ANALYSIS OF KEYWORDS FOR SEO IN A9 ALGORITHM

- **Surya N**

PROBLEM DESCRIPTION:

Search engine optimization is the process of optimizing web pages and their content to be easily discoverable by users searching for terms relevant to your website. The term SEO also describes the process of making web pages easier for search engine indexing software, known as "crawlers," to find, scan, and index your site. The A9 Algorithm is the system which Amazon uses to decide how products are ranked in search results. It is similar to the algorithm which Google uses for its search results, in that it considers keywords in deciding which results are most relevant to the search and therefore which it will display first. However, there is one key difference between Google and Amazon's algorithms: the A9 algorithm also puts a strong emphasis on sales conversions. This is because Amazon is a business, and has a vested interest in promoting listings which are more likely to result in sales. Therefore Amazon will rank listings with a strong sales history and high conversion rate more highly. Understanding how the algorithm works means you can rank highly on Amazon searches, which is the number one thing you can do to drive traffic to your listings, to ultimately drive sales. As mentioned, keywords are one of the main factors Amazon looks for in determining relevance to search queries and therefore setting rankings on its results pages. Therefore it is critical to integrate high volume and importantly relevant keywords as part of your listings.

PROBLEM STATEMENT:

A retail shoe store decided to list and sell their product on Amazon. In the process of selling , an Amazon Business Advisory(ABA) insists to run campaigns

for keywords related to the product that is sold. The ABA has given us a dataset of related keywords and has also given a initial level suggestion to use the generated keywords or not. The retail store owner needs to know on what basis the ABA has decided to use a keyword with the given dataset. We have created some supervised learning Classification models for the dataset to determine the usability of a particular keyword. Out of all keywords, we shall also choose the best model for our requirement.

DATASET DESCRIPTION:

The dataset used for analysis is the keywords selection data from the SEO keywords repository called E-Grow. In that dataset there are 500 keywords related to shoes. They include 21 attributes, 19 are numeric and "keyword" attribute alone is a string. This also includes a class attribute. The detailed attribute information is given below.

ATTRIBUTES	**ATTRIBUTE DESCRIPTION**
Keyword	Keyword to be used for Campaign
Search_Volume	Google Search Volume of the Keyword
Total_Results	Total Results Obtained in a A9 Search
BSR_Min	Minimum Best Seller Rank of the product which uses that Keyword
BSR_Avg	Average Best Seller Rank of the product which uses that Keyword
BSR_Max	Maximum Best Seller Rank of the product which uses that Keyword

Price_Min	Minimum Price of the product which uses that Keyword
Price_Avg	Average Price of the product which uses that Keyword
Price_Max	Maximum Price of the product which uses that Keyword
Reviews_Min	Minimum Number of Reviews of the product which uses that Keyword
Reviews_Avg	Average Number of Reviews of the product which uses that Keyword
Reviews_Max	Maximum Number of Reviews of the product which uses that Keyword
Sales_Min	Minimum Sales of the product which uses that Keyword
Sales_Avg	Avearge Sales of the product which uses that Keyword
Sales_Max	Maximum Sales of the product which uses that Keyword
Revenue_Min	Minimum Revenue generated of the product which uses that Keyword
Revenue_Avg	Avearge Revenue generated of the product which uses that Keyword
Revenue_Max	Maximum Revenue generated of the product which uses that Keyword
Revenue_Total	Total Revenue generated of the product which uses that Keyword

Oppurtunity_Score	Oppurtunity Score Given by the ABA to the Keyword
Use	Decision to use the Keyword

SAMPLE DATASET:

SOURCE LINK: EGrow Keywords Repository

https://in.egrow.io/member/keyword-niche-tool

Keyword	Search_Vc	Total_Res	BSR_Min	BSR_Avg	BSR_Max	Price_Min	Price_Avg	Price_Max	Reviews_M	Reviews_A	Reviews_M	Sales_Min	Sales_Avg	Sales_Max	Revenue_I	Revenue_A	Revenue_I	Revenue_T	Oppurtunit	Use
11children shoes office shoes	0	272	1456	22576	62712	379	577	799	0	3	19	1	8	30	599	4087	12784	40873	0	NO
361 shoes white shoes	0	230	65120	65120	65120	1764	1764	1764	0	0	0	1	1	1	1764	1764	1764	17640	1	NO
a6 shoes running shoes white colour	0	2	7	84810	438890	299	741	1299	0	414	2878	0	91	336	0	80637	334656	645094	2	NO
aces shoes running shoes	0	50	663	65085	335402	454	1307	2799	0	51	249	0	10	50	0	12526	61578	112731	3	YES
action shoes for men shoes	0	739	903	5301	11836	559	720	1299	0	35	216	5	15	41	2995	11178	24559	100600	2	NO
adidas shoes	260	5000	452	4801	16747	1349	2474	3999	0	29	133	4	25	63	9995	55549	120267	555493	3	NO
all shoes company shoes men	0	8000	2	189	785	449	776	1739	37	1047	3232	45	187	460	56387	128637	350649	1157730	4	YES
all shoes men sports shoes	0	20000	2	6470	39503	475	740	1610	0	930	3894	1	146	460	1610	98485	289575	984848	5	YES
all star shoes for men shoes	0	1000	114	15299	75240	340	1970	4999	0	128	659	1	24	125	4688	33539	194875	335392	2	NO
allen shoes for men shoes	0	1000	618	8539	55738	999	1576	2067	0	120	466	1	17	52	1648	24672	53592	246718	1	NO
allen shoes running shoes	0	98	643	11975	41028	999	1254	1449	0	11	85	1	15	51	1399	17365	64950	173654	2	NO
allen shoes sport shoes	0	115	216	11471	43972	999	1313	1749	0	51	466	1	24	83	999	33863	136037	338626	0	NO
allstar shoes for men shoes	0	1000	114	11217	38359	409	1738	3149	0	123	659	1	24	125	1080	36013	194875	360129	1	NO
amico shoes running shoes	0	43	68	11020	27840	449	494	499	0	15	97	2	38	156	998	18727	77844	187274	5	YES
appe shoes red shoes	0	93	1703	22189	51958	409	536	600	0	7	22	1	7	27	599	3567	15120	32101	2	YES
appe shoes sneaker shoes	0	194	6036	27758	92944	489	578	600	0	7	23	0	4	10	0	2282	5990	22823	1	NO
asion shoes for men shoes	0	10000	1	855	4385	389	511	648	7	525	3209	13	139	532	7137	72805	265468	655243	4	YES
axonza shoes for black shoe	0	115	2428	17756	44372	399	484	599	0	215	484	1	8	21	399	3792	10479	37916	1	NO
bata shoes for men shoes	0	2000	199	9162	43226	599	913	1499	8	142	457	1	33	96	824	27707	80160	277069	4	YES
batar shoes formal shoes	0	1000	185	11755	56141	549	949	1499	1	51	174	1	30	100	549	26089	74900	260894	4	YES
belt shoes in school shoes	0	5000	6	1384	5379	284	660	975	0	394	3197	11	90	351	4260	65795	289575	657945	5	YES
big shoes small shoes	0	2000	3	80	565	232	800	1749	485	1856	3496	55	258	418	81432	163914	289575	1639141	5	YES
black shoe black shoe	260	80000	6	688	4065	394	755	1409	4	579	3197	14	174	351	12586	142967	289575	1286705	5	YES
board shoes skating shoes	0	109	3200	21493	46133	199	6972	29519	0	1	5	0	8	24	0	6137	25143	30686	2	NO
boot shoes	40	70000	52	3481	14199	399	569	899	0	50	407	4	55	173	2994	30176	103627	271587	4	YES

