Northeastern

College of Professional Studies
Northeastern University San Jose

MPS Analytics

Course: ALY6020

Assignment:

Module 3 – Project

Submitted on:

October 13, 2023

Submitted to:

Professor: Ahmadi Behzad

Submitted by:

Heejae Roh

Introduction

This dataset mainly contains content related to marketing. I will use ‘response’ as the target variable. ‘response’ indicates 'if customer accepted the offer in the last campaign, 0 otherwise'. I will focus on data preprocessing and reorganizing data with many columns. Next, I will apply logistic regression and SVM (Support Vector Machine) models and compare the results. In the process of analysis, let's think about how our model can be used for marketing purposes.

Abstraction

1. proper data cleansing techniques to ensure that you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data.

I focused on data preprocessing in this analysis. Data preprocessing is a crucial step in machine learning that involves cleaning and transforming the data before feeding it into the model. In logistic regression, data preprocessing can help improve the model accuracy by removing noise, filling in missing values, and normalizing the data (saturncloud, n.d.).

1. I changed year_birth and dt_customer to age and enrollment_days based on how far the dates are from the current point.
2. All Mnt___ columns representing the amount of spending are grouped into spending. marital_status and kidhome+teenhome were changed to binary variables of 0 and 1.
3. Education was changed from a categorical column to a numerical variable indicating the period of education.
4. Finally, outliers in age, income, and spending were checked, and values that were significantly higher than the second largest income (113K), which was 666K in income, were removed as outliers.

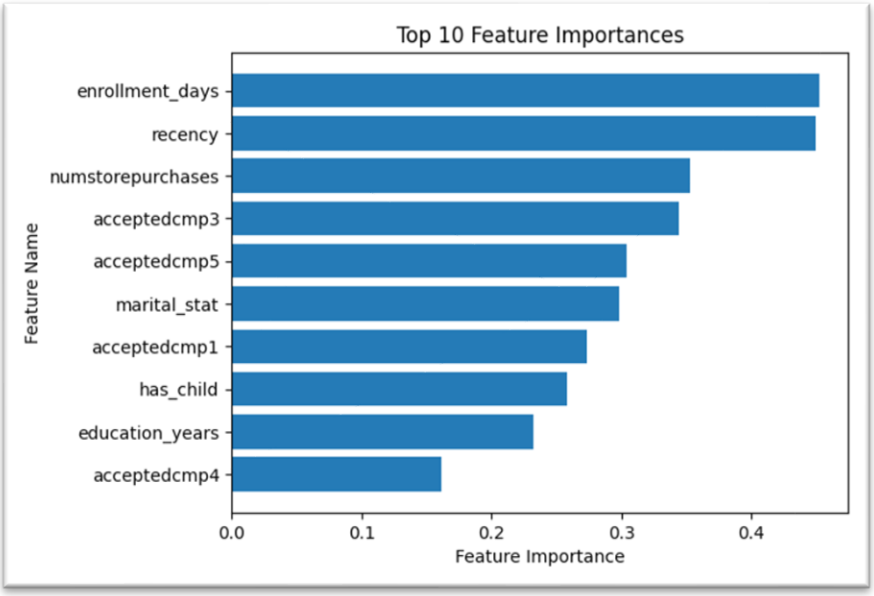
2. Build a logistic model to accurately predict subscription behavior. Discuss which variables are significant, their business impact, and how that may help you learn about the business.

Column Name	coefficient	p-value	Rank of P-value
recency	-0.0296	9.53e-20	1
accepetedcmp3	2.2276	7.57e-19	2
enrollment_days	0.0041	9.08e-16	3
marital_stat	-1.3159	4.29e-13	4
accetedcmp5	1.7684	4.12e-09	5

Most negative influence: recency > marital_stat
Most positive influence: acceptedcmp3 > enrollment_days > acceptedcmp5

1. Analyzing only the business part, recency with a p-value rank of 1 means that the closer the last purchase time is, the higher the probability of response is. This means that getting consumers to continue making purchases is the most important factor in responding. In other words, you need to think about ways to reduce recency rather than increase it from a marketing perspective.
2. acceptedcmp3 indicates whether the 3rd campaign was accepted. Through this, we can further analyze what kind of marketing was done in the third campaign and proceed with future campaigns based on this. acceptedcmp5, which appears as rank 5, can also be analyzed through a similar process.
3. enrollment_days means that older customers are more likely to respond, and it could mean that we have a lot of loyal customers, but it requires additional collaboration with the marketing department.
4. marital_stat means that the response rate is higher for unmarried consumers, and companies should prepare marketing to target unmarried consumers.

