Predicting Purchase Intention

Will browsing sessions on an e-commerce site convert to purchases for online shoppers?

IE 7374 ST Machine Learning in Engineering
Group 8

Outline

- Introduction
- Data Source and Description
- Data Exploration
- Data Preprocessing
- Model fitting and Results
- Discussion
- Conclusion

Introduction

- **Problem Setting**

  The increased e-commerce usage has generated great potential in the market over the past few years. It is crucial for companies to devise smart marketing and promotion strategies to retain as many shoppers as possible when they are browsing e-commerce sites

- **Project Objective**

  Our team is interested in predicting online shoppers' purchasing intentions. What will influence a shopper's decision to make purchases? Are there any other interesting facts to be discovered?

  In order to answer these questions, we will create several machine learning models to predict if browsing sessions of shoppers will successfully convert to purchases.

  We hope that insights from implementing these models will help us to understand a little more about online shoppers' behavior and potentially shed some light on increasing profits for businesses and improving e-commerce site quality.

Data Source and Description

- Data Source: UCI Machine Learning Repository
- Dataset Description:
  - Browsing sessions data collected from different users who browses an e-commerce site in a 1-year period.
  - This dataset contains 12330 observations and 18 attributes (including one target variable). Among these 17 predictors, there are 10 numerical and 7 categorical variables

```
df = pd.read_csv('online_shoppers_intention.csv')
```

```
df.head()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	S
0	0	0.0	0	0.0	1	0.000000	0.20	0.20	0.0	
1	0	0.0	0	0.0	2	64.000000	0.00	0.10	0.0	
2	0	0.0	0	0.0	1	0.000000	0.20	0.20	0.0	
3	0	0.0	0	0.0	2	2.666667	0.05	0.14	0.0	
4	0	0.0	0	0.0	10	627.500000	0.02	0.05	0.0	