MODULES USED:

Our project involves 6 modules which involves Import module, Data Pre-processing module, EDA module, Classification model Fitting module, Hyperparameter tuning module and Best Model Identification module.

DATA PRE-PROCESSING:

Firstly, we need to check for null values in the dataset. Here our dataset is yet to be cleaned, so we convert the "?" symbols and null values to Nan. Then we analyse each and every column of the dataset. Only the first, penultimate and the last columns of the dataset are non-empty. Rest of the 18 attributes consists of Nan values, which we replace it with the mean of that particular column. We should also label encode the class attribute as "YES" = 1 and "NO" = 0.Then, we visualize the dataset as a heatmap with the correlation coefficients into it. We can very well see a complex box like patterns near the diagonal of the heatmap. This strongly suggests that the values of BSR, Price, Sales, Reviews and Revenue are repeated as three different attributes namely Minimum, Average and Maximum. The absolute positive correlation suggests us to remove any 2 out of 3 different kind of attributes for BSR, Price, Sales, Reviews and Revenue. We also remove the Search_Volume and Total_Results because these are Google's data and thus has the least significant attachment with the A9 algorithm.This does not affect the performance of the model at all. Furthermore, it optimizes the time complexity of the code. Then , we scale the data with Standard Scalar. Now our data pre-processing is over and now the cleaned data is ready for data analysis.

EXPLORATORY DATA ANALYSIS:

Exploratory Data Analysis (EDA) is an approach for analysing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modelling task. For better understanding of our keywords dataset, we will perform EDA in such a way that we can clearly see the dependencies and weightage of values clearly. After understanding the variables, it will be easy for us to implement them in the

model. After the EDA is complete, we shall split the data into training and test set with the ratio of 75:25.

CODE FOR DATA PRE-PROCESSING AND EDA:

```
#IMPORTING THE DATASET
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
from matplotlib.colors import ListedColormap
from sklearn.metrics import scorer
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc, f1_score, recall_score, pre
cision_score


#DATAPREPROCESSING AND EDA
from sklearn import preprocessing
dataframe=pd.read_csv("/content/shoesseodataset.csv")
print(dataframe)
fig,axes = plt.subplots(figsize=(10,8))
correl = dataframe.corr()
sns.heatmap(dataframe.corr(),cmap='Spectral',annot=True,ax=axes)
x = dataframe.drop(columns=['Oppurtunity_Score']).values
y = dataframe['Oppurtunity_Score'].values
plt.show()
dataframe.drop(dataframe.columns[[1,2,3,5,6,8,9,11,12,14,15,17,18]], ax
is = 1, inplace = True)
dataframe['BSR_Avg'].fillna(value=dataframe['BSR_Avg'].mean(), inplace=
True)
dataframe['Price_Avg'].fillna(value=dataframe['Price_Avg'].mean(), inpl
ace=True)
dataframe['Reviews_Avg'].fillna(value=dataframe['Reviews_Avg'].mean(),
inplace=True)
dataframe['Sales_Avg'].fillna(value=dataframe['Sales_Avg'].mean(), inpl
ace=True)
```

