Lewis Francisco Handy-Cardenas, Matrix Methods for Machine Learning, ECE532, Project Update 2.

Project Update 2 - "Online Shopper Intention Classification"

Data preprocessing [Revised & Finished]

The pre-processing of the data was rethought and revisited to be able to appropriately transform all of the data types to numeric types as it was found that some of the string values found in some of the attributes were object and Boolean type. The pandas and sklearn libraries were used to facilitate this procedure and facilitate the error analysis. The numpy libraries were still used to perform the different operations. The following tables show the data types of each of the 18 attributes before (left table) and after (right table) the preprocessing steps.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12316 non-null  float64
 1   Administrative_Duration  12316 non-null  float64
 2   Informational            12316 non-null  float64
 3   Informational_Duration   12316 non-null  float64
 4   ProductRelated           12316 non-null  float64
 5   ProductRelated_Duration  12316 non-null  float64
 6   BounceRates              12316 non-null  float64
 7   ExitRates                12316 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(10), int64(4), object(2)
memory usage: 1.5+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12316 entries, 0 to 12315
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12316 non-null  float64
 1   Administrative_Duration  12316 non-null  float64
 2   Informational            12316 non-null  float64
 3   Informational_Duration   12316 non-null  float64
 4   ProductRelated           12316 non-null  float64
 5   ProductRelated_Duration  12316 non-null  float64
 6   BounceRates              12316 non-null  float64
 7   ExitRates                12316 non-null  float64
 8   PageValues               12316 non-null  float64
 9   SpecialDay               12316 non-null  float64
 10  Month                    12316 non-null  int64
 11  OperatingSystems         12316 non-null  int64
 12  Browser                  12316 non-null  int64
 13  Region                   12316 non-null  int64
 14  TrafficType              12316 non-null  int64
 15  Weekend                  12316 non-null  int64
 16  Revenue                  12316 non-null  int64
 17  V_New_Visitor            12316 non-null  uint8
 18  V_Other                  12316 non-null  uint8
 19  V_Returning_Visitor      12316 non-null  uint8
dtypes: float64(10), int64(7), uint8(3)
memory usage: 1.6 MB
```

During the preprocessing step, it was also found that from the 12330 rows, some were missing entries. These were sought out and extracted from the data; the resulting data from this step left 12316 rows which were used to design the classifiers.

The data was then separated into training data and evaluation data. The current split value is 80%, meaning this is the percentage of the raw data that is withheld for use in training data, and the remaining 20% is used to evaluate the obtained weight vectors. Cross-validation methods will be used to iterate around different withheld subsets of data to determine the optimum value to use for the data splitting. Keep in mind that for this data, the "revenue" category was used as labels.

The sparsity given in some of the values of these attributes could be conflicting with some of the machine learning techniques that are being implemented, it could well be that these attributes are not contributing a lot into the weight vector, diminishing their importance to be included in the predicting model to be developed.

Classifier 1 - Linear regression: Least squares classifier

The implementation of the least squares classifier utilizing the training data set has been fully implemented. The weights calculated from this are no longer affected by the size of the training set, which used to be a problem. The following have been found to be the optimal weight vector values:

Training & evaluating - Least Squares

In [15]:
```
# Classifier 1 - Training Data
#w = (X^T X)^(-1)X^T y
X = X_train
y = y_train
w_train = np.linalg.inv(X.transpose()@X)@X.transpose()@y
#A = np.linalg.inv(X@X.T)

print(np.round(w_train,2))
```

```
[ 0.    0.    0.01 -0.01  0.01  0.02  0.02 -0.05  0.17 -0.01  0.02 -0.01
  0.   -0.    0.    0.    0.   -0.01 -0.02]
```

The weight vector still needs to be validated by running cross-validation techniques on the available dataset, in order to determine if the current weight vector is the best one for the model.

Classifier 2 - Linear regression: Truncated SVD

Truncated SVD decomposition has been also implemented in order to evaluate a set of weights which could also be used to classify data. This classifier was implemented in place of the previously proposed "Page Rank Algorithm". This latter one is currently being evaluated and still considered for implementation, but the applicability to the type of classification problem we are dealing with here does not seem to fit easily with the way this algorithm works; which has made the implementation difficult. As an alternative, Truncated SVD has been selected and implemented in case the "Page Rank Algorithm" implementation doesn't work. The truncated SVD later be combined with PCA in order to provide a better and solid 2nd classifier to evaluate the data with. The following have been found to be the optimal weight vector values:

```
In [27]: #w = VT.T@np.diag(1/s)@U.T@y_train
         w_svd = VT.T@np.diag(1/s)@U.T@y_train
         print(np.round(w_svd,2))

         [-1.00000000e-02  1.00000000e-02  1.00000000e-02  0.00000000e+00
           2.00000000e-02  1.00000000e-02  2.00000000e-02 -5.00000000e-02
           1.60000000e-01 -2.00000000e-02  2.00000000e-02 -0.00000000e+00
           0.00000000e+00 -1.00000000e-02 -0.00000000e+00 -0.00000000e+00
           3.78107592e+13  9.08857910e+12  3.85924197e+13]
```

Performance with current classifiers

Both the linear regression and Truncated SVD's classifier performance has not been very good with predicting the correct labels for the evaluation data. The mean squared error of the testing set has been found to be around 95%, and slightly better for the SVD. This value is not very good and this might still be a function of the sparsity of the data for some of the attributes. The evaluation will continue with both of these classifiers to identify the reason the performance is so bad. Some statistic evaluation on the raw data will be added to the preprocessing step to identify the impact of the sparsity.

Timeline evaluation

The project is still running a little bit delayed with respect to what was projected with the Gant diagram. The pre-processing step has been fully developed and complete and the design of the third classifier, the Neural Networks, is being worked on. There is a strong feeling and hope for this third classifier to outperform the other two.

At this stage of the project there are some indicators that show the Page Rank Algorithm selection may not be appropriate given the classification problem at hand; once evaluation of the applicability is complete it will be determined if a new algorithmic approach must be included outside of these because of the structure of the dataset. For now, a counter suggestion would be Truncated SVD; which can later be combined with PCA to design a better classifier.

Project Github

A link to the project files (including updates) can be found under the following path:

https://github.com/handycardena/ECE532_Final_Project_Handy

https://github.com/handycardena

Timeline for the progress of the project

Important dates:

October 22 → Delivery of Proposal

November 17th → First update

December 1st → Second update

December 12th → Delivery

The following Gantt diagram breaks down the project goals for the project development timeline.