```
df.columns
```

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

Description of Variables

Categorical

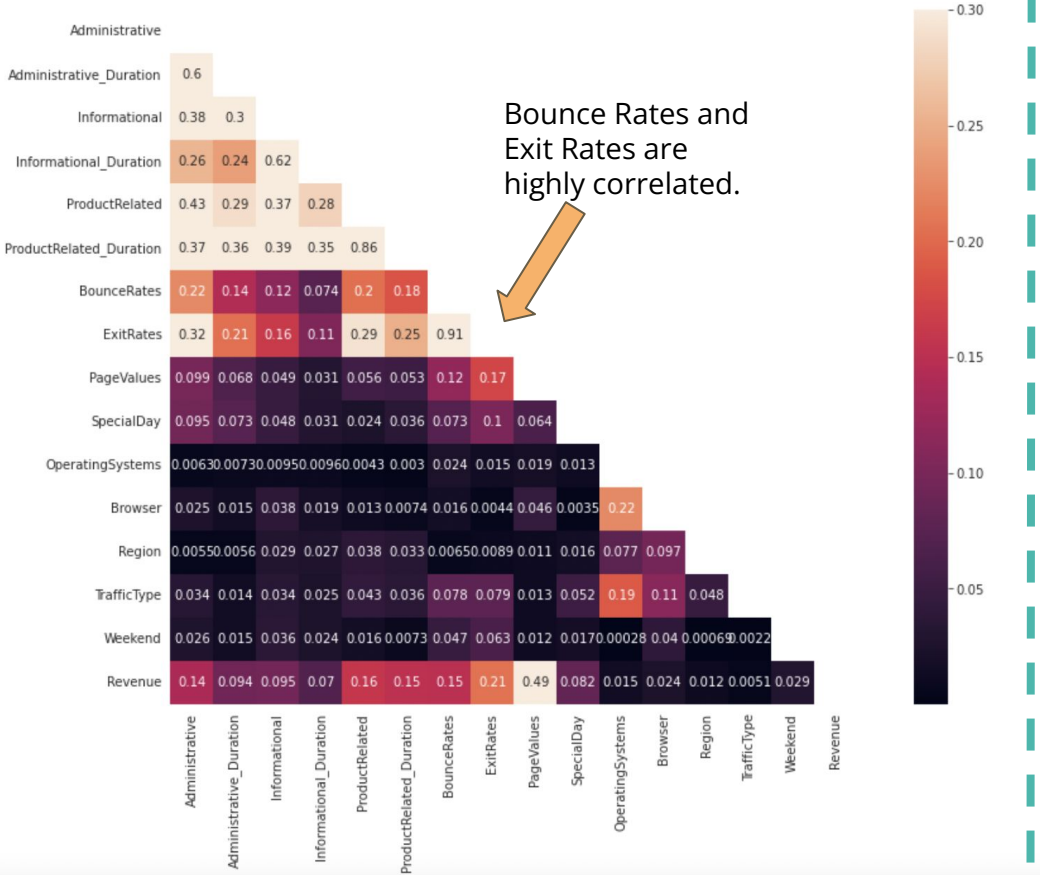
Numerical

Feature name	Feature description	Number of categorical values
OperatingSystems	Operating system of the visitor	8
Browser	Browser of the visitor	13
Region	Geographic region from which the session has been started by the visitor	9
TrafficType	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)	20
VisitorType	Visitor type as "New Visitor," "Returning Visitor," and "Other"	3
Weekend	Boolean value indicating whether the date of the visit is weekend	2
Month	Month value of the visit date	12
Revenue	Class label indicating whether the visit has been finalized with a transaction	2

Feature name	Feature description	Min. value	Max. value	SD
Administrative	Number of pages visited by the visitor about account management	0	27	3.32
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management related pages	0	3398	176.70
Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site	0	24	1.26
Informational duration	Total amount of time (in seconds) spent by the visitor on informational pages	0	2549	140.64
Product related	Number of pages visited by visitor about product related pages	0	705	44.45
Product related duration	Total amount of time (in seconds) spent by the visitor on product related pages	0	63,973	1912.25
Bounce rate	Average bounce rate value of the pages visited by the visitor	0	0.2	0.04
Exit rate	Average exit rate value of the pages visited by the visitor	0	0.2	0.05
Page value	Average page value of the pages visited by the visitor	0	361	18.55
Special day	Closeness of the site visiting time to a special day	0	1.0	0.19

Note: variable summary taken from the original paper

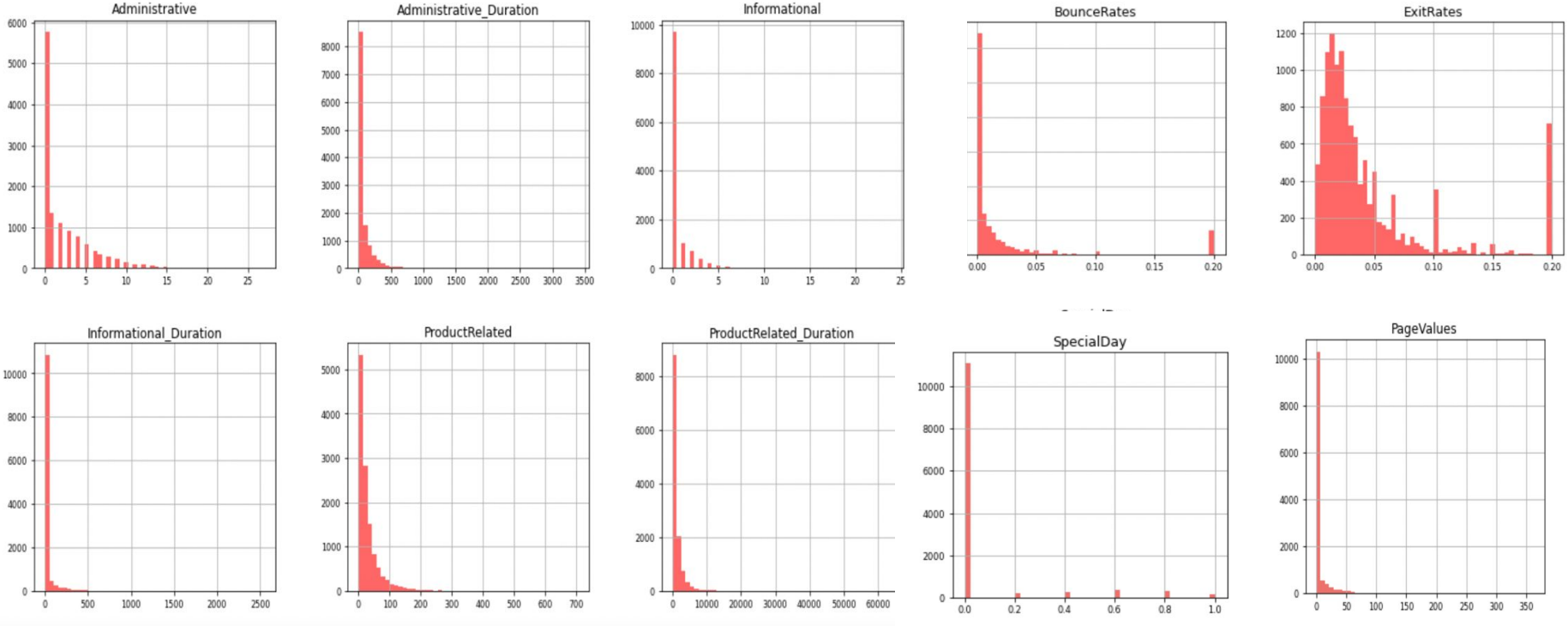