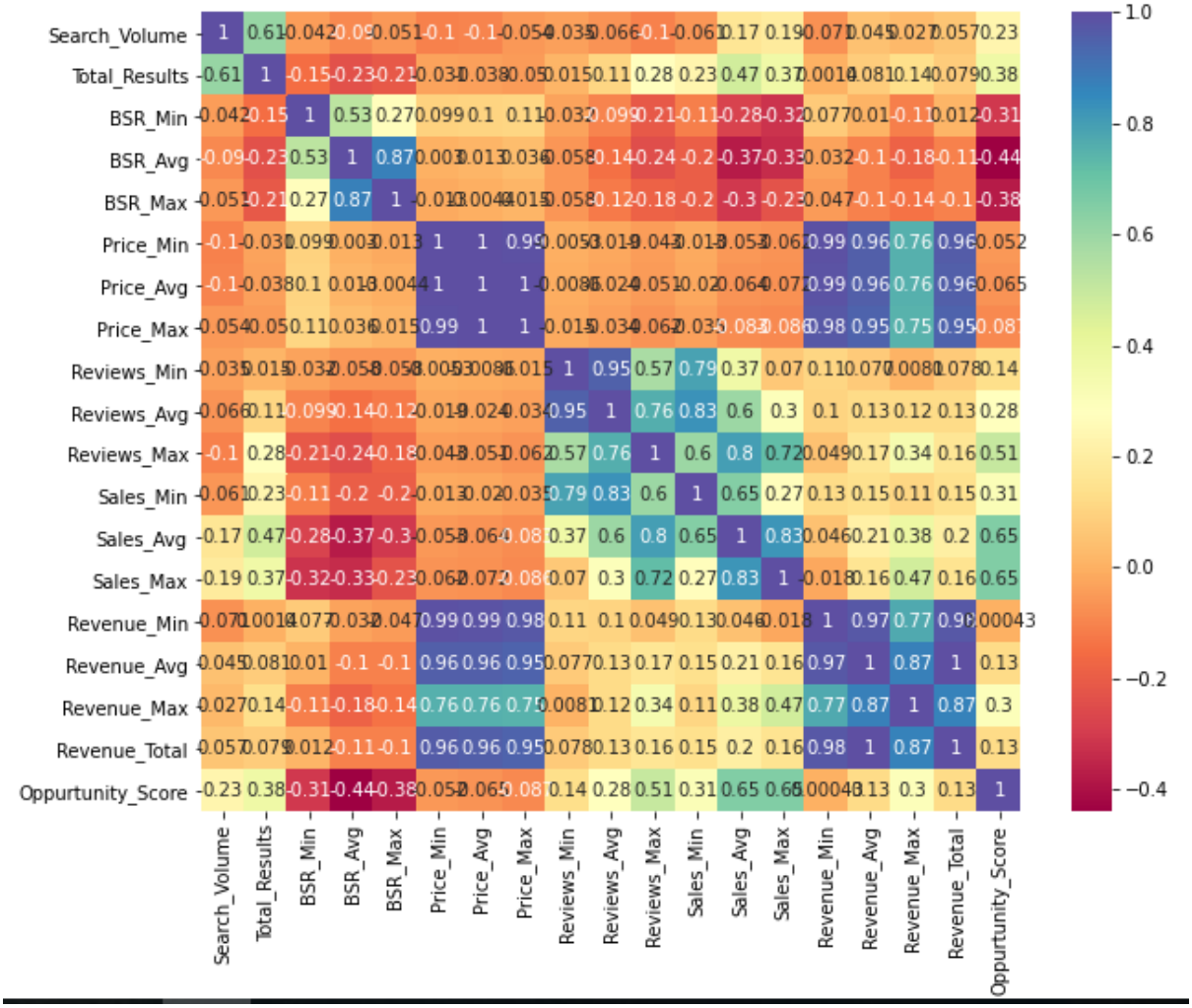
```
dataframe['Revenue_Avg'].fillna(value=dataframe['Revenue_Avg'].mean(),
inplace=True)
print(dataframe)
fig,axes = plt.subplots(figsize=(10,8))
correl = dataframe.corr()
sns.heatmap(dataframe.corr(),cmap='Spectral',annot=True,ax=axes)
x = dataframe.drop(columns=['Oppurtunity_Score']).values
y = dataframe['Oppurtunity_Score'].values
plt.show()
dataframe=dataframe.replace(to_replace ="NO", value =0)
dataframe=dataframe.replace(to_replace ="YES", value =1)
print("\n\t\t\tCOUNTPLOT\t")
sns.countplot(x="Use",data=dataframe,palette="hls")
plt.show()
print("\nNUMBER OF YES AND NO")
print(dataframe["Use"].value_counts())
pd.set_option('display.max_colwidth',1000)
X = dataframe.iloc[:,1:-1 ].values
y = dataframe.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.25, random_state = 19,stratify=y)
#FEATURE SCALING
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from matplotlib import pyplot
pyplot.scatter(dataframe.BSR_Avg, dataframe.Oppurtunity_Score)
pyplot.title("Relationship- BSR_Avg and Oppurtunity_Score")
pyplot.xlabel("BSR_Avg")
pyplot.ylabel("Oppurtunity_Score")
pyplot.show()
pyplot.scatter(dataframe.Revenue_Avg, dataframe.Price_Avg)
pyplot.title("Relationship- Revenue_Avg and Price_Avg")
pyplot.xlabel("Revenue_Avg")
pyplot.ylabel("Price_Avg")
pyplot.show()
pyplot.scatter(dataframe.Sales_Avg, dataframe.BSR_Avg)
pyplot.title("Relationship- Sales_Avg and BSR_Avg")
pyplot.xlabel("Sales_Avg")
pyplot.ylabel("BSR_Avg")
pyplot.show()
pyplot.scatter(dataframe.Sales_Avg, dataframe.Oppurtunity_Score)
pyplot.title("Relationship- Sales_Avg and Oppurtunity_Score")
pyplot.xlabel("Sales_Avg")
pyplot.ylabel("Oppurtunity_Score")
pyplot.show()
```

OUTPUT:

DATA-PREPROCESSING :

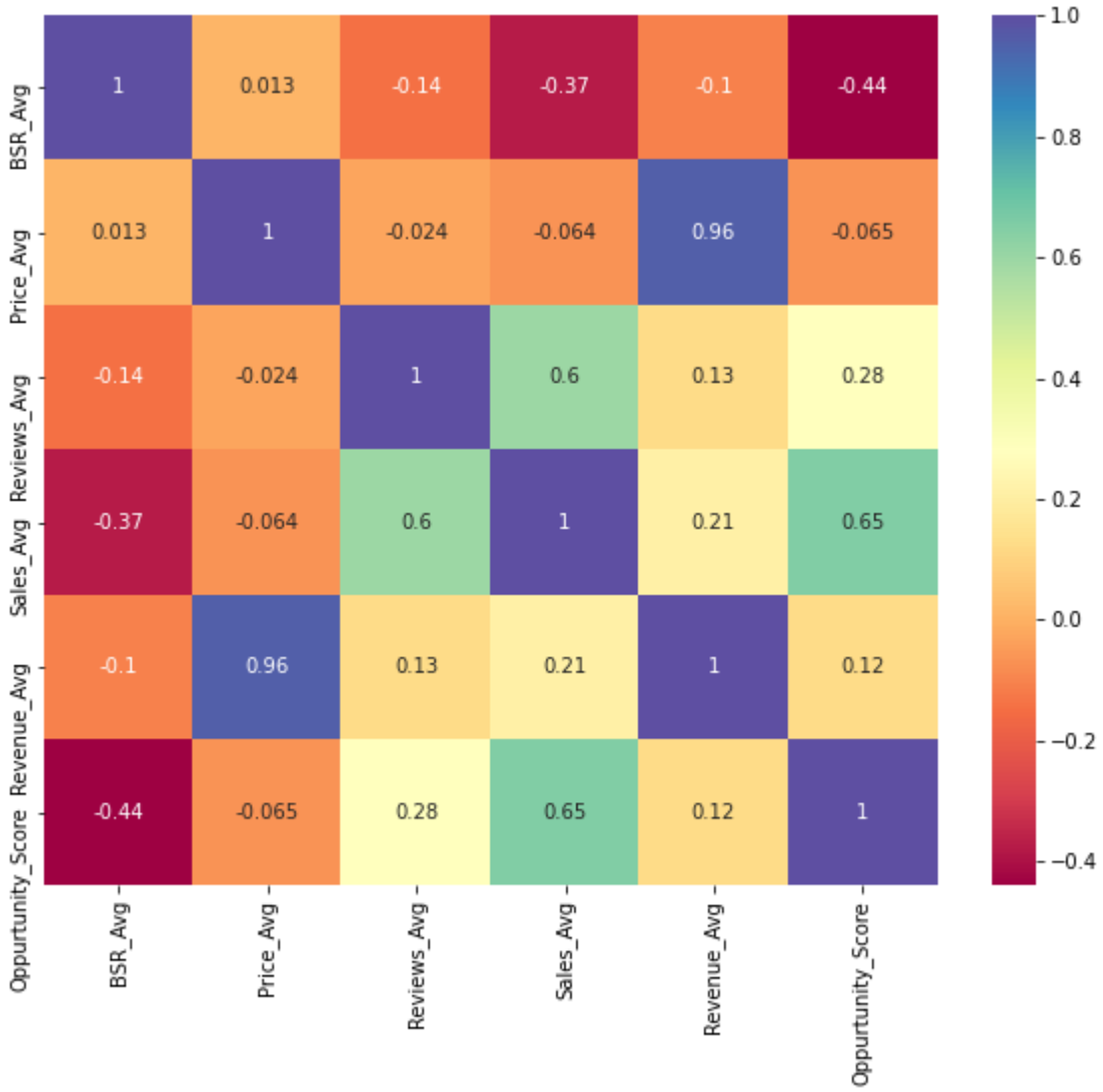
	Keyword	Search_Volume	...	Oppurtunity_Score	Use
0	11children shoes office shoes	0.0	...	0	NO
1	361 shoes white shoes	0.0	...	1	NO
2	a6 shoes running shoes white colour	0.0	...	2	NO
3	aces shoes running shoes	0.0	...	3	YES
4	action shoes for men shoes	0.0	...	2	NO
..	...	...	...	...	...
495	vogue shoes running shoe	NaN	...	1	NO
496	volleyball shoes sports shoes	NaN	...	5	YES
497	wildcraft shoes	NaN	...	1	NO
498	woodland shoes running shoes	NaN	...	3	YES
499	xpert shoes running shoes	NaN	...	5	YES

