Predicting Purchases

Presented by:
Karan Jain

PageValues is the most important feature in predicting purchases.

What is Predicting Purchase?

1. Everytime a customer shops, they either buy something or they don't.
2. During these sessions, customer's activity is tracked.

Given past shopping activity, predict a (new) shopping trip's outcome.

Why Predict Future Purchases?

1. Methods of predicting purchases can inform why a purchase was made.
2. This information can help intervention - Convert ‘sessions’ to purchases.
    a. Realtime
        i. Send offer(s) to user
        ii. Send price changes to user
        iii. Send reminder to user
    b. Longterm
        i. Improve interfaces - Website, App
        ii. Improve Services
        iii. Improve Product
3. Sell more products by informing where to display recommended products.
4. Personalize ‘experiences’ to individual customers

To improve bottom line of business - More profit.

How to Predict Future Purchases?

Build Predictive Models.

Following models were built:

1.	Random Forest Classifier
2.	K Nearest Neighbor
3.	Logistic Regression - Baseline Model

With 90% Accuracy, Random Forest Classifier performed the best.

Performance Metric - Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Optimizing for Accuracy automatically optimizes for better F1, Precision, Recall

Dataset

1. Dataset has data points for 12330 shopping visits.
2. Dataset has 10 numerical, 7 categorical features and one target variable.
3. Various train:validation:test splits were tried before arriving at 60:20:20.
4. train:validation:test splits were stratified.
5. Final classifier was selected by comparing performance on Validation Set.
6. If accuraries were tied, classifier with best F1 Score was picked.

Features Used

A+I+PR	numerical
Ad+Id+PRd	numerical
BounceRates	numerical
PageValues	numerical
SpecialDay	numerical
Month	categorical
OperatingSystems	categorical
Browser	categorical
Region	categorical
TrafficType	categorical
VisitorType	categorical
Weekend	categorical
ExitRates	numerical

Legend
Generated
Original
Deleted

A+I+PR, Ad+Id+PRd are measures of time spent on website.

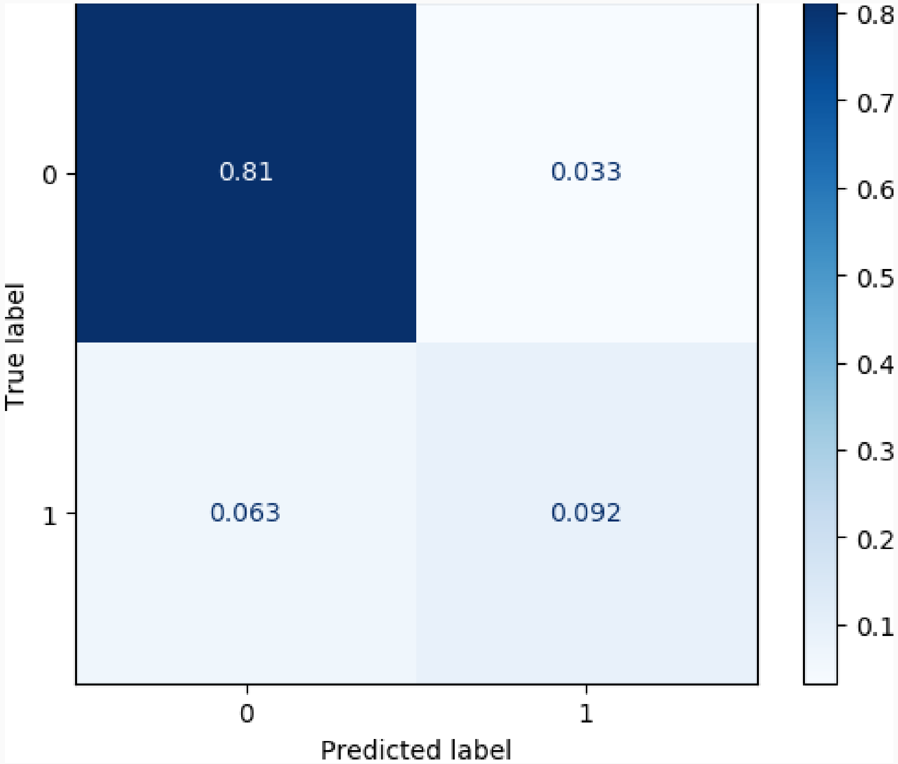
Random Forest Classifier - Model Specification

All combinations of following parameters were Cross Validated.

```
parameters = {
            'class_weight': [None, 'balanced', 'balanced_subsample'],
            'n_estimators': [5, 50, 75, 100],
            'max_depth': [2, 10, 20, None]
        }
```

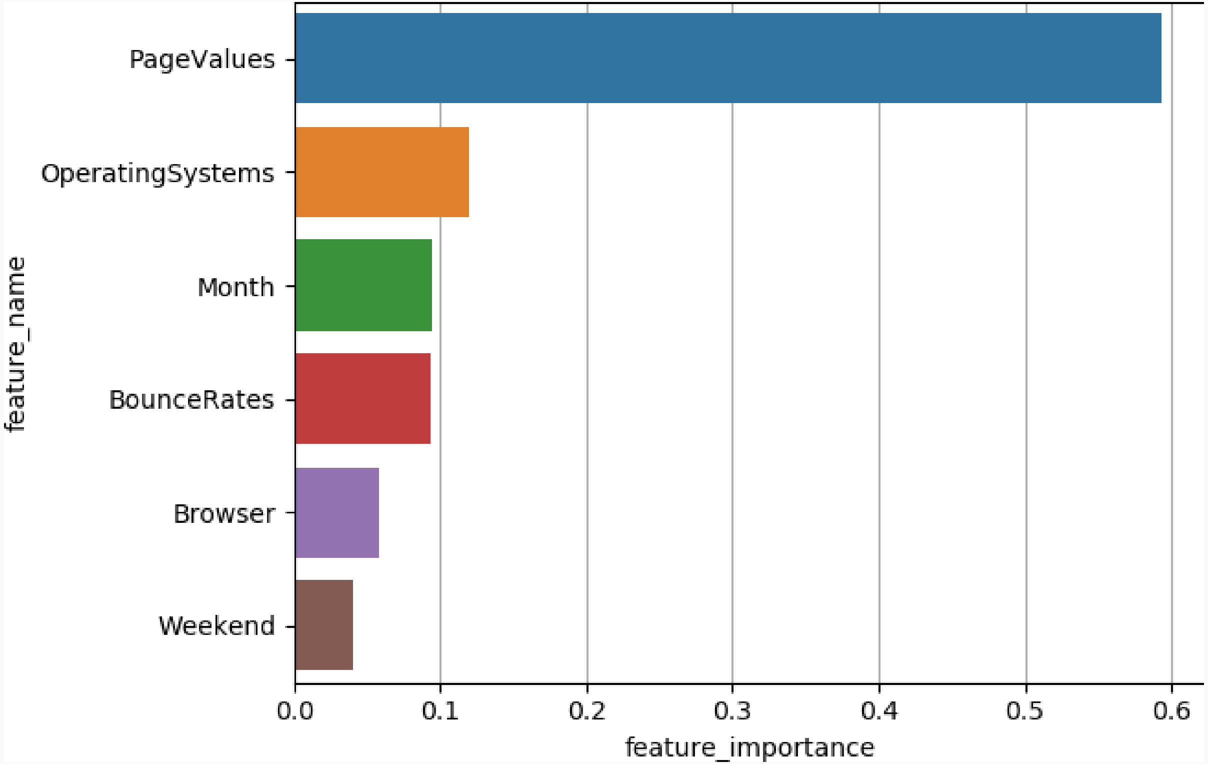
Highest CV accuracy of 90% for 'class_weight': None, 'max_depth': 10, 'n_estimators': 75