3. Build an SVM model to accurately predict subscription behavior. Discuss which variables are significant, their business impact, and how that may help you learn about the business.

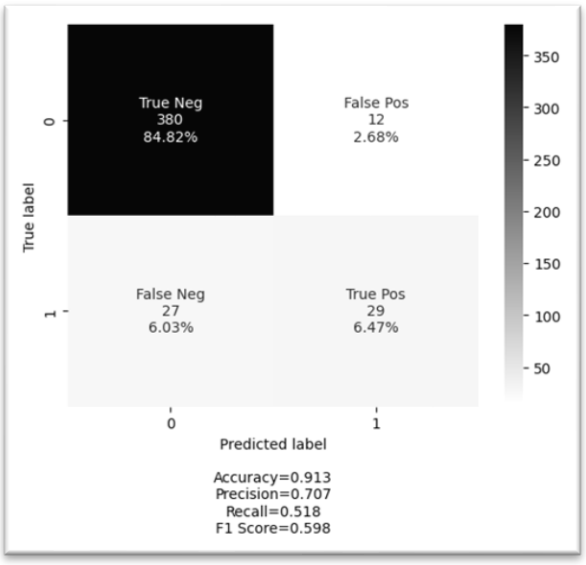


Most significant influence: enrollment_days > recency > numstorepurchases > acceptedcmp3, 5 in a row

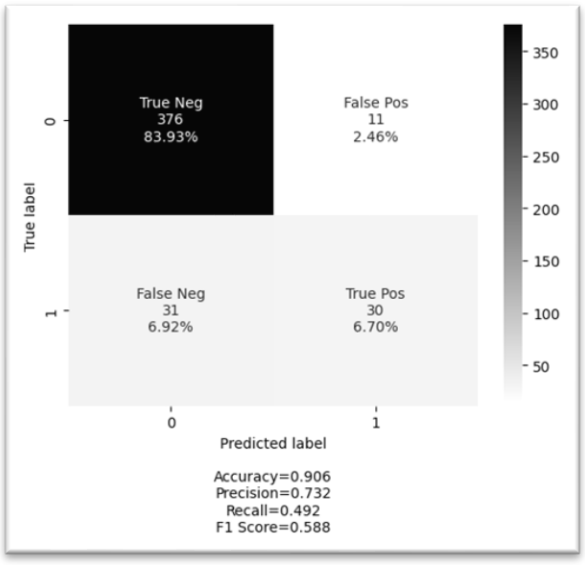
1. As a result of identifying the top 10 feature imports in the SVM model, enrollment_days was found to be the most significant in the SVM model. ‘When you became a customer: enrollment_days’ had the greatest influence on the response.
2. Next, recency, which represents 'Number of days since the last purchase', was found to have a strong impact, as shown as the first significant variable in the previously logistic regression.
3. The top 10 influence of SVM model includes numstorepurchases. This attribute represents 'Number of purchases made directly in stores', in other words, the number of times a customer visited the store directly and made a purchase had a significant impact on the response. We can conduct marketing by targeting customers with this high frequency.
4. We can see that marital_stat and acceptedcmp, which were influential attributes in the logistic regression previously, have emerged as important factors again.

4. Compare the accuracy of both models (overall accuracy, precision, recall) and the overall variables that were deemed significant. Discuss which model you would recommend based upon these three metrics.

Logistic Regression



SVM



The accuracy of the logistic regression model was 0.906, with a precision of 0.732, recall of 0.492, and an F1 score of 0.588. Notably, the low recall was due to the disproportionate presence of '0' data compared to '1' data in the dataset.