[500 rows x 21 columns]

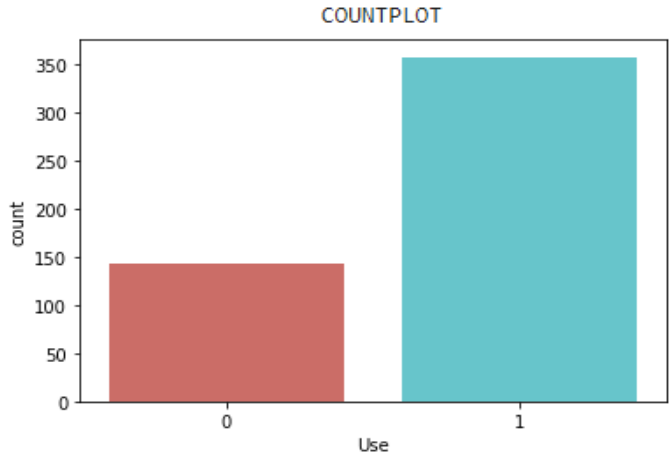


	Keyword	BSR_Avg	...	Oppurtunity_Score	Use
0	11children shoes office shoes	22576.0	...	0	NO
1	361 shoes white shoes	65120.0	...	1	NO
2	a6 shoes running shoes white colour	84810.0	...	2	NO
3	aces shoes running shoes	65085.0	...	3	YES
4	action shoes for men shoes	5301.0	...	2	NO
..	...	...	...	...	...
495	vogue shoes running shoe	14378.0	...	1	NO
496	volleyball shoes sports shoes	376.0	...	5	YES
497	wildcraft shoes	45140.0	...	1	NO
498	woodland shoes running shoes	14021.0	...	3	YES
499	xpert shoes running shoes	2554.0	...	5	YES

[500 rows x 8 columns]



EDA :

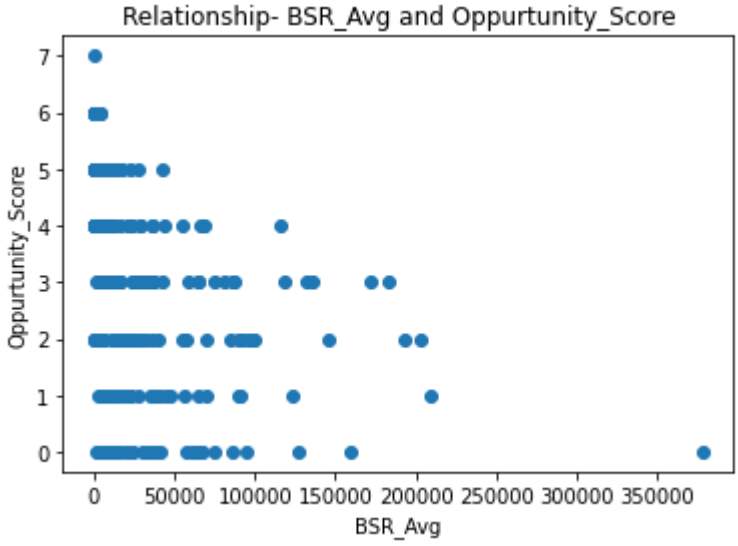



```
NUMBER OF YES AND NO
1    357
0    143
Name: Use, dtype: int64
```

INFERENCE:

From the count plot we can find that there are more keyword acceptance (357) than keyword rejections(143). The dataset shows the interest of the person to include as many keywords as possible which has a decent opportunity.

RELATIONSHIP BETWEEN BSR_Avg AND Opportunity_Score:



INFERNECE:

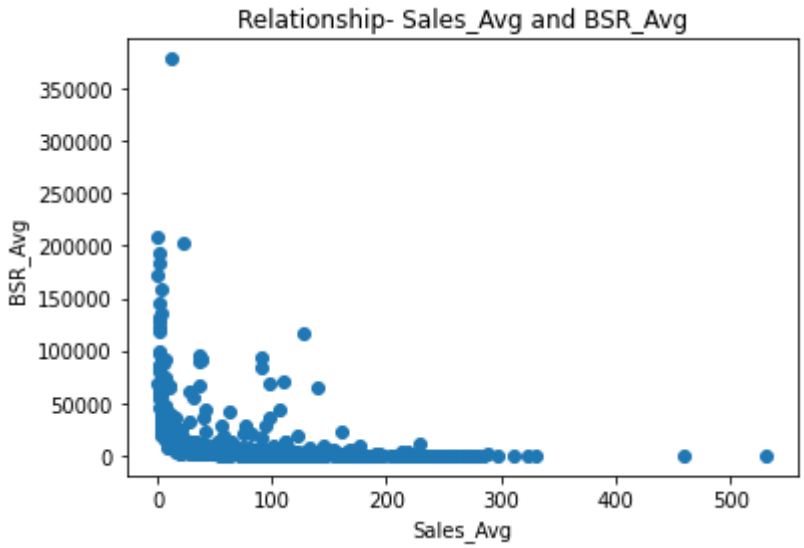
Here, we can find that the probability of rejection increases when the BSR is large and the Opportunity Score is less than 3. We can also find that the graph suggests that most of the keywords has its Average BSR within 100000. We also found the correlation coefficient between these two attributes to be -0.44. This highly suggests a negative correlation between the two variables.

RELATIONSHIP BETWEEN Revenue_Avg and Price_Avg:

INFERENCE:

Here, we can find that there is only one outlier on the top right of the graph. When we consider the rest of the data, we can find the Average price of shoes roughly comes between 200-10000. The Price corresponds to the Revenue as well. Through the heatmap, we find that there lies a strong positive correlation between them with correlation coefficient as 0.96.