Random Forest Classifier - Performance - Validation Set



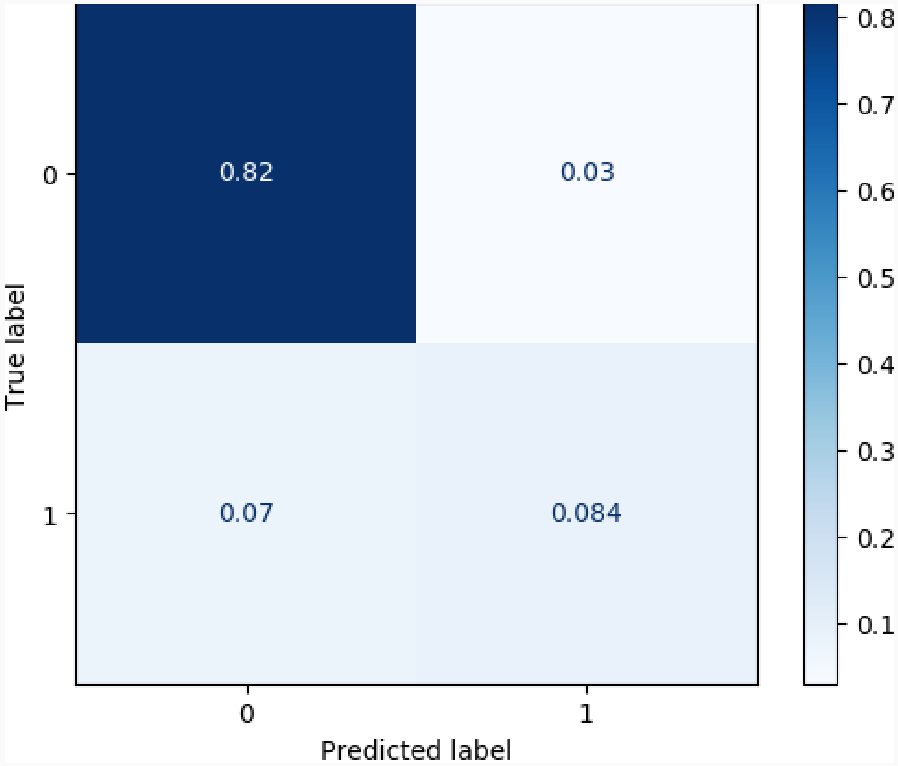
At 90%, Highest Accuracy Among All Classifiers.

Random Forest Classifier - Features



PageValues is the most important feature.

Random Forest Classifier - Performance - Test Set



90% Accuracy on Test Set.

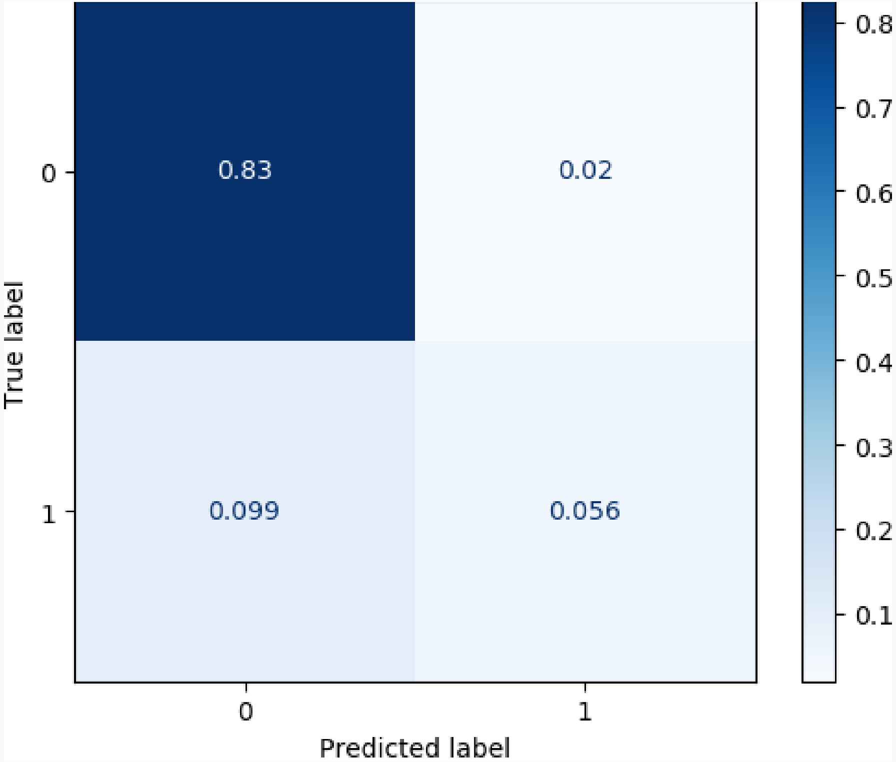
Logistic Regression - Model Specification

All combinations of following parameters were Cross Validated.

```
parameters = {
        'class_weight': [None, 'balanced'],
        'C': np.arange(0.1,1.05,0.05),
    }
```

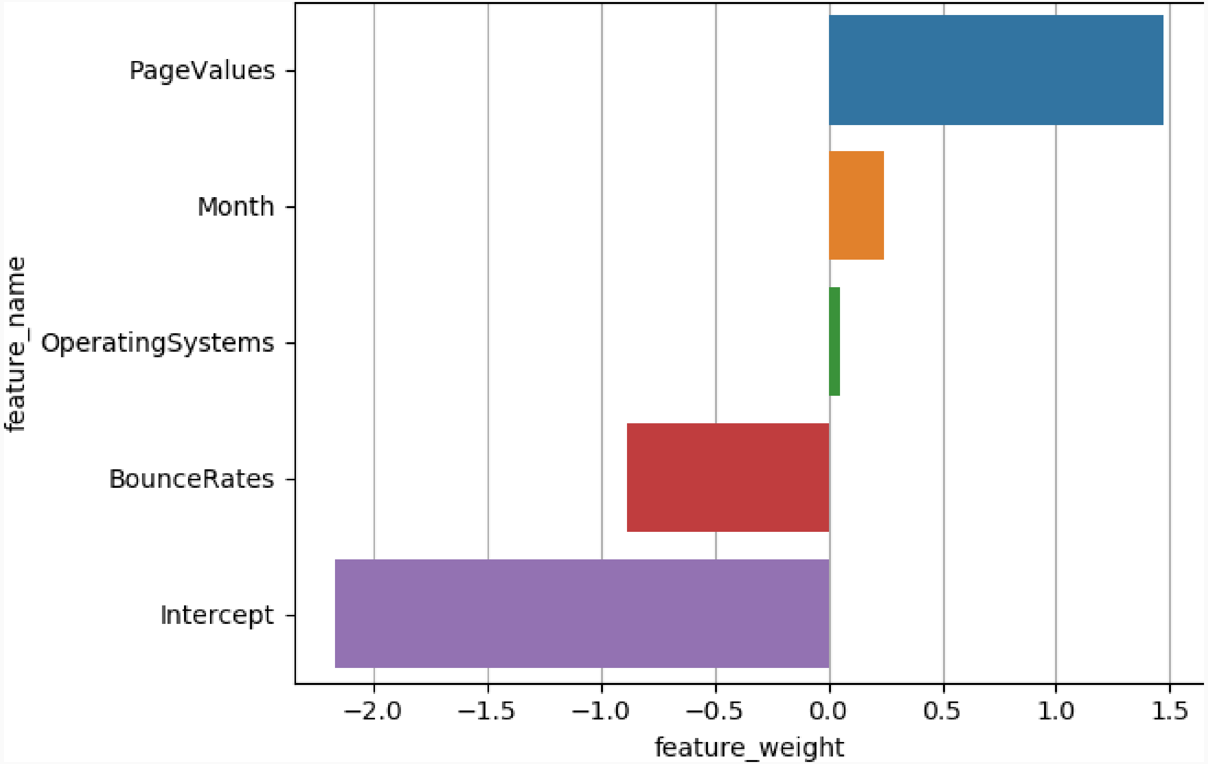
Highest CV accuracy of 88% for 'C': 0.45, 'class_weight': None

Logistic Regression - Performance - Validation Set



At 88%, Lowest Accuracy Among All Classifiers.

Logistic Regression - Features



PageValues is the most important feature.

Logistic Regression - Features - Interpretation

$$ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Unit change in a predictor gives change in Ln(odds) when all other predictors are constant.

K Nearest Neighbors - Model Specifications

All combinations of following parameters were Cross Validated.

```
parameters = {
            'weights': ['uniform', 'distance'],
            'p': np.arange(1,3,1),
            'n_neighbors': np.arange(1,20,1)
        }
```

Highest CV accuracy of 89% for 'n_neighbors': 18, 'p': 2, 'weights': 'uniform'.