Data Exploration - Correlation



- Exiting a single-page session counts as both a bounce and an exit since the user exited the first page they landed on when entering the website.
- For multi-pages sessions, intermediate bounces and one exit are counted.

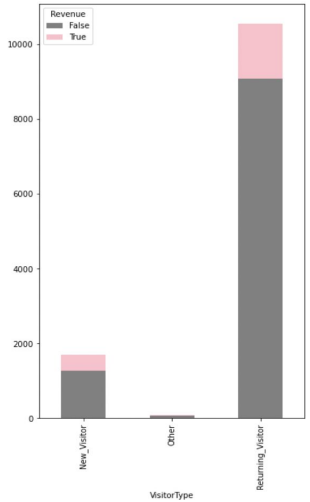
Data Visualization - Histograms

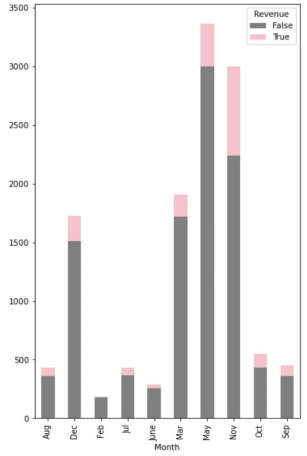
From the plot below, we can see that all the numerical variables have skewed distributions.

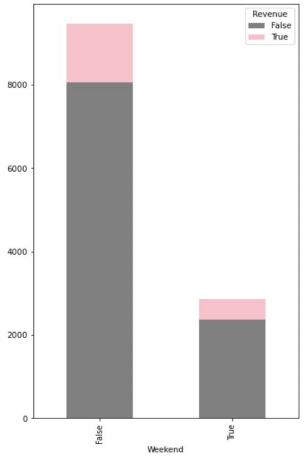


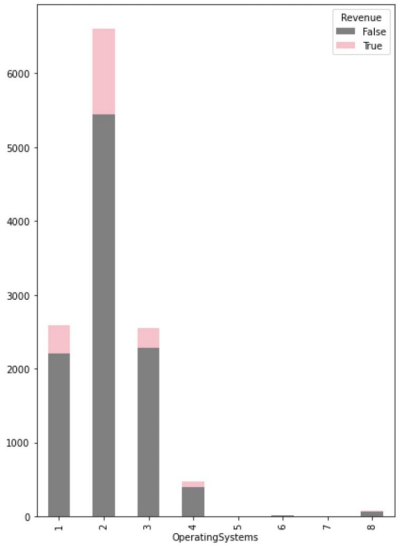
Data Visualization - Stacked bar chart

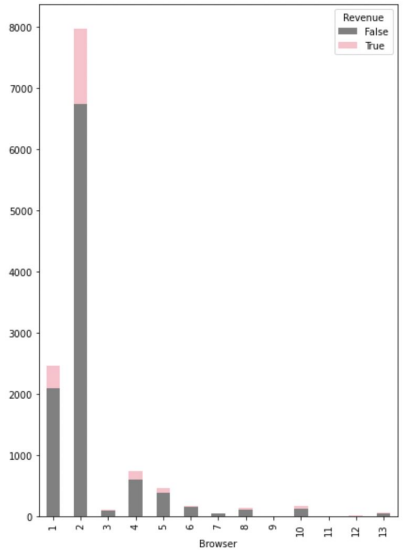
- Lots of purchases were made in March, May, November and December when the holiday seasons approach.
- Most of the purchases were made by returning customers, and Browser type 1 & 2 seem to be extremely common for shoppers.
- Rare categories in OS, Browser and traffic type.

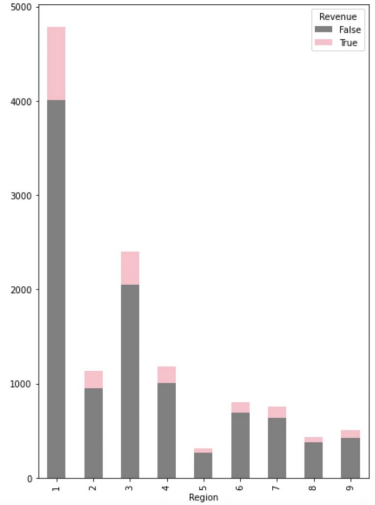


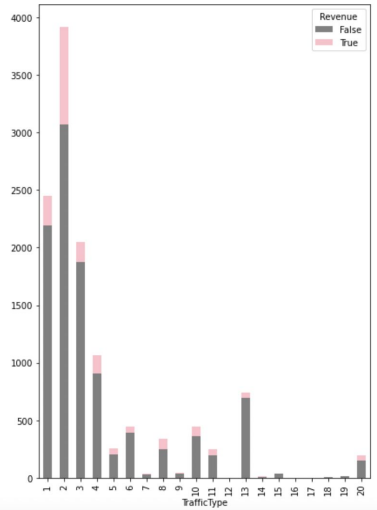






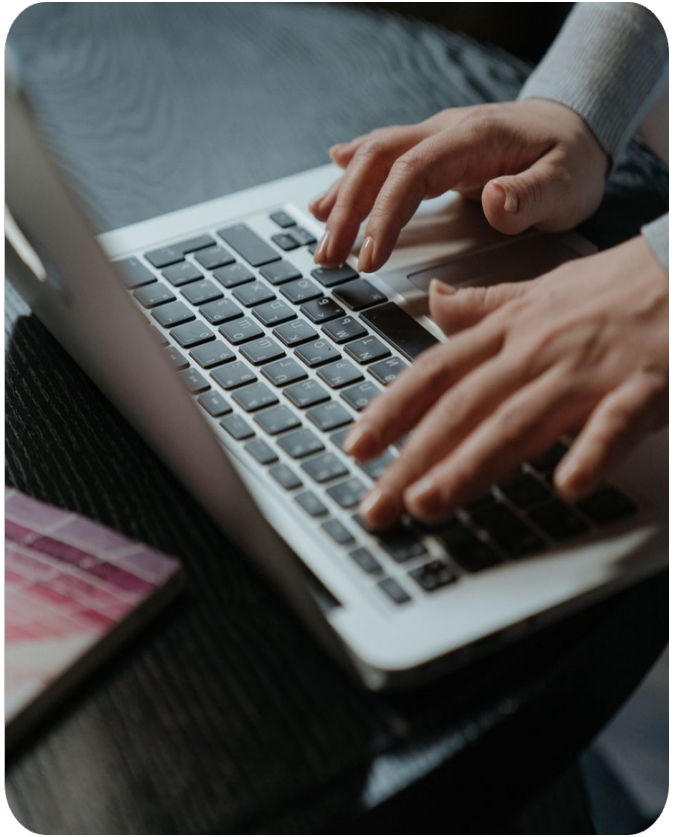






Other Observations

- Imbalanced data: only 15.5% (1908 observations) of the data belongs to the success class (“Revenue = Yes” a.k.a. purchase made)
- Type conversion needed for some variables: “Weekend” and the target variable “Revenue” are stored as boolean variables. “Traffic type” not stored as categorical data.



Data Preprocessing

- Normalize numerical variables and create dummies for categorical data
- Binning and dropping rare categories
  - “Traffic Type”, “Operating System” and “Browser” have categories with very limited presence. Binning these categories will reduce the dimension of data when fitting models.
  - “Other” visitor type not only has low presence, but also doesn’t really make sense logically since a visitor is either new or returning. Drop these category before.