RELATIONSHIP BETWEEN SALES_Avg AND BSR_Avg:



INFERENCE:

We can find a datapoint in the top left corner of the graph. Here, we can find the Average sales of shoes roughly comes between 0-300. The Price corresponds to the BSR as well. Through the heatmap, we find that there lies a negative correlation between them with correlation coefficient as -0.37.

RELATIONSHIP BETWEEN SALES_Avg AND Opportunity_Score:

INFERENCE:

When we consider the scattered datapoints, we can find the Average Sales of shoes roughly comes between 0-300. The Price corresponds to the Revenue as well. Through the heatmap, we find that there lies a positive correlation between them with correlation coefficient as 0.65.

FIXING VARIOUS CLASSIFICATION MODELS:

After splitting the model into training and test sets, the dataset is now ready for analysis. We have implemented 5 classification models which are:

1)K – Nearest Neighbour (KNN)

2)Logistic Regression

3)Naive Bayes Classifier

4)Decision Tree Classifier

5)Random Forest Classifier

K – Nearest Neighbour (KNN):

K-Nearest Neighbours (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures). Classification is done by a majority vote to its neighbours. Classification is done by a majority vote of neighbours. If K = 1, then the class is single nearest neighbour. In a common weighting scheme, individual neighbour is assigned to a weight of 1/d if d is the distance to the neighbour. The shortest distance between any two neighbours is always a straight line and the distance is known as Euclidean distance. In the same way the KNN classifies whether the keyword can be accepted or rejected based on the attributes involved in classification.

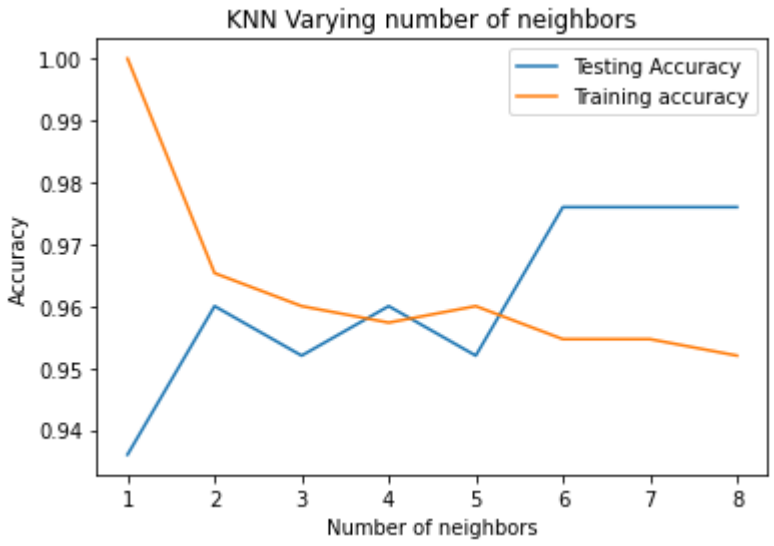
CODE:

```
#KNN MODEL FITTING AND IMPLEMENTATION
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=6)
classifier.fit(X_train, y_train)
neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
#Compute accuracy
for i,k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)
#GRAPH
plt.title('KNN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train,y_train)
print("Accuracy", knn.score(X_test,y_test)*100,"\n")
y_pred = knn.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
confusion_matrix=confusion_matrix(y_test,y_pred)
print(pd.crosstab(y_test, y_pred, rownames=['ACTUAL'], colnames=['PREDI
CTED'], margins=True))
print("\nCONFUSION MATRIX")
print(confusion_matrix)
#PREDICTION AND CLASSIFICATION REPORT
y_pred = classifier.predict(X_test)
yp = (np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len
(y_test),1)),1))
#print("[ 'PREDICTED DECISION' 'ACTUAL DECISION' ]")
#print(yp)
creport = (classification_report(y_test, y_pred))
print('\nClassification Report:\n\n',creport)
print('\nModel Accuracy:', accuracy_score(y_test, y_pred)*100)
print("Training data Accuracy:", classifier.score(X_train, y_train)*100
)
print("Testing data Accuracy:", classifier.score(X_test, y_test)*100)
```

OUTPUT:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```



```
Accuracy 97.6

PREDICTED   0   1  All
ACTUAL
0          36   0   36
1           3  86   89
All        39  86  125

CONFUSION MATRIX
[[36  0]
 [ 3 86]]

Classification Report:

              precision    recall  f1-score   support

           0       0.92      1.00      0.96        36
           1       1.00      0.97      0.98        89

    accuracy                           0.98       125
   macro avg       0.96      0.98      0.97       125
weighted avg       0.98      0.98      0.98       125


Model Accuracy: 97.6
Training data Accuracy: 95.46666666666667
Testing data Accuracy: 97.6
```

INFERENCE:

From the graph we can infer that in the testing accuracy there is a peak when the k value is 6 or more. So we consider the k value is 6 for proceeding the model.

Now comes the classification report which displays the precision, recall, F1 and support scores for the model. Precision Score for class 0 (negative) is 0.92 and for class 1 (positive) is 1.00, indicating the preciseness of the model. Recall value for class 0 is 1.00 and class 1 is 0.97, which describes the amount up – to which the model can predict the output.

From the confusion matrix we can see that there are 36 true positives, 0 false positives, 3 false negatives and finally 86 true negative decisions. The model made did not make those small errors on all the instances of the confusion matrix when compared to rest of the models. The training and test accuracy of

the model is 95.46% and 97.6% respectively. Finally we have come to the accuracy of the model. Here we have the model accuracy of 80.83%. The model has worked in the best manner and classified the keywords with an accuracy of 97.6%. The model is not over-fitted as model optimisation techniques like K fold cross validation is done for 10 folds. The Randomized Search CV techniques used will be explained later.

LOGISTIC REGRESSION:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. It is a widely used model to analyse the relationship between multiple independent variables and one categorical dependent variable. After fixing the model we can see how it performs by looking into its accuracy and precision.