K Nearest Neighbors - Features

PageValues

BounceRates

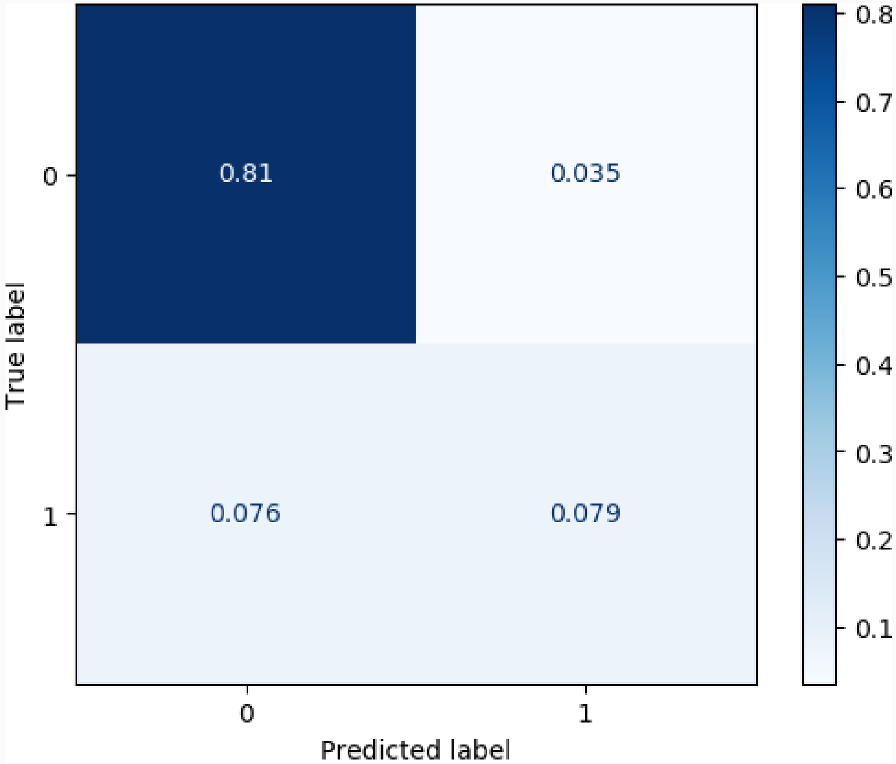
Month

OperatingSystems

Legend
Most Significant
Significant
Least Significant

Most of the features were dropped during feature selection.

K Nearest Neighbors - Performance - Validation Set



At 89%, Second Highest Accuracy Among All Classifiers.

Insights From Exploratory Data Analysis

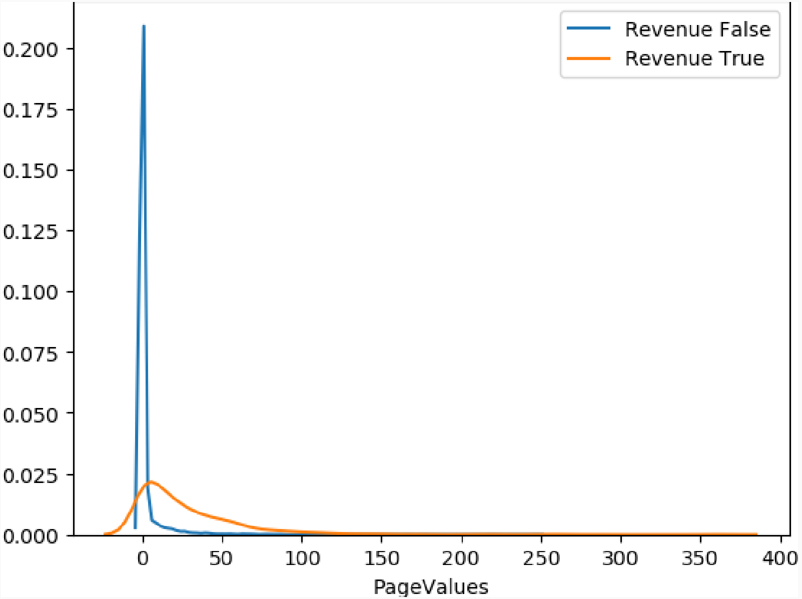
1. Only 15.4% of the shopping trips resulted in Revenue (target variable).
2. PageValues is correlated with Revenue.
3. ExitRates and BounceRates are highly correlated.
4. ProductRelated and ProductRelated_Duration are highly correlated.
5. Administrative, Informational, ProductRelated correlated with each other.
6. Durations of Administrative, Informational, ProductRelated also correlated.
7. <page_type> and <page_type>_Durations were moderately Correlated, where <page_type> is either of Administrative or Informational.
8. Among Month, Nov had the highest conversion rate of 25%.
9. Among Month, Mar had most visits however conversion rate of only 12%.
10. Among Visitors, New_Visitors had the highest conversion rate of 25%.

Conclusions

1. PageValue is the most important feature in predicting purchases.
2. BounceRate is the second most important feature in predicting purchases.
3. RFC’s accuracy was 5.4% better than Dumb Model (always predict 0).
4. Performance on RFC > KNN > LR
5. Higher values of PageValues promote Revenue.
6. Lower values of BounceRates promote Revenue.
7. While Month, OperatingSystem, Browser, Weekend are predictor of Revenue, they much less significant than PageValues, BounceRates.
8. Region, TrafficeType, VisitorType do not significantly affect purchases.

Business Outcomes - PageValues

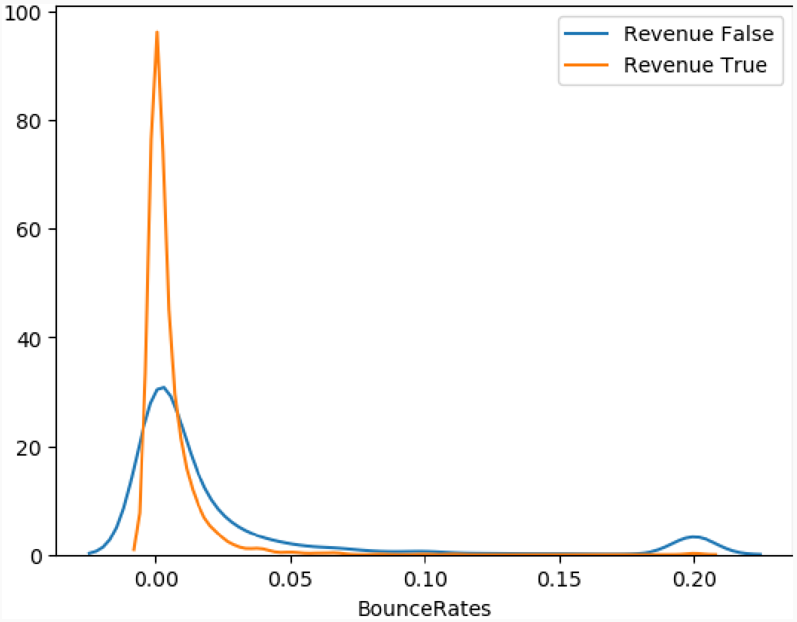
Shopping trips with low PageValues are more likely to have no Revenue.



Further dissect PageValues when Revenue was True. Work to increase PageValues.

Business Outcomes - BounceRates

Shopping trips with low BounceRates were more likely to have Revenue.



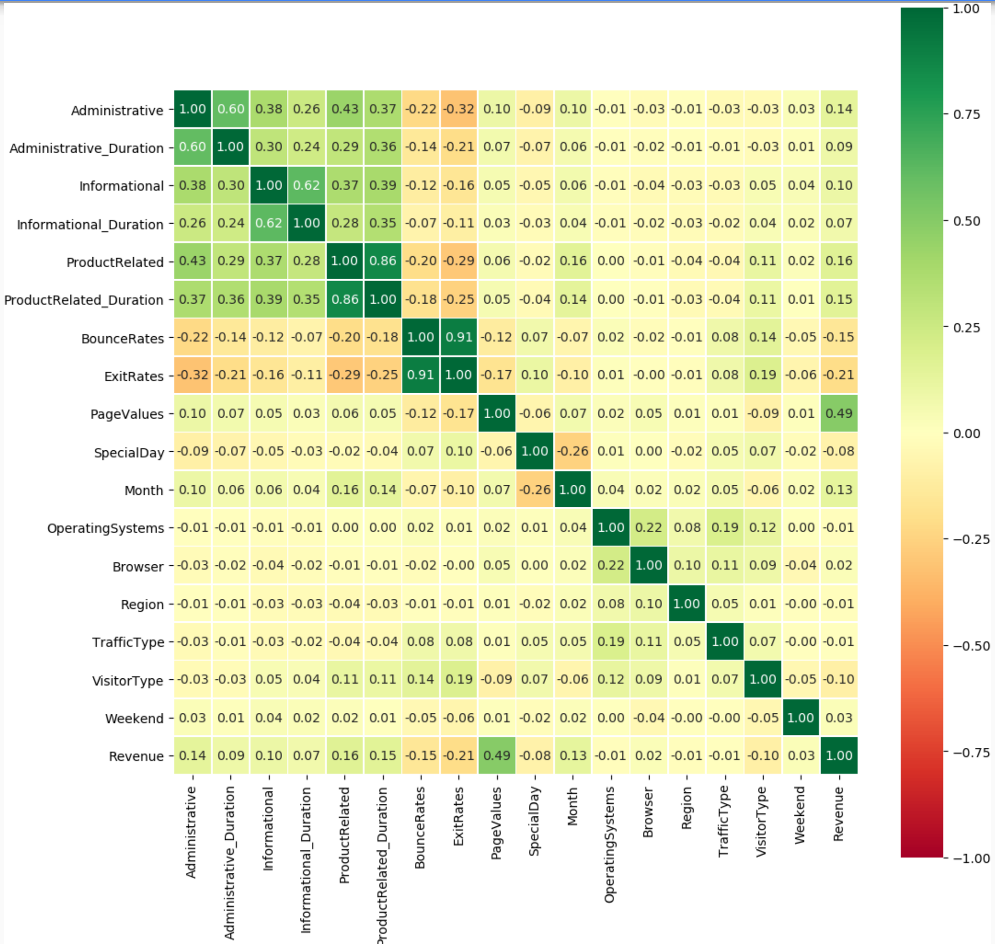
Further dissect BounceRates when Revenue was True. Work to decrease BounceRates.