  - In total, 5 OS types, 10 Browser types, 5 Regions and 10 Traffic types were binned into “other” category. 30+1 = 31 columns were binned into 4, reducing our input space dimension from 75 to 48.

Data Preprocessing

- Oversampling using SMOTE
  - Using Synthetic Minority Oversampling Technique provided by “imblearn” library to generate synthetic data for minority class based on the nearest neighbors . This method keeps the volume of our original dataset without adding additional information. It will help the model to learn the the success class (purchase conversions).
  - After oversampling, the training set ends up with 14546 (compared to original 12330) records, each of the classes accounts for half of them (7273)

Model Selection

Based on our dataset and our task of classification, we selected the following models to be trained. Most of the models used 70% of the data as training set and the rest 30% as testing set.

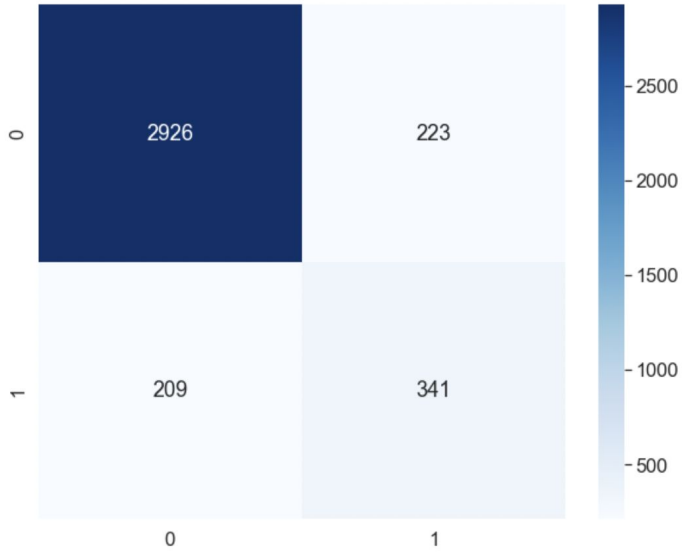
- Logistic Regression
- K-Nearest Neighbors
- Soft Margin Support Vector Machine
- Feed-forward neural networks

Logistic Regression

- Intuition
  - Baseline model for performance evaluation
  - Works with both numerical and categorical variables
  - Easy interpretation and implementation
- Additional Preprocessing Step
  - Drop “ExitRate”: “Bounce Rate” seems to be a better candidate if we are only keeping one of the highly correlated variable. As explained previously, some exits counts as bounces too. It encompasses more information about how the shoppers navigate between pages

Logistic Regression - Results

- Learning rate = 1e-4, tolerance = 0.001
- Test set - Accuracy 87.8%, Recall 63.1%, Precision 58%
- Relatively well in classifying class 1



Confusion Matrix

Training set:

```
Accuracy is 0.8880791970301114
Recall is 0.8446308263440121
precision is  0.9250112934798976
```

Testing set:

```
pred = lg_clf.predict(X_test)
```

```
performance(y_test, pred)
```

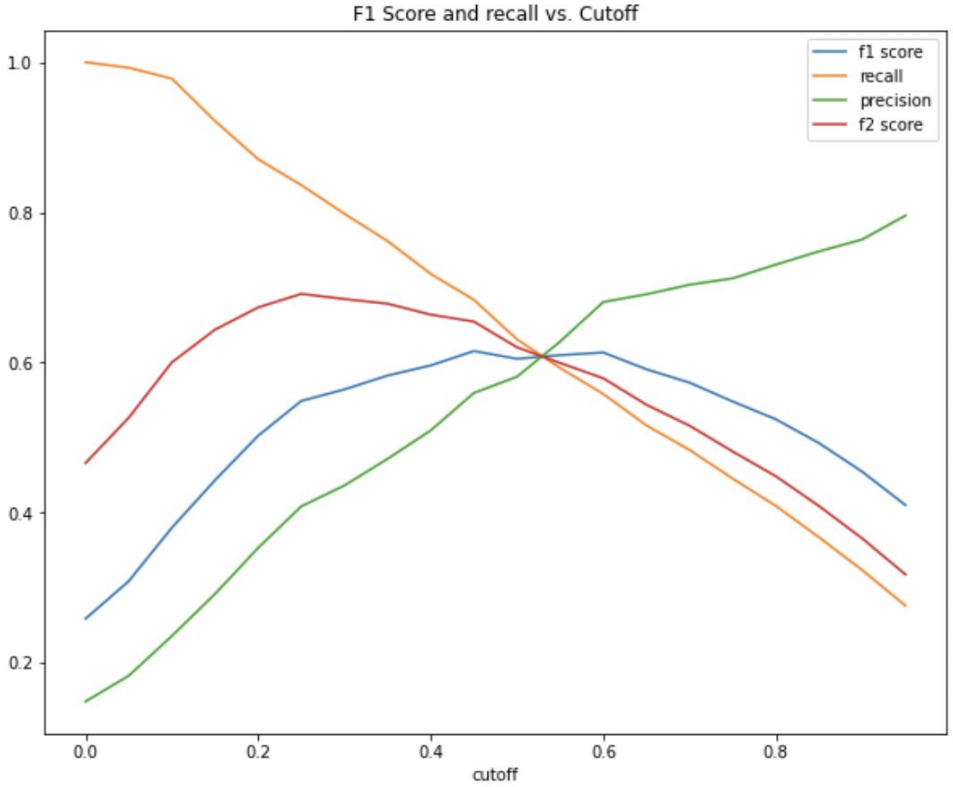
```
Recall is 0.6309090909090909
Precision is 0.5812395309882747
F-1 score is 0.6050566695727986
Overall accuracy is 0.8775344687753447
```

Logistic Regression - coefficients

Month_Feb	-4.049157
TrafficType_13	-2.983076
Month_June	-2.394001
Region_7	-2.359948
otherRegion	-2.320367
TrafficType_6	-2.290649
Region_4	-2.260385
TrafficType_3	-2.188079
Month_Mar	-2.170451
Region_6	-2.164988
Region_2	-2.073493
Month_Dec	-2.064279
TrafficType_10	-2.062738
Month_Aug	-2.011668
Month_May	-2.002203
Region_3	-1.928332
TrafficType_1	-1.834478
Region_1	-1.810210
otherTrafficType1	-1.802431
TrafficType_5	-1.786472
Month_Jul	-1.702430
Month_Oct	-1.698993
Month_Sep	-1.655042
TrafficType_4	-1.627035
TrafficType_11	-1.589628
TrafficType_8	-1.456980
TrafficType_2	-1.271567
BounceRates	-1.029606
Month_Nov	-0.620790
operOthers	-0.157635
Weekend	-0.138200
SpecialDay	-0.112807
Informational	-0.043960
Administrative_Duration	-0.017641
Informational_Duration	-0.007121
Administrative	0.010890
ProductRelated_Duration	0.085976
Browser_4	0.094247
ProductRelated	0.094288
OperatingSystems_3	0.162249
OperatingSystems_1	0.247887
otherBrowser	0.261452
OperatingSystems_2	0.491392
Browser_2	0.536669
Browser_1	0.595442
PageValues	2.254421
VisitorType_New_Visitor	3.115436
VisitorType_Returning_Visitor	3.196757

- Visitor Type, Page values, some values of month, traffic and region have strong presence in the model.
- If the browsing session was in February, it's about 4 times less unlikely to convert to purchase for online shoppers compared to browsing sessions in other months.
- The page value measured by Google Analytics also influenced our target variable.
- Being a returning visitor also seems to be crucial in successful purchases
- Surprisingly, weekend browsings sessions and closeness to special day doesn't have strong influence

Logistic Regression - Sensitivity analysis