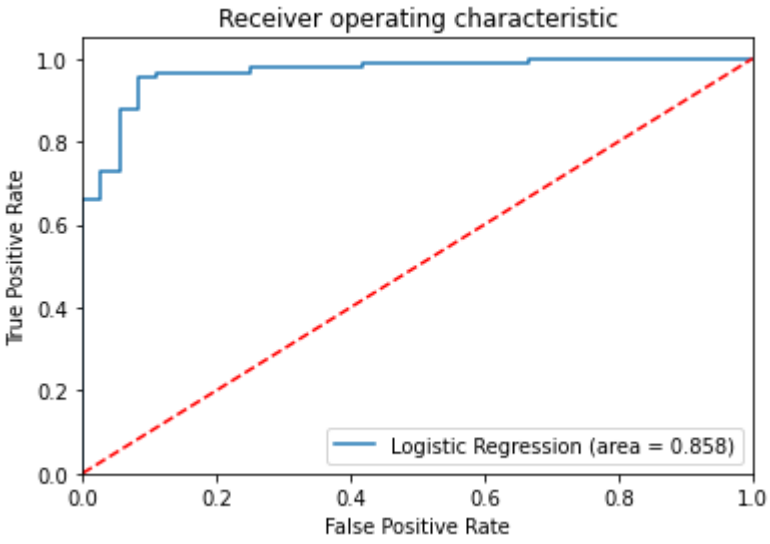
CODE:

```
#LOGISTIC REGRESSINO MODEL FITTING AND IMPLEMENTATION
import statsmodels.api as sm
import pandas.util.testing as tm
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.25, random_state = 19,stratify=y)
logit_model=sm.Logit(y,X)
result=logit_model.fit()
#print(result.summary2())
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(solver='liblinear', random_state=24)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", confusion_matrix)
#PREDICTION AND CLASSIFICATION REPORT
y_pred = classifier.predict(X_test)
yp = (np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len
(y_test),1)),1))
creport = (classification_report(y_test, y_pred))
print('\nClassification Report:\n\n',creport)
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
#ROC_AUC_GRAPH
import sklearn.linear_model as sk
logreg = sk.LogisticRegressionCV()
logreg.fit(X_train,y_train)
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:
,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.3f)' % logit_r
oc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
#LOGISTIC REGRESSION OPTIMIZED RESULTS
logistic_reg = LogisticRegression(solver='liblinear', random_state=19)
precision_average = precision_score(y_test, y_pred, average="weighted",
 pos_label=1)
recall_average = recall_score(y_test, y_pred, average="weighted", pos_l
abel=1)
f1_score_average = f1_score(y_test,y_pred,average="weighted",pos_label=
1)
print("CLASSIFICATION REPORT OPTIMIZATION RESULTS \n\n")
print("Best Precision : ",precision_average)
print("Best Recall : ", recall_average)
print("Best f1_score : ",f1_score_average)
```

OUTPUT:

```
Optimization terminated successfully.
         Current function value: 0.210412
         Iterations 10
Confusion Matrix :
 [[28  8]
 [ 2 87]]

Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.78      0.85        36
           1       0.92      0.98      0.95        89

    accuracy                           0.92       125
   macro avg       0.92      0.88      0.90       125
weighted avg       0.92      0.92      0.92       125


CLASSIFICATION REPORT OPTIMIZATION RESULTS


Best Precision :  0.9208421052631579
Best Recall :  0.92
Best f1_score :  0.9176679841897234
```



INFERENCE:

The classification report displays the precision, recall, F1 and support scores for the model. Precision Score for class 0 (negative) is 0.93 and for class 1 (positive) is 0.92 .Recall value for class 0 is 0.92 and class 1 is 0.98, which describes the amount up to which the model can predict the output.

From the Confusion matrix we can see that there are 28 true positives, 8 false positive and 2 false negative and finally 87 true negative values. The model makes errors and predicted the values. We have the model accuracy of 92%. The model has worked well but it is not as good as a KNN classifier model.

We have also optimized the precision , recall and f1-score of the classification model with pos label as 1 which denotes the "YES" in the decision attribute. We do this because , we are more concerned about including a keyword which are a potential hit for our campaign as sponsored ads. The reliable Precision Score, Recall and f1 score is calculated by taking a weighted average of all their respective scores obtained for all possible testing sets. The obtained results are Best Precision : 0.920,Best Recall : 0.92,Best f1_score : 0.917 for the model built by using logistic regression.

NAIVE BAYES CLASSIFIER:

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. Let us look into the classification results obtained by using this model.

CODE:

```
#NAIVE BAYES MODEL FITTING AND IMPLEMENTATION
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,classification_report
model = GaussianNB()
model.fit(X_train, y_train);
y_pred = model.predict(X_test)
yp = (np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len
(y_test),1)),1))
confmat = confusion_matrix(y_test,y_pred)
print("\nConfusion matrix:\n\n",confmat)
creport = (classification_report(y_test, y_pred))
print('\nClassification Report:\n\n',creport)
print('Model Accuracy:', accuracy_score(y_test, y_pred)*100)
print("Training data Accuracy:", classifier.score(X_train, y_train)*100
)
print("Testing data Accuracy:", classifier.score(X_test, y_test)*100)
#NAIVE BAYES CLASSIFIER - OPTIMIZED RESULTS
precision_average = precision_score(y_test, y_pred, average="weighted",
 pos_label=1)
recall_average = recall_score(y_test, y_pred, average="weighted", pos_l
abel=1)
f1_score_average = f1_score(y_test,y_pred,average="weighted",pos_label=
1)
print("CLASSIFICATION REPORT OPTIMIZATION RESULTS \n\n")
print("Best Precision : ",precision_average)
print("Best Recall : ", recall_average)
print("Best f1_score : ",f1_score_average)
```

OUTPUT:

```
Confusion matrix:

 [[34  2]
 [11 78]]

Classification Report:

              precision    recall  f1-score   support

           0       0.76      0.94      0.84        36
           1       0.97      0.88      0.92        89

    accuracy                           0.90       125
   macro avg       0.87      0.91      0.88       125
weighted avg       0.91      0.90      0.90       125

Model Accuracy: 89.60000000000001
Training data Accuracy: 88.8
Testing data Accuracy: 92.0
```

```
CLASSIFICATION REPORT OPTIMIZATION RESULTS


Best Precision :  0.9117999999999999
Best Recall :  0.896
Best f1_score :  0.8990085470085472
```

INFERENCE:

The classification report displays the precision, recall, F1 and support scores for the model. Precision Score for class 0 (negative) is 0.76 and for class 1 (positive) is 0.97 .Recall value for class 0 is 0.94 and class 1 is 0.88, which describes the amount up to which the model can predict the output. The training and test accuracy of the model is 88.8% and 92.0% respectively. Finally we have come to the accuracy of the model. Here we have the model accuracy of 89.6%.

From the Confusion matrix we can see that there are 34 true positives, 2 false positive and 11 false negative and finally 28 true negative values. The model makes errors and predicted the values. We have the model accuracy of 90%. The model has worked but it is not as good as a KNN classifier model.

The reliable Precision Score, Recall and f1 score is calculated by taking a weighted average of all their respective scores obtained for all possible testing sets. The obtained results are Best Precision : 0.912,Best Recall : 0.896,Best f1_score : 0.899 for the model built by using Naïve Bayes Classifier.

DECISION TREE CLASSIFIER:

Decision Tree is one of the classification algorithms, which is used to solve regression and classification problems. The general objective of using Decision Tree is to create a model that predicts classes or values of target variables by generating decision rules derived from training data sets. Decision tree algorithm follows a tree structure with roots, branches and leaves. The attributes of decision making are the internal nodes and class labels are

represented as leaf nodes. Let us look into the classification results obtained by using this model.