Business Outcomes - Month, OS, Browser, Weekend

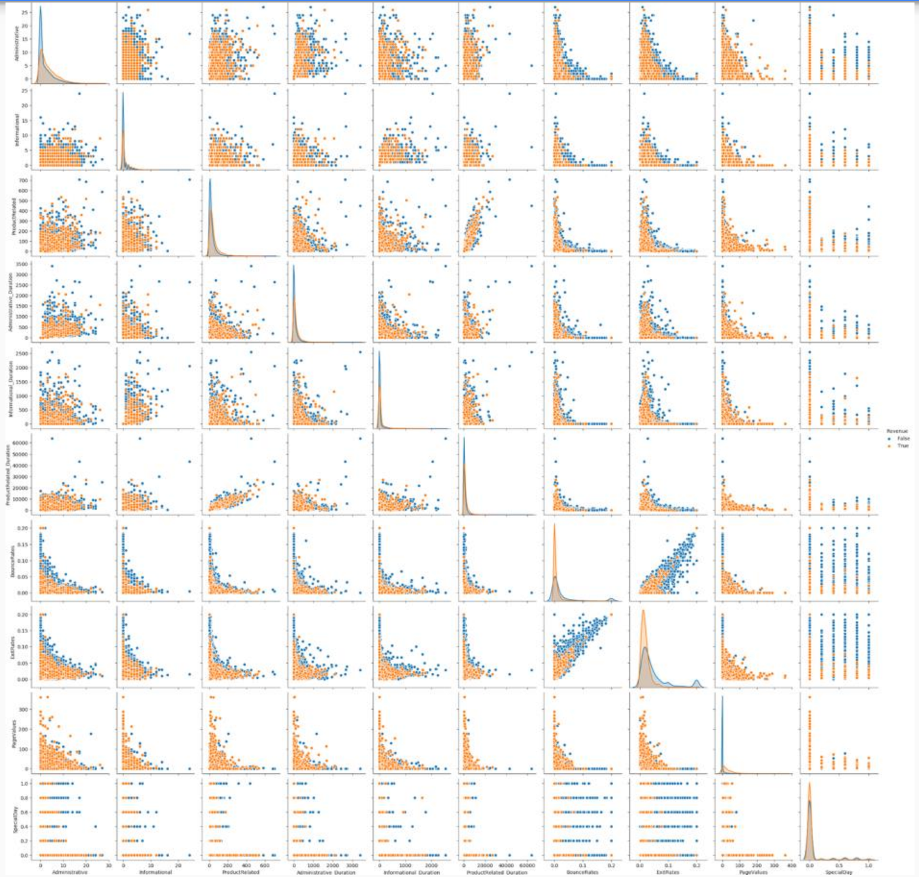
1. Selectively run promotions in Oct and Nov.
2. Prioritize/Selectively roll out updates to apps/services for OSs 1 & 2.
3. Prioritize/Selectively roll out updates to apps/services for browsers 1 & 2.
4. Since probabilities of Revenue are comparable for Weekend or not, selectively run promotions on weekdays to take advantage of more visits.

Appendix

Correlation Matrix

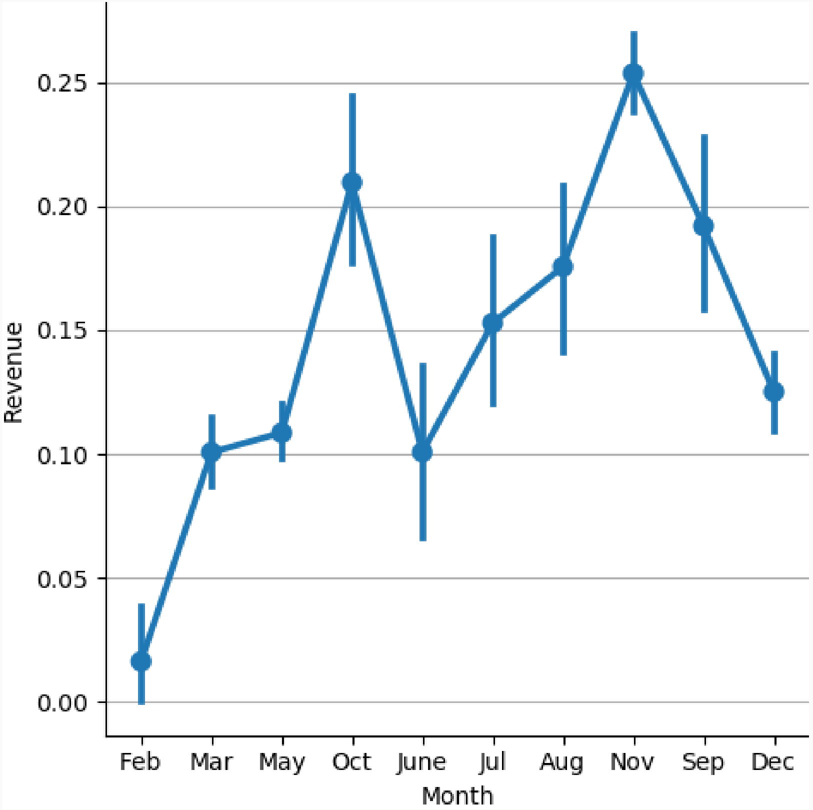


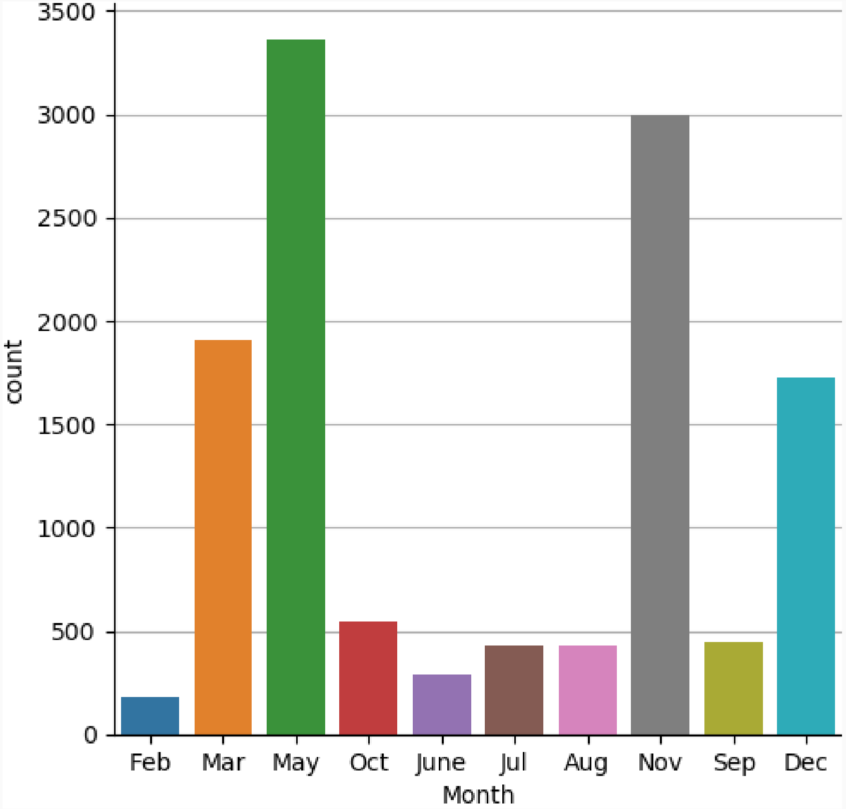
EDA - Pairwise Relationships



ExitRates and BounceRates Highly Correlated. PageValues discriminates over Revenue.