Comparatively, the SVM model exhibited an accuracy of 0.888, slightly lower than the logistic regression. However, its precision was notably higher at 0.788, an increase of 0.56. Conversely, the recall dropped to 0.377, a decrease of 0.115. The resulting F1-score was 0.510, a decrease of 0.078 compared to the logistic regression model.

Based on the above results, we can see that the Logistic regression model has higher accuracy than the SVM model. When evaluating these metrics, we should not simply focus on the size of the numbers, but rather decide which metrics are important based on the specific goals and constraints of the problem we are solving.

- Accuracy: Logistic regression outperforms SVM in terms of overall accuracy, indicating that it makes correct predictions for a higher proportion of the total dataset.
- Precision: SVM has a higher precision, meaning that when it predicts positive instances, it is more likely to be correct. In other words, when SVM predicts a positive outcome, it is often accurate.
- Recall: Logistic regression has a higher recall, indicating that it is better at identifying all relevant instances in the dataset. It has a lower chance of missing positive cases.

Considering these metrics, if the goal is to maximize the overall correctness of predictions (balance between precision and recall), the logistic regression model might be preferred due to its higher accuracy and balanced recall and precision. However, if minimizing false positives is crucial (high precision is a priority), then the SVM model might be the choice despite its slightly lower accuracy and recall.

Dataset Understanding

To predict subscription behavior, I will use ‘response’ as a target variable. Response is the column about customer accepting the offer in the last campaign or not. There are 29 columns and 2240 entries. It is assumed that various data were gathered to predict whether a customer will subscribe (accept campaign offer or not).

Description of the variables/features in the dataset.

#	column name	Description
1	**ID**	Customer's id
2	**Year_Birth**	Customer's year of birth
3	**Education**	Customer’s level of education
4	**Marital_Status**	Customer’s marital status
5	**Income**	Customer’s yearly household income
6	**Kidhome**	Number of small children in customer’s household
7	**Teenhome**	Number of teenagers in customer’s household
8	**Dt_Customer**	Date of customer’s enrollment with the company
9	**Recency**	Number of days since the last purchase
10	**MntWines**	Amount spent on wine products in the last 2 years
11	**MntFruits**	Amount spent on fruits products in the last 2 years
12	**MntMeatProducts**	Amount spent on meat products in the last 2 years
13	**MntFishProducts**	Amount spent on fish products in the last 2 years
14	**MntSweetProducts**	Amount spent on sweet products in the last 2 years

15	**MntGoldProds**	Amount spent on gold products in the last 2 years
16	**NumDealsPurchases**	Number of purchases made with discount
17	**NumWebPurchases**	Number of purchases made through company's web site
18	**NumCatalogPurchases**	Number of purchases made using catalogue
19	**NumStorePurchases**	Number of purchases made directly in stores
20	**NumWebVisitsMonth**	Number of purchases made through company's web site
21	**AcceptedCmp3**	1 if customer accepted the offer in the 3rd campaign, 0 otherwise
22	**AcceptedCmp4**	1 if customer accepted the offer in the 4th campaign, 0 otherwise
23	**AcceptedCmp5**	1 if customer accepted the offer in the 5th campaign, 0 otherwise
24	**AcceptedCmp1**	1 if customer accepted the offer in the 1st campaign, 0 otherwise
25	**AcceptedCmp2**	1 if customer accepted the offer in the 2nd campaign, 0 otherwise
26	**Complain**	1 if customer complained in the last 2 years
27	**Z_CostContact**	Cost to contact a customer
28	**Z_Revenue**	Revenue after client accepting campaign
29	**Response**	1 if customer accepted the offer in the last campaign, 0 otherwise