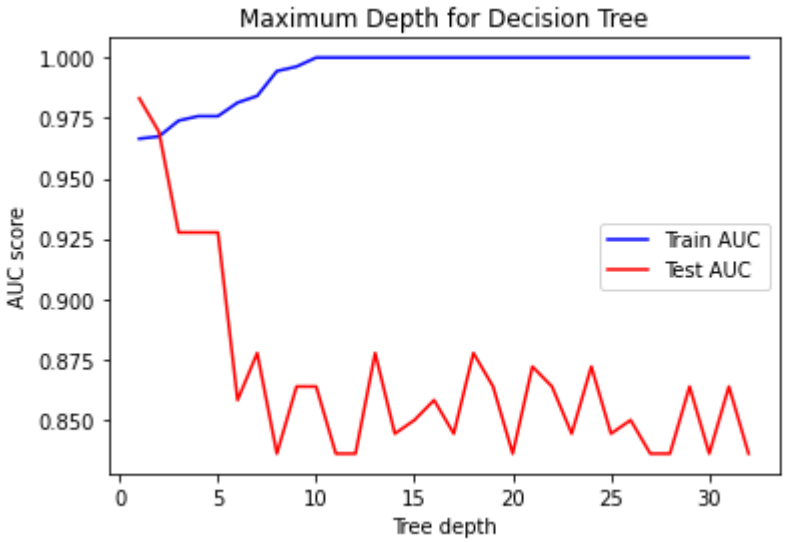
CODE:

```
#DECISION TREE CLASSIFIER MODEL IMPLEMENTATION AND FITTING
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'gini', random_state =
19)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, accuracy_score, precision
_score,classification_report
cm = confusion_matrix(y_test, y_pred)
print('\n\nClassification Report\n\n', classification_report(y_test, y_
pred))
print('\n\nConfusion Matrix\n', confusion_matrix(y_test, y_pred))
print('\nModel Accuracy:', accuracy_score(y_test, y_pred)*100)
print("Training data Accuracy:", classifier.score(X_train, y_train)*100
)
print("Testing data Accuracy:", classifier.score(X_test, y_test)*100)

from sklearn.metrics import roc_curve, auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,
 y_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

max_depths = np.linspace(1, 32, 32, endpoint=True)
train_results = []
test_results = []
for max_depth in max_depths:
  dt = DecisionTreeClassifier(max_depth=max_depth)
  dt.fit(X_train, y_train)
  train_pred = dt.predict(X_train)
  false_positive_rate, true_positive_rate, thresholds = roc_curve(y_tra
in, train_pred)
  roc_auc = auc(false_positive_rate, true_positive_rate)
  # Add auc score to previous train results
  train_results.append(roc_auc)
  y_pred = dt.predict(X_test)
  false_positive_rate, true_positive_rate, thresholds = roc_curve(y_tes
t, y_pred)
  roc_auc = auc(false_positive_rate, true_positive_rate)
  # Add auc score to previous test results
  test_results.append(roc_auc)
from matplotlib.legend_handler import HandlerLine2D
line1 = plt.plot(max_depths, train_results, 'b', label='Train AUC')
```

```
line2 = plt.plot(max_depths, test_results, 'r', label='Test AUC')
plt.title("Maximum Depth for Decision Tree")
plt.legend()
plt.ylabel('AUC score')
plt.xlabel('Tree depth')
plt.show()
#DECISION TREE CLASSIFIER - OPTIMIZED RESULTS
precision_average = precision_score(y_test, y_pred, average="weighted",
 pos_label=1)
recall_average = recall_score(y_test, y_pred, average="weighted", pos_l
abel=1)
f1_score_average = f1_score(y_test,y_pred,average="weighted",pos_label=
1)
print("CLASSIFICATION REPORT OPTIMIZATION RESULTS \n\n")
print("Best Precision : ",precision_average)
print("Best Recall : ", recall_average)
print("Best f1_score : ",f1_score_average)
```

OUTPUT:

```
Classification Report

              precision    recall  f1-score   support

           0       0.93      0.75      0.83        36
           1       0.91      0.98      0.94        89

    accuracy                           0.91       125
   macro avg       0.92      0.86      0.89       125
weighted avg       0.91      0.91      0.91       125


Confusion Matrix
 [[27  9]
 [ 2 87]]

Model Accuracy: 91.2
Training data Accuracy: 100.0
Testing data Accuracy: 91.2

 CLASSIFICATION REPORT OPTIMIZATION RESULTS


 Best Precision :  0.898748299319728
 Best Recall :  0.896
 Best f1_score :  0.89107410236822
```



INFERENCE:

Gini criterion is used to build the decision tree classifier. It uses information gain to decide which attribute it should choose to branch the tree. The training and test accuracy of the model is 100% and 91.2% respectively. Finally we have come to the accuracy of the model. Here we have the model accuracy of 91.2%.

The graph shows the maximum depth of the training and the test dataset considering various Tree depth and the corresponding AUC (Area Under the Curve)score.

The classification report displays the precision, recall, F1 and support scores for the model. Precision Score for class 0 (negative) is 0.93 and for class 1 (positive) is 0.91 .Recall value for class 0 is 0.75 and class 1 is 0.98, which describes the amount up to which the model can predict the output.

From the Confusion matrix we can see that there are 27 true positives, 8 false positive and 2 false negative and finally 87 true negative values. The model makes errors and predicted the values. We have the model accuracy of 91.2%. The model has worked well but it is not as good as a KNN classifier model.

The reliable Precision Score, Recall and f1 score is calculated by taking a weighted average of all their respective scores obtained for all possible testing sets. The obtained results are Best Precision : 0.898,Best Recall : 0.896,Best f1_score : 0.891 for the model built by using Decision Tree classifier.

RANDOM FOREST CLASSIFIER:

Random forest algorithm constructs multiple decision trees to act as an ensemble of classification and regression process. A number of decision trees are constructed using a random subsets of the training data sets. A large collection of decision trees provide higher accuracy of results. The runtime of the algorithm is comparatively fast and also accommodates missing data. Random forest randomizes the algorithm and not the training data set. The decision class is the mode of classes generated by decision trees. Let us look into the classification results obtained by using this model.