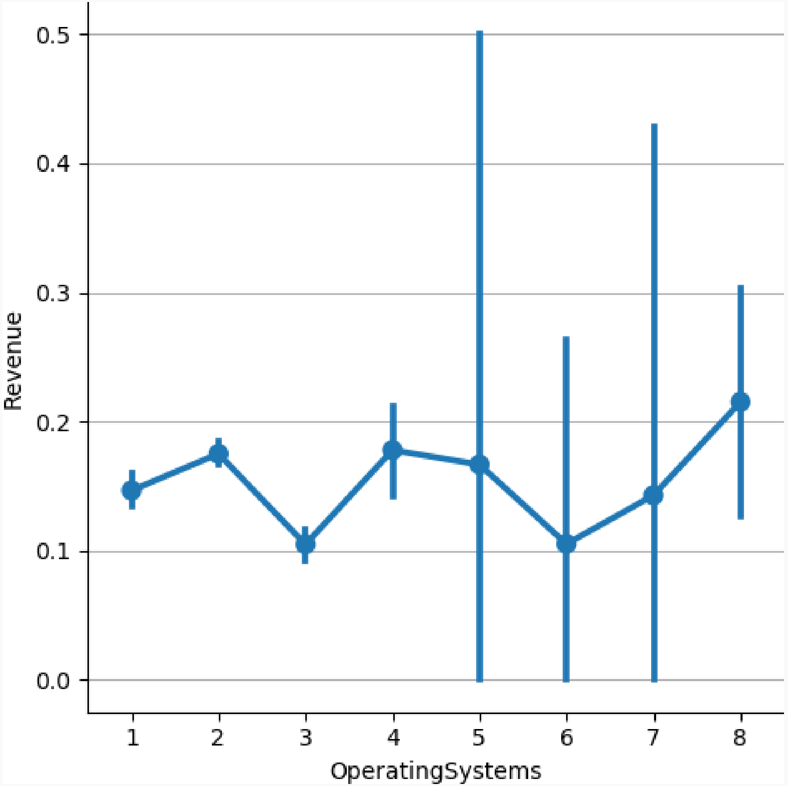
EDA - Month

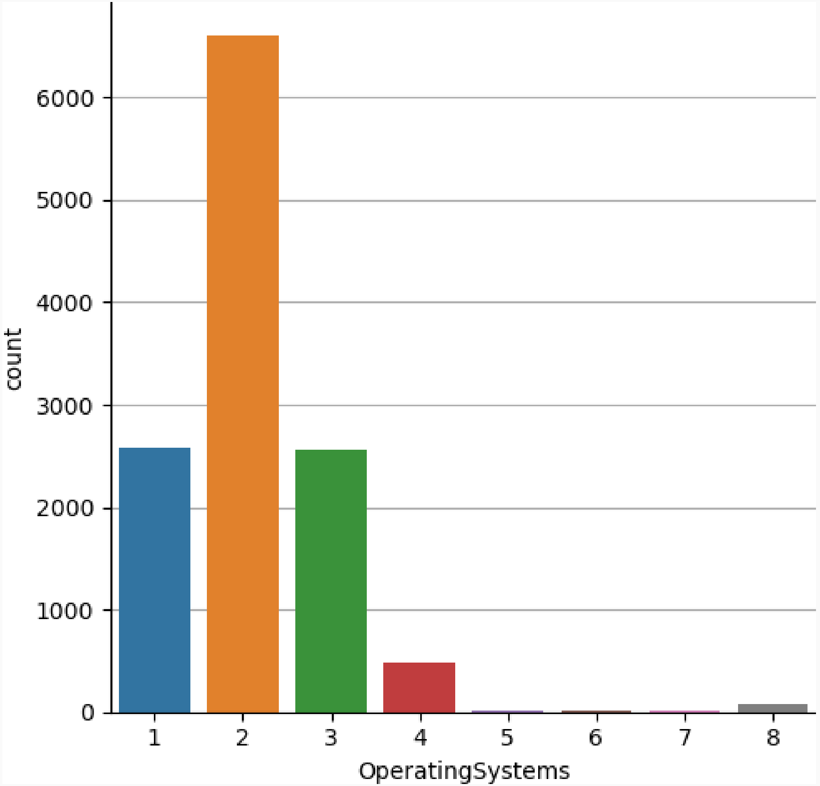




Highest Visits in May Yet Low Conversion. Highest Conversion Rate in Nov.

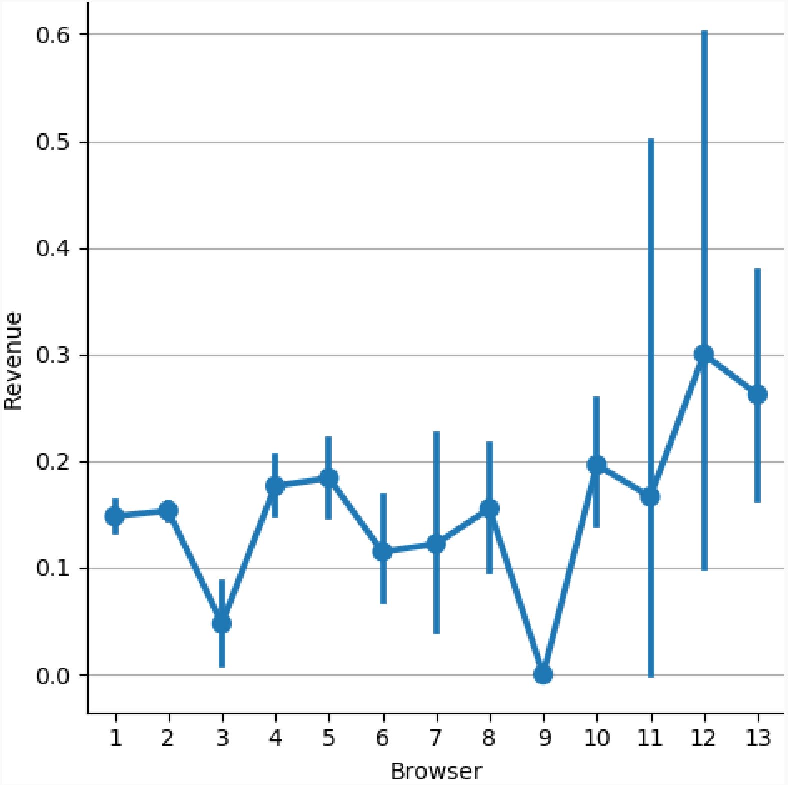
EDA - Operating System

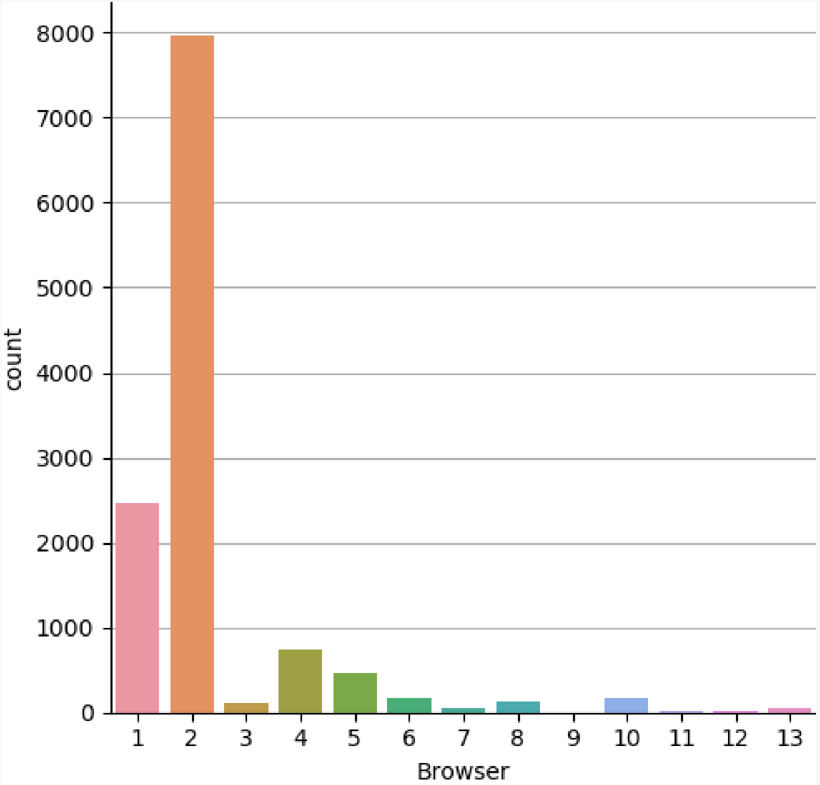




With most visits, OS 2 has one of the best conversion rates with low error.

