifier, 0.84 for the KNN Classifier, and 0.86 for the SVM classifier using the test set. The accuracy for the decision Tree Classifier, KNN and SVM

Classifiers using the training set were 1.00, 0.89 and 0.89 respectively. All of these classifiers were close to optimal prediction standards and no further methods were needed as a result. The Decision Tree Classifier yielded a 0.58 in precision, 0.57 recall and 0.57 f1-score. The KNN Classifier yielded 0.62 precision, 0.23 recall and an f1-score of 0.33, lower than the indicators yielded from the DecisionTree Classifier. The SVM Classifier had a higher precision, 0.76, than both DescionTrere and KNN, however its recall,0.28 was lower than that of the Decision Tree Classifier and f1-score of 0.41. The run time for the Decision Tree Classifier was 22 seconds, the KNN Classifier being 29 seconds and the SVM Classifier being 31 seconds.

Decision Tree Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	2340	219
YES	225	299

KNN Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	2485	74
YES	404	120

SVM Classifier Confusion Matrix

Prediction	**NO**	**YES**
NO	2513	46
YES	375	149

Decision Tree Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

	precision	recall	f1-score	support
0	0.91	0.91	0.91	2559
1	0.58	0.57	0.57	524
accuracy			0.86	3083
macro avg	0.74	0.74	0.74	3083
weighted avg	0.86	0.86	0.86	3083

KNN Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

	precision	recall	f1-score	support
0	0.86	0.97	0.91	2559
1	0.62	0.23	0.33	524
accuracy			0.84	3083
macro avg	0.74	0.60	0.62	3083
weighted avg	0.82	0.84	0.81	3083

SVM Classifier (Precision, recall, f1-score and support) * 1 is train set 2 is the test set

	precision	recall	f1-score	support
0	0.87	0.98	0.92	2559
1	0.76	0.28	0.41	524
accuracy			0.86	3083
macro avg	0.82	0.63	0.67	3083
weighted avg	0.85	0.86	0.84	3083

Discussion

The methods that we used to conduct the analysis and evaluation of the Datasets helped us determine the target attributes for each Dataset. For Dataset 1, Method 1 - Removal of Non-Predictive Attributes and Method 2 - SKlearn SelectBest Function had various errors. For example, Method 1 had an accuracy of 0.02 for the Decision Tree Classifier, 0.01 for the KNN Classifier and 0.05 for SVM Classifier. The training data set also had a low accuracy of 0.23 for KNN Classifier and 0.07 for SVM Classifier. Therefore, it is clear this method was not the most suitable for Dataset 1. In addition Method 2, SKlearn SelectBest Function, had an accuracy of 0.00 for KNN, 0.06 for Decision Tree and 0.01 for SVM. The training data set accuracy was also 0.02 for SVM, 0 for KNN and 0.06 DescionTree. This accuracy rate was very low for all algorithms meaning it was not the most functional method to be used. Lastly, Method 3 Pandas.Cut for Dataset 1 and Method 1 for Dataset 2, were the most suitable methods to use as they had the highest accuracy rates for each of the algorithms (Decision Tree Classifier, KNN Classifier & SVM Classifier).

(‘Online_news_popularity’) and only 1 learning method for the second dataset (‘Online_shoppers_intention’)

Conclusion

This report was created in order to make a method that is both competitive in its accuracy and its run time. For the first data set, ('Online_news_popularity'), we believe that would be Method 3 using the KNN Classifier as it had a high accuracy and was one of higher precision. Although this method did not have the highest F1-score, it did have one of the fastest run-times amongst all other classifiers we conducted with an average time of 26 seconds. For the second data set, ('Online_shoppers_intention', we did not have any other methods to compare it with although we did have a classifier that was objectively the best choice. In our method, the Decision Tree Classifier was the best choice as it had the highest F1-Score, although its accuracy was tied at 0.86 with the SVM Classifier its F1-Score, Precision and Recall were all higher than that of SCM including the run time, with an average of 22 seconds.

Sources:

https://stackoverflow.com/questions/32011359/convert-categorical-data-in-pandas-dataframe
https://stats.stackexchange.com/questions/146294/what-is-misclassification-rate-how-do-we-calculate-it
https://www.absentdata.com/pandas/pandas-cut-continuous-to-categorical/
https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest'
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
https://stackoverflow.com/questions/34561850/what-does-it-mean-predictive-attribute#:~:text=Predictive%20attribute%20are%20attributes%20that,usually%20fall%20into%20this%20category
https://www.youtube.com/watch?v=J2gz0mbvg78&ab_channel=soumilshah1995
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.qcut.html#pandas.qcut
https://stackoverflow.com/questions/47168693/looking-to-transform-continuous-variables-into-categorical
https://www.absentdata.com/pandas/pandas-cut-continuous-to-categorical/
https://stackoverflow.com/questions/30746460/how-to-interpret-scikits-learn-confusion-matrix-and-classification-report
https://stats.stackexchange.com/questions/323154/precision-vs-recall-acceptable-limits
https://stats.stackexchange.com/questions/108964/high-precision-with-low-recall-svm
https://stackoverflow.com/questions/32011359/convert-categorical-data-in-pandas-dataframe
https://towardsdatascience.com/data-exploration-101-with-pandas-e059d0661313