CODE:

```
#RANDOM FOREST CLASSIFIER MODEL IMPLEMENTATION AND FITTING
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'gin
i', random_state = 19)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, accuracy_score, precision
_score,classification_report
print('\n\nClassification Report\n\n', classification_report(y_test, y_
pred))
print('\nConfusion Matrix\n', confusion_matrix(y_test, y_pred))
print('Model Accuracy', accuracy_score(y_test, y_pred)*100)
print("Training Accuracy", classifier.score(X_train, y_train)*100)
print("Testing Accuracy", classifier.score(X_test, y_test)*100)
#RANDOM FOREST CLASSIFIER - OPTIMIZED RESULT
precision_average = precision_score(y_test, y_pred, average="weighted",
 pos_label=1)
recall_average = recall_score(y_test, y_pred, average="weighted", pos_l
abel=1)
f1_score_average = f1_score(y_test,y_pred,average="weighted",pos_label=
1)
```

```
print("CLASSIFICATION REPORT OPTIMIZATION RESULTS \n\n")
print("Best Precision : ",precision_average)
print("Best Recall : ", recall_average)
print("Best f1_score : ",f1_score_average)
```

OUTPUT:

```
Classification Report

              precision    recall  f1-score   support

           0       0.92      0.94      0.93        36
           1       0.98      0.97      0.97        89

    accuracy                           0.96       125
   macro avg       0.95      0.96      0.95       125
weighted avg       0.96      0.96      0.96       125


Confusion Matrix
 [[34  2]
 [ 3 86]]
Model Accuracy 96.0
Training Accuracy 99.73333333333333
Testing Accuracy 96.0


 CLASSIFICATION REPORT OPTIMIZATION RESULTS


 Best Precision :  0.9604668304668306
 Best Recall :  0.96
 Best f1_score :  0.9601609782524573
```

INFERENCE:

Gini criterion is used to build the Random Forest classifier. It uses information gain to decide which attribute it should choose to branch the tree. The training and test accuracy of the model is 99.73% and 96% respectively. Finally we have come to the accuracy of the model. Here we have the model accuracy of 96%.

The classification report displays the precision, recall, F1 and support scores for the model. Precision Score for class 0 (negative) is 0.92 and for class 1 (positive)

is 0.98 .Recall value for class 0 is 0.94 and class 1 is 0.97, which describes the amount up to which the model can predict the output.

From the Confusion matrix we can see that there are 34 true positives, 2 false positive and 3 false negative and finally 86 true negative values. The model does not make errors like Naïve bayes , Logistic or Decision Tree classifiers and thus predicts the values. We have the model accuracy of 96%. The model has worked really well but KNN classifier model still has better precision score and accuracy.

The reliable Precision Score, Recall and f1 score is calculated by taking a weighted average of all their respective scores obtained for all possible testing sets. The obtained results are Best Precision : 0.960,Best Recall : 0.960,Best f1_score : 0.960 for the model built by using Random Forest classifier.

HYPERPARAMETER TUNING:

We found the best model for our problem is the KNN classifier with an accuracy of 97.6%. We shall use Randomized Search CV technique to finetune the results.

CODE:

```
#Model Optimization(Hyper Parameter Tuning Using Randomized Search Cros
s Validation)
knn = KNeighborsClassifier()
params = {'n_neighbors':np.arange(2,11,step=1),'metric':['minkowski','m
anhattan','euclidean']}
random_search = RandomizedSearchCV(knn,params,scoring='precision',cv=10
)
precision_average = precision_score(y_test, y_pred, average="weighted",
 pos_label=1)
recall_average = recall_score(y_test, y_pred, average="weighted", pos_l
abel=1)
f1_score_average = f1_score(y_test,y_pred,average="weighted",pos_label=
1)
random_search.fit(X_train,y_train)
print("RANDOM SEARCH RESULTS \n\n")
print("Best Params : ",random_search.best_params_)
print("Best Precision : ",precision_average)
print("Best Recall : ", recall_average)
```

```
print("Best f1_score : ",f1_score_average)
print("Best Model: \n",random_search.best_estimator_)
tuned_model = random_search.best_estimator_
```

OUTPUT:

```
RANDOM SEARCH RESULTS


Best Params :  {'n_neighbors': 4, 'metric': 'manhattan'}
Best Precision :  0.9778461538461538
Best Recall :  0.976
Best f1_score :  0.9762742857142857
Best Model:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='manhattan',
                      metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                      weights='uniform')
```

INFERENCE:

The KNN model is optimized with the best Parameters , best Precision, Best recall , Best f1 score and Best model with the number of neighbours to be considered. Thus, this model can be finally used.

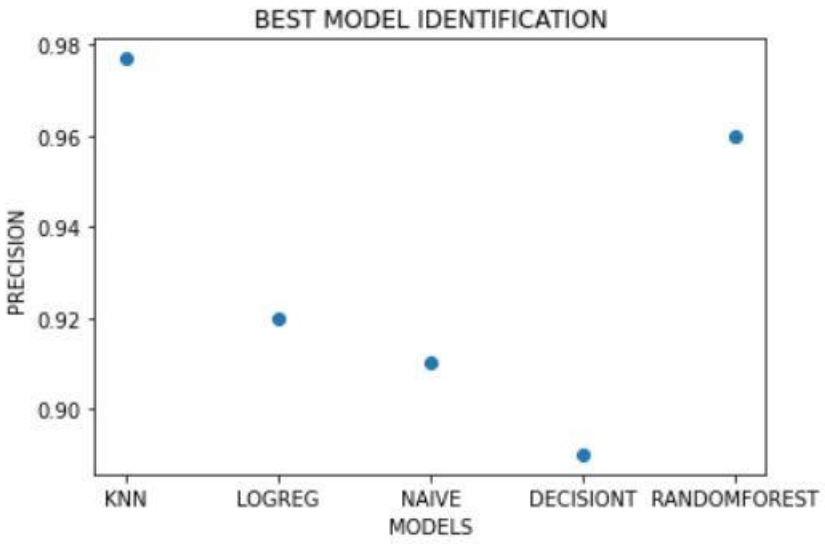
BEST MODEL IDENTIFICATION:

Now all the classifiers have been fitted completely to the dataset. Let us clearly see accuracy of all the models, which can help us to infer the best model.

CODE:

```
#BEST MODEL IDENTIFICATION
models=("KNN","LOGREG","NAIVE","DECISIONT","RANDOMFOREST")
precisionformodels=(0.977,0.92,0.91,0.89,0.96)
pyplot.scatter(models,precisionformodels)
pyplot.title("BEST MODEL IDENTIFICATION")
pyplot.xlabel("MODELS")
pyplot.ylabel("PRECISION")
pyplot.show()
```

OUTPUT:



INFERENCE:

Here, we have thus compared the precision of all models and we can find that KNN has the best precision score. We consider only the precision because We do this because , we are more concerned about including a keyword which are a potential hit for our campaign as sponsored ads.

CONCLUSION:

The results of these predictions can be added to the domain of SEO and can be used for providing suggestions in the domain by making it easy for professionals in identifying the best keywords to use for Campaign. Future work should mainly focus on implementing more big data oriented tools and techniques which makes the process much faster and effective. The growing number of keywords demands the same.