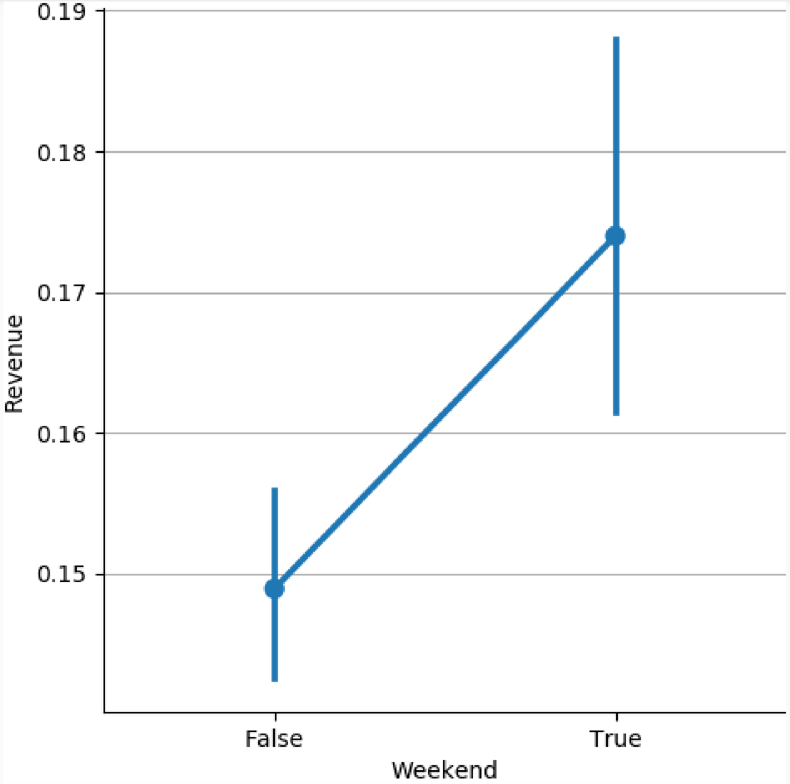
EDA - Browser

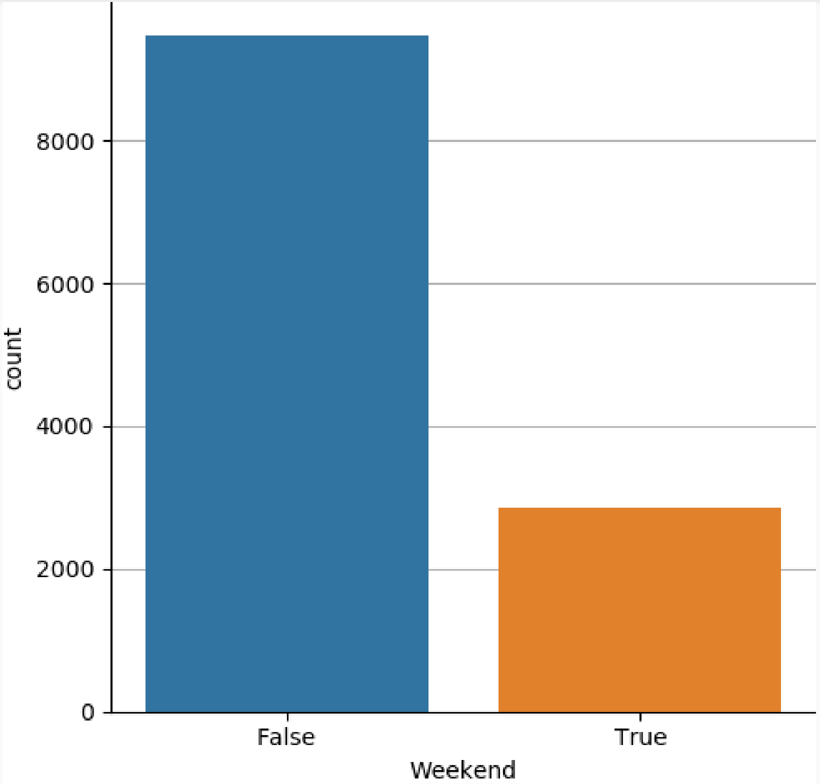




Browser 2 is the preferred choice of browser among visitors.

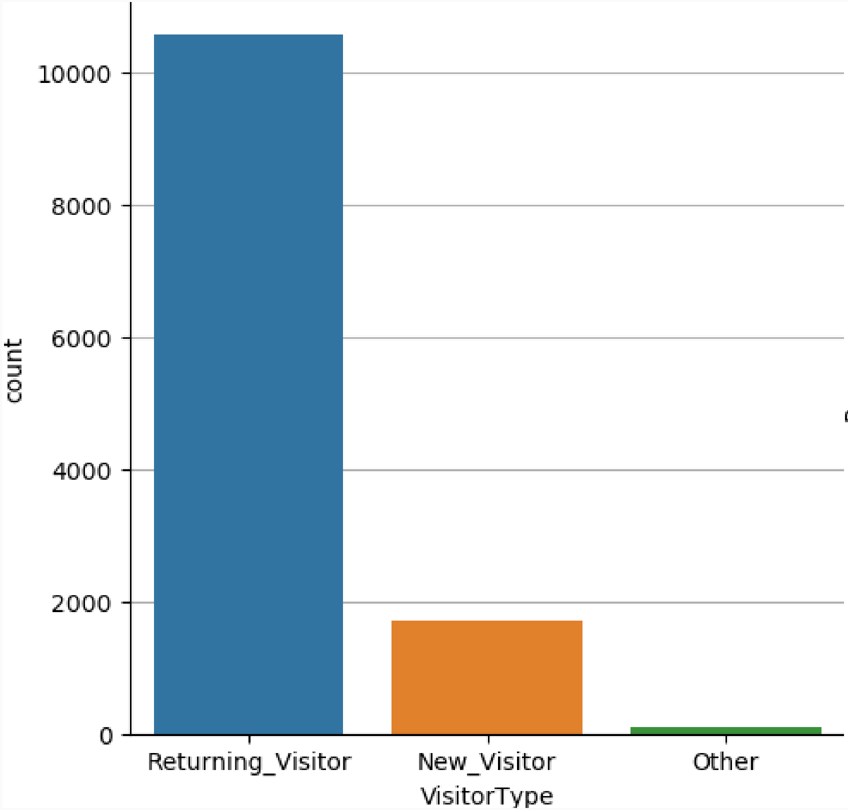
EDA - Weekend

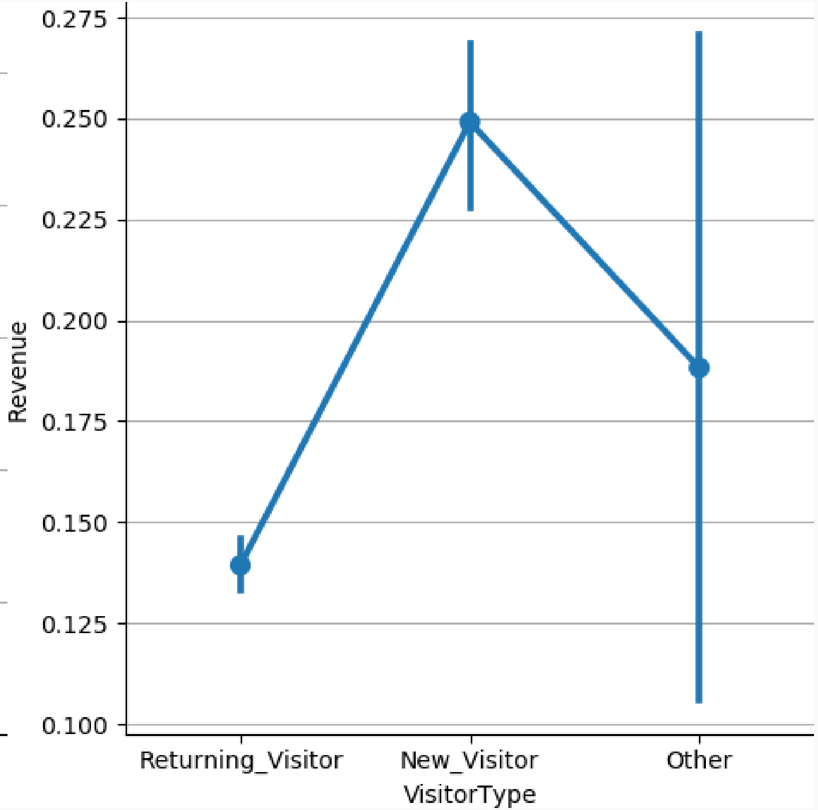




P(Revenue) ~ P(No Revenue). Most visits on weekends.