```
maximum f1 score  0.6153846153846154 reached at cutoff of  0.45
```

- Optimal cutoff point 0.45 seems to occur near the default cutoff
- F1 score around 0.6. Relatively acceptable performance in classifying the class of interest.

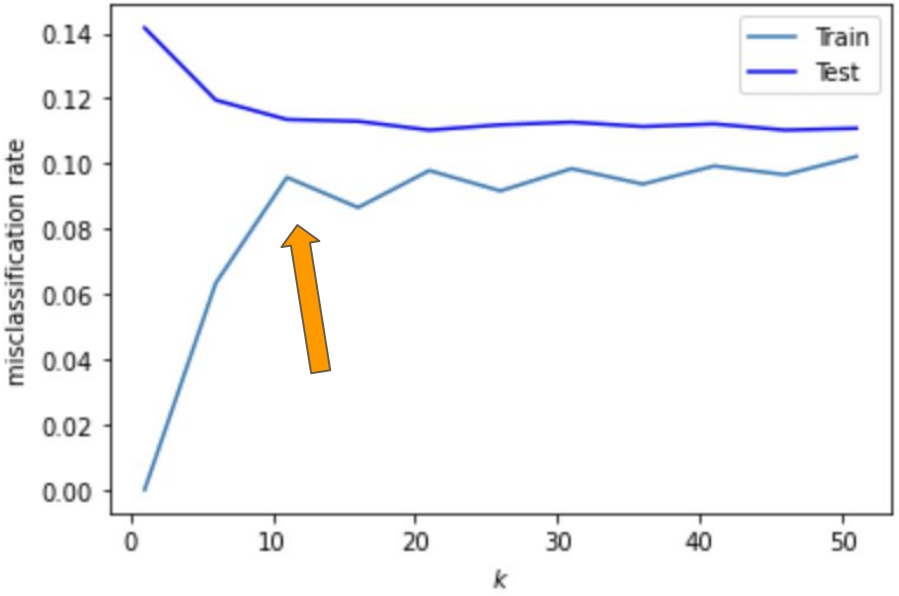
K-Nearest Neighbor

- Intuition: “Lazy learner” - simple but very versatile. Its base on the similarity between data points and can potentially help us catch similar characteristics among online shoppers.
- Initial results using k = 50, and Euclidean distance. Similar result in performance metrics compared to LR.

```
1
2 print('Accuracy:',metrics.accuracy_score(y_pred,y_test))
3 print('F1 Score:',metrics.f1_score(y_pred,y_test))
4 print('Recall:',metrics.recall_score(y_pred,y_test))
```

```
Accuracy: 0.8888888888888888
F1 Score: 0.6155285313376988
Recall: 0.6339113680154143
```

KNN - Tuning for K



- Finding a K value that smooths out the misclassification rates for training and testing data.
- Test k value from 1 - 50, increment by 5. From the graph, we can see that when k = 10 misclassification rate of testing set starts smooth out and training errors fluctuate
- K =10 obtains similar performance with K = 50. Choose smaller number for faster calculation

Soft Margin SVM

- Intuition
  - Kernels could be used to project and separate data into higher dimensions
  - Once training is done, making predictions are really fast since we only need several support vector points.
- Fitting the model
  - For faster training process, around 15% of the data is used for training and all the rest to testing set.
  - PCA could be applied (more details to come)

SVM - Kernel & PCA

- RBF Kernel
  - Radial basis function kernel is popular kernel function used in various machine learning algorithm, which has following form:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2\sigma^2}\right)$$

  - In our codes, we use gamma(γ) instead of σ, the relationship of γ and σ is:

$$\frac{2\gamma}{n_{feature}} = \frac{1}{2\sigma^2}$$

SVM - Kernel & PCA

- PCA
  - ❏ Since we will apply kernel, we can't directly obtain a linear function(w cannot be calculated), making it hard to interpret the model. Using PCA would not exert extra problem of losing explanation ability.
  - ❏ PCA filters noise by dropping the last few PCs which contain most of noise and increases model performance.
  - ❏ The PCA solves high correlation between variables by projecting data into perpendicular axes, reducing the number of predictors.

44	Month_Sep
45	operOthers
46	otherBrowser
47	otherRegion
48	otherTrafficType1

PC32	0.886946
PC33	0.904513
PC34	0.921239
PC35	0.937515
PC36	0.952937

SVM - Modeling

- Modified our model to use GridSearchCV in sklearn, we set cv = 3 and use only 5% to search best hyperparameters(C =1, γ = 0.53).
- Compare performance with oversampled training and without.
- Pass these hyperparameters into our model, trained with 15% data, we have the result:

```
Recall is 0.5473333333333333
Precision is 0.7005119453924915
F-1 score is 0.6145209580838323
Overall accuracy is 0.8955798864557989
```

Revenue	-1	1
Prediction		
-1	8013	679
1	351	821

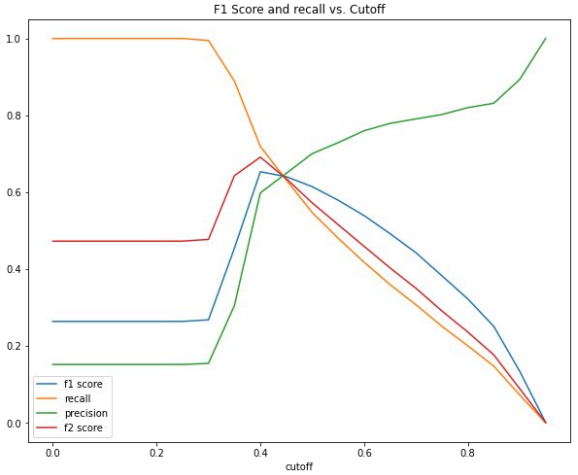
```
Recall is 0.7256857855361596
Precision is 0.5467355565993424
F-1 score is 0.6236271095633538
Overall accuracy is 0.8659479057341857
```

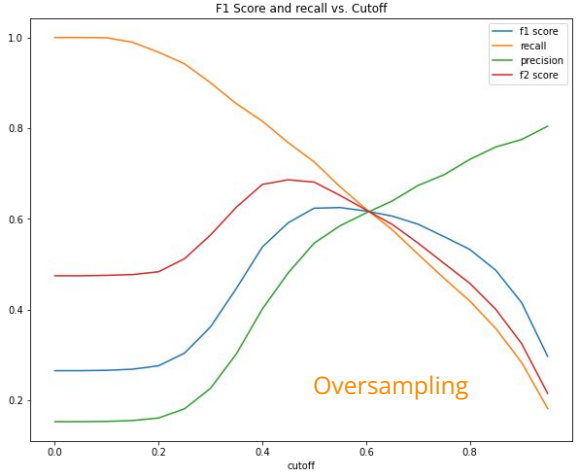
Revenue	-1	1
Prediction		
-1	7912	440
1	965	1164

Oversampling

SVM - Sensitivity Analysis

- The model without SMOTE is under-confident. We can see the metrics do not change when cut-off in [0, 0.2], which means the model isn't confident enough to help us divide the data.
- It seems that oversampling is necessary and the resulting recall and F-1 score look promising.
- Overall, although we faced challenges in tuning the model, SVM is powerful classifier if when we only used a portion of records.





Neural Net: Initial Model

Intuition: Capturing complex relationships between predictors

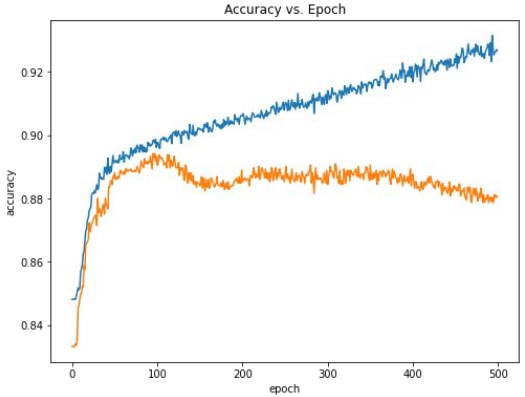
Model Architecture:

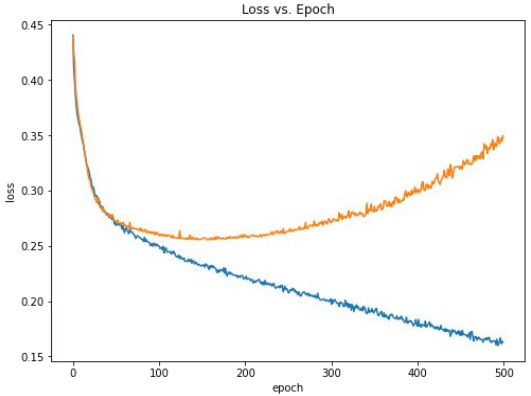
- Fully connected NN
- Two hidden layers
  - 64, 32 nodes respectively
  - Activation Function: ReLU
  - Output Layer: Softmax
- Optimizer: Adam
- Additional Dropout layer

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	4800
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 2)	66

Total params: 6,946
Trainable params: 6,946
Non-trainable params: 0

Neural Net: Initial Model Result





The best result occurred at around 100 epochs, so we stopped there and log the results:

	precision	recall	f1-score	support
0	0.90	0.96	0.93	2055
1	0.74	0.49	0.59	411
accuracy			0.89	2466
macro avg	0.82	0.73	0.76	2466
weighted avg	0.88	0.89	0.88	2466

Neural Net: H

- Tuner: Hyperb
  - A variant of
  - Successive h
  - Pause-and-r
- Search space:
  - # of hidden
  - # of node ea
  - Activation Fu
  - Learning rat
  - Best epoch
- Early Stop dep

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 384)	28800
dense_6 (Dense)	(None, 128)	49280
dense_7 (Dense)	(None, 480)	61920
dense_8 (Dense)	(None, 192)	92352
dense_9 (Dense)	(None, 32)	6176
dense_10 (Dense)	(None, 384)	12672
dense_11 (Dense)	(None, 416)	160160
dropout_1 (Dropout)	(None, 416)	0
dense_12 (Dense)	(None, 2)	834

Total params: 412,194
Trainable params: 412,194
Non-trainable params: 0

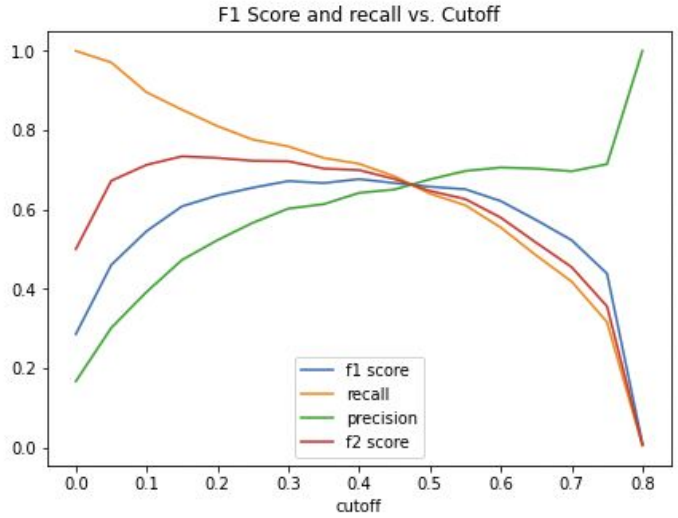
```
en layers