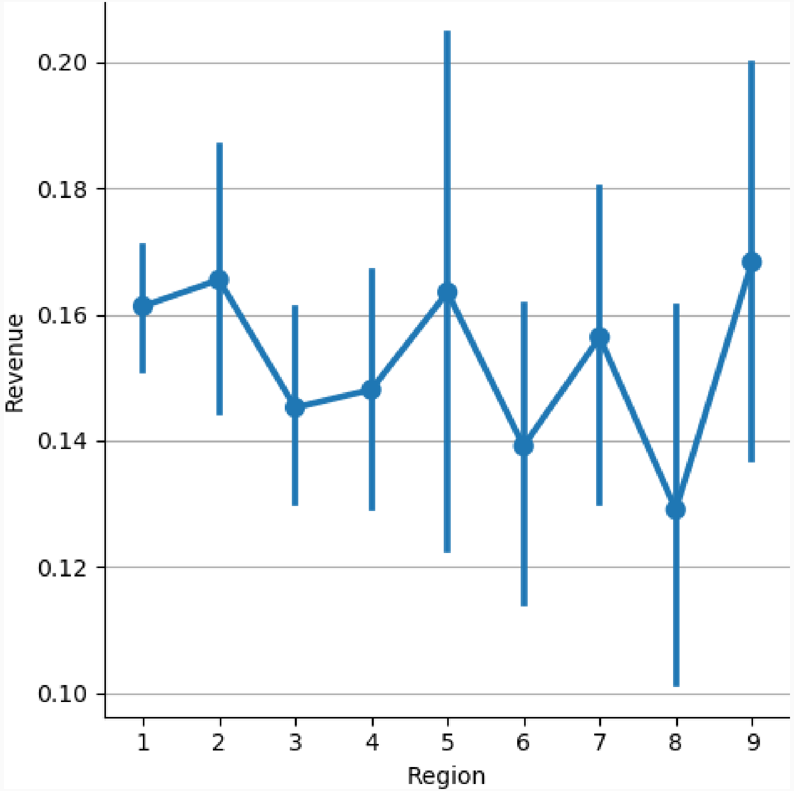
EDA - VisitorType

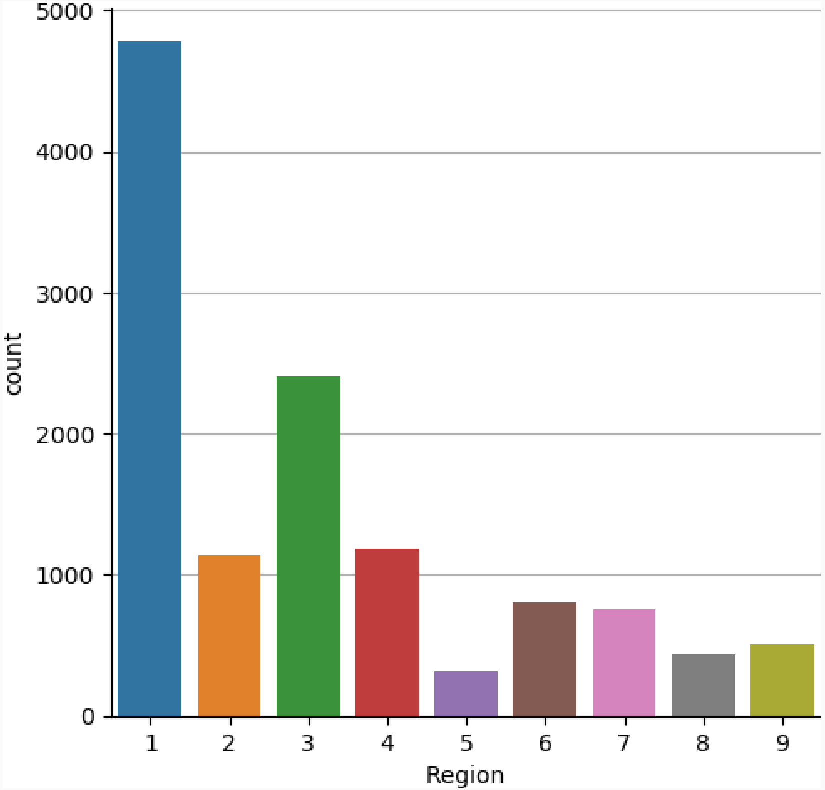




Despite Most Visits by Returning_Visitor, New_Visitor Most Likely To Purchase.

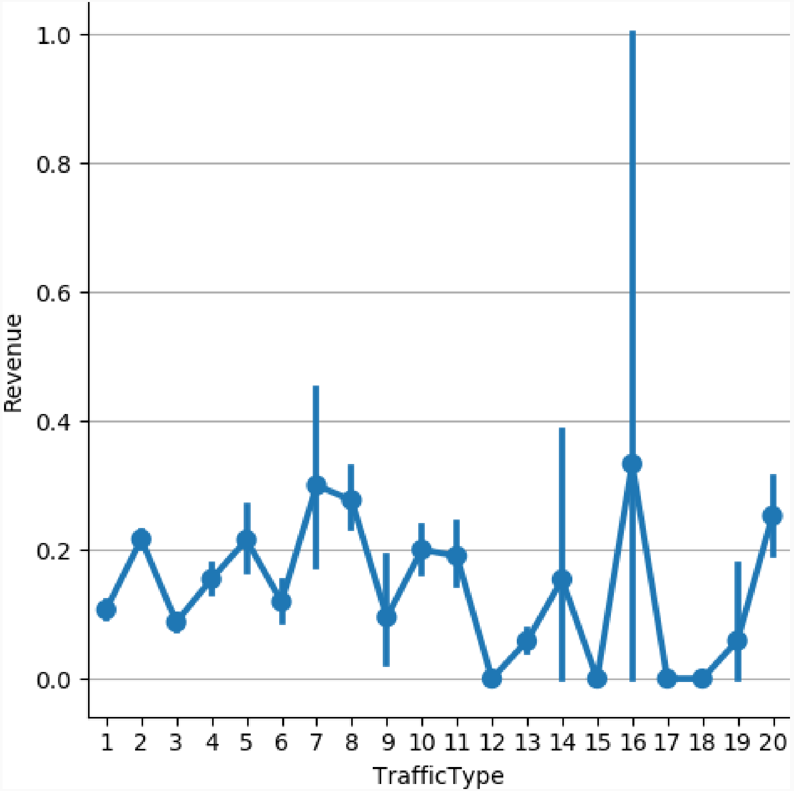
EDA - Region

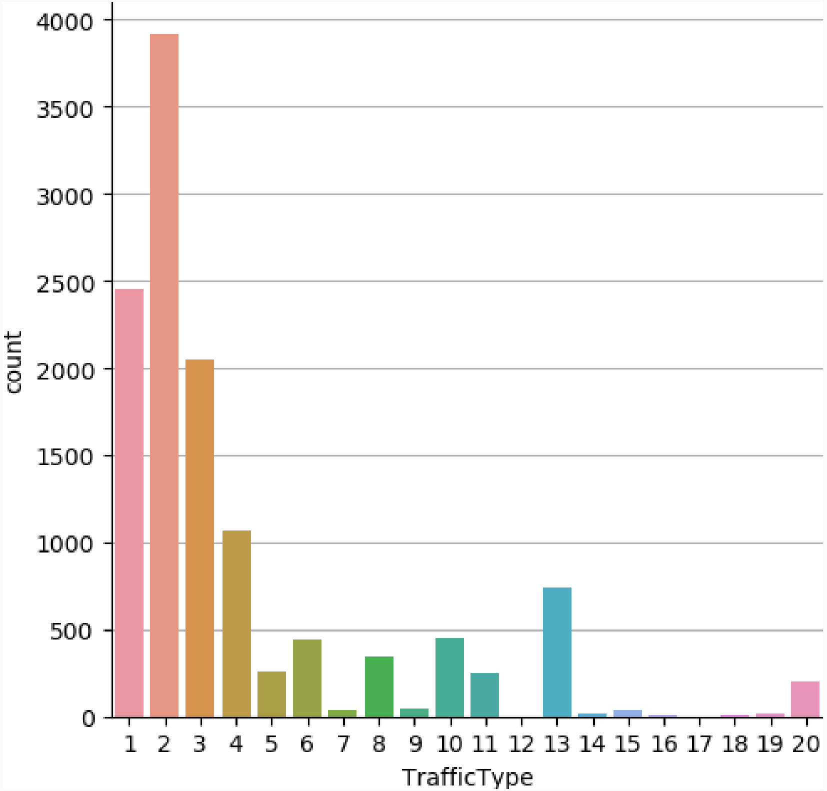




With most visits, Region 1 has one of the best conversion rates with low error.

EDA - TrafficType





With most visits, type 2 has one of the best conversion rates with low error.

Feature Extraction

1. <page_type> and <page_type>_Durations were correlated, where <page_type> is either of Administrative, Informational.
2. Therefore, these features were combined and tested. Eg. A+Ad = Administrative + Administrative_Duration.
3. Since, ProductRelated and ProductRelated_Duration were highly correlated, ProductRelated_Duration was dropped.
4. ExitRates was also dropped due to high correlation with BounceRates.
5. However, performance on Validation Set was inferior to A+I+PR, Ad+Id+PRd scheme and hence they were not evaluated on Test Set.

Preprocessing/Model Selection

1. Categorical variables Month, VisitorType, Weekend and Revenue were Level Encoded .
2. All numerical features were scaled to zero mean and unit variance.
3. OneHot encoding was tried but level encoding gave better results.
4. Model parameters were selected using 5 fold Cross Validation.

Feature Selection

1. An instance of RandomForestClassifier with default parameters was used for selecting features from the 17 Level Encoded features.
2. RandomForestClassifier was chosen for Feature Selection cause RFCs are non-linear models.
3. For each of the classifiers built, ‘mean’ and ‘median’ thresholds were tried for SelectFromModel while selecting features.
4. ‘mean’ threshold selected less features.
5. ‘median’ threhold selected more features.

CV Results - Random Forest Classifier

___RandomForestClassifier___

BEST PARAMS: {'class_weight': None, 'max_depth': 10, 'n_estimators': 75}

0.869 (+/-0.013) for {'class_weight': None, 'max_depth': 2, 'n_estimators': 5}
0.881 (+/-0.017) for {'class_weight': None, 'max_depth': 2, 'n_estimators': 50}
0.864 (+/-0.007) for {'class_weight': None, 'max_depth': 2, 'n_estimators': 75}
0.852 (+/-0.006) for {'class_weight': None, 'max_depth': 2, 'n_estimators': 100}
0.894 (+/-0.012) for {'class_weight': None, 'max_depth': 10, 'n_estimators': 5}
0.904 (+/-0.009) for {'class_weight': None, 'max_depth': 10, 'n_estimators': 50}
0.905 (+/-0.009) for {'class_weight': None, 'max_depth': 10, 'n_estimators': 75}
0.905 (+/-0.01) for {'class_weight': None, 'max_depth': 10, 'n_estimators': 100}
0.894 (+/-0.01) for {'class_weight': None, 'max_depth': 20, 'n_estimators': 5}
0.9 (+/-0.013) for {'class_weight': None, 'max_depth': 20, 'n_estimators': 50}
0.9 (+/-0.012) for {'class_weight': None, 'max_depth': 20, 'n_estimators': 75}
0.901 (+/-0.013) for {'class_weight': None, 'max_depth': 20, 'n_estimators': 100}
0.887 (+/-0.008) for {'class_weight': None, 'max_depth': None, 'n_estimators': 5}
0.899 (+/-0.015) for {'class_weight': None, 'max_depth': None, 'n_estimators': 50}
0.898 (+/-0.011) for {'class_weight': None, 'max_depth': None, 'n_estimators': 75}
0.898 (+/-0.013) for {'class_weight': None, 'max_depth': None, 'n_estimators': 100}
0.876 (+/-0.02) for {'class_weight': 'balanced', 'max_depth': 2, 'n_estimators': 5}
0.874 (+/-0.02) for {'class_weight': 'balanced', 'max_depth': 2, 'n_estimators': 50}
0.874 (+/-0.02) for {'class_weight': 'balanced', 'max_depth': 2, 'n_estimators': 75}
0.874 (+/-0.02) for {'class_weight': 'balanced', 'max_depth': 2, 'n_estimators': 100}
0.881 (+/-0.016) for {'class_weight': 'balanced', 'max_depth': 10, 'n_estimators': 5}
0.89 (+/-0.016) for {'class_weight': 'balanced', 'max_depth': 10, 'n_estimators': 50}
0.89 (+/-0.013) for {'class_weight': 'balanced', 'max_depth': 10, 'n_estimators': 75}
0.89 (+/-0.013) for {'class_weight': 'balanced', 'max_depth': 10, 'n_estimators': 100}
0.893 (+/-0.014) for {'class_weight': 'balanced', 'max_depth': 20, 'n_estimators': 5}
0.899 (+/-0.012) for {'class_weight': 'balanced', 'max_depth': 20, 'n_estimators': 50}
0.9 (+/-0.009) for {'class_weight': 'balanced', 'max_depth': 20, 'n_estimators': 75}
0.9 (+/-0.009) for {'class_weight': 'balanced', 'max_depth': 20, 'n_estimators': 100}

CV Results - K Nearest Neighbors

___KNeighborsClassifier___

BEST PARAMS: {'n_neighbors': 18, 'p': 2, 'weights': 'uniform'}

0.853 (+/-0.014) for {'n_neighbors': 1, 'p': 1, 'weights': 'uniform'}
0.853 (+/-0.014) for {'n_neighbors': 1, 'p': 1, 'weights': 'distance'}
0.853 (+/-0.012) for {'n_neighbors': 1, 'p': 2, 'weights': 'uniform'}
0.853 (+/-0.012) for {'n_neighbors': 1, 'p': 2, 'weights': 'distance'}
0.876 (+/-0.007) for {'n_neighbors': 2, 'p': 1, 'weights': 'uniform'}
0.853 (+/-0.014) for {'n_neighbors': 2, 'p': 1, 'weights': 'distance'}
0.874 (+/-0.007) for {'n_neighbors': 2, 'p': 2, 'weights': 'uniform'}
0.853 (+/-0.012) for {'n_neighbors': 2, 'p': 2, 'weights': 'distance'}
0.881 (+/-0.019) for {'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}
0.871 (+/-0.014) for {'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
0.882 (+/-0.012) for {'n_neighbors': 3, 'p': 2, 'weights': 'uniform'}
0.874 (+/-0.014) for {'n_neighbors': 3, 'p': 2, 'weights': 'distance'}
0.886 (+/-0.011) for {'n_neighbors': 4, 'p': 1, 'weights': 'uniform'}
0.878 (+/-0.009) for {'n_neighbors': 4, 'p': 1, 'weights': 'distance'}
0.884 (+/-0.011) for {'n_neighbors': 4, 'p': 2, 'weights': 'uniform'}
0.878 (+/-0.009) for {'n_neighbors': 4, 'p': 2, 'weights': 'distance'}
0.887 (+/-0.011) for {'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}
0.88 (+/-0.01) for {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
0.887 (+/-0.012) for {'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}
0.883 (+/-0.012) for {'n_neighbors': 5, 'p': 2, 'weights': 'distance'}
0.889 (+/-0.011) for {'n_neighbors': 6, 'p': 1, 'weights': 'uniform'}
0.883 (+/-0.009) for {'n_neighbors': 6, 'p': 1, 'weights': 'distance'}
0.886 (+/-0.013) for {'n_neighbors': 6, 'p': 2, 'weights': 'uniform'}
0.882 (+/-0.01) for {'n_neighbors': 6, 'p': 2, 'weights': 'distance'}
0.891 (+/-0.008) for {'n_neighbors': 7, 'p': 1, 'weights': 'uniform'}
0.884 (+/-0.012) for {'n_neighbors': 7, 'p': 1, 'weights': 'distance'}
0.887 (+/-0.008) for {'n_neighbors': 7, 'p': 2, 'weights': 'uniform'}
0.882 (+/-0.01) for {'n_neighbors': 7, 'p': 2, 'weights': 'distance'}

CV Results - Logistic Regression

___LogisticRegression___

BEST PARAMS: {'C': 0.45000000000000007, 'class_weight': None}

0.883 (+/-0.01) for {'C': 0.1, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.1, 'class_weight': 'balanced'}
0.883 (+/-0.01) for {'C': 0.15000000000000002, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.15000000000000002, 'class_weight': 'balanced'}
0.883 (+/-0.01) for {'C': 0.20000000000000004, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.20000000000000004, 'class_weight': 'balanced'}
0.883 (+/-0.011) for {'C': 0.25000000000000006, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.25000000000000006, 'class_weight': 'balanced'}
0.884 (+/-0.01) for {'C': 0.30000000000000004, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.30000000000000004, 'class_weight': 'balanced'}
0.884 (+/-0.01) for {'C': 0.3500000000000001, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.3500000000000001, 'class_weight': 'balanced'}
0.884 (+/-0.01) for {'C': 0.40000000000000013, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.40000000000000013, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.45000000000000007, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.45000000000000007, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.5000000000000001, 'class_weight': None}
0.876 (+/-0.017) for {'C': 0.5000000000000001, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.5500000000000002, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.5500000000000002, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.6000000000000002, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.6000000000000002, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.6500000000000001, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.6500000000000001, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.7000000000000002, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.7000000000000002, 'class_weight': 'balanced'}
0.884 (+/-0.011) for {'C': 0.7500000000000002, 'class_weight': None}
0.875 (+/-0.017) for {'C': 0.7500000000000002, 'class_weight': 'balanced'}

Languages & Libraries Used

1. Language: Python 3
2. Libraries:
    a. Scikit-learn
    b. Seaborn
    c. Matplotlib
    d. Pandas
    e. Numpy