in_value=32, max_value=512, step
itializer', ['uniform', 'normal'
', ['relu', 'sigmoid', 'tanh']))

ializer', ['uniform', 'normal'])
e-3, 1e-4])

ng_rate),
```

Neural Net: Sensitivity Analysis




```
maximum f1 score   0.6766398158803222 reached at cutoff of   0.4
maximum f2 score   0.7340604026845639 reached at cutoff of   0.15
```

Pick 0.4 as final threshold:

	precision	recall	f1-score	support
0	0.94	0.92	0.93	2055
1	0.64	0.72	0.68	411
accuracy			0.89	2466
macro avg	0.79	0.82	0.80	2466
weighted avg	0.89	0.89	0.89	2466

Recall significantly improved from 49% to 72% without losing accuracy! 👍

Discussion

- These 4 models have similar accuracy, however, SVM with oversampling technique, and the tuned NN model with 0.4 as cutoff value, both achieve high recall.
- Depending on the business setting, if we need to obtain fast result and retain explainability, we can resort to logistic regression model as it's simpler to explain, and much faster in training.
- If allocated more resources and timeframe to train on greater volume of data, adopting more complex models like SVM and neural networks model have great potentials since limited amount of data already perform well, especially in classifying success class

Challenges and Future Improvements

- Very time consuming to train and obtain results from SVM using full dataset to gauge its performance before we could do further explorations and tuning parameters. Only a small portion of the data is used to fit SVM model.
- Due to the nature of our original dataset, some of the variables lack detailed information in their categories. For example, we didn't know what exactly the 20 traffic types and the 8 operating systems are. This information is stated neither on the UCI website nor in the original paper.
- Outside the context of this course, more ML packages and resources could be explored to train new versions of SVM or NN models.

Thank you!

Group 8
Chushi Chen, Ziwei Duan, Binghui Lai, Weikun Zhuang