Headtail of Dataset 1

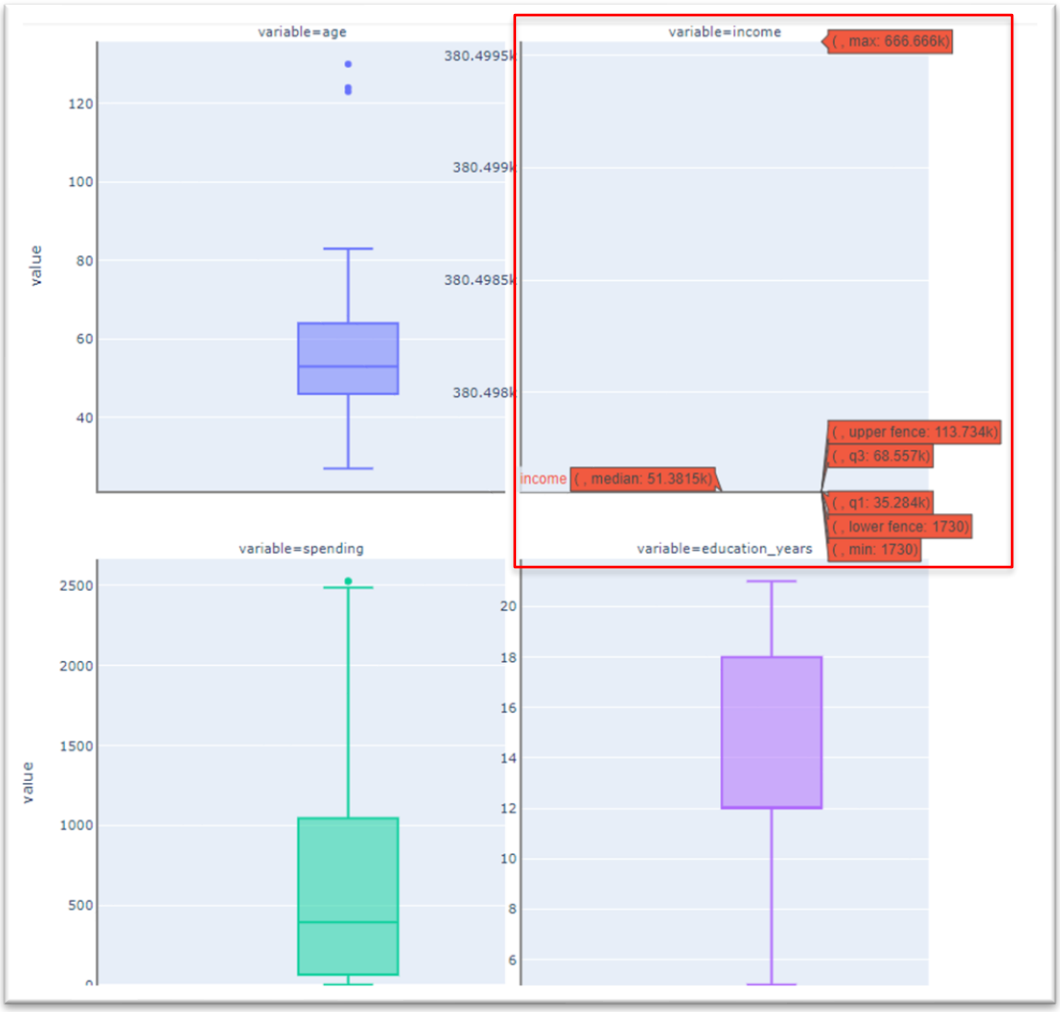
id	age	experience	income	zip_code	family	ccavg	education
1	25	1	49	91107	4	1.6	1
2	3	19	34	90089	3	1.5	1
3	39	15	11	94720	1	1.0	1
…				…			
4998	63	39	24	93023	2	0.3	3
4999	65	40	49	90034	3	0.5	2
5000	28	4	83	92612	3	0.8	1

Headtail of Dataset 2

id	mortgage	loan	securities_account	cd_account	online	creditcard
1	0	0	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	0	0
…			…			
4998	0	0	0	0	0	0
4999	0	0	0	0	1	0
5000	0	0	0	0	1	1

Data Cleansing

1. After looking at the data, I decided to combine or organize columns and clarify the data, referring to data pre-processing by RAPHAEL (2020) in Kaggle.
2. First, year_birth was changed to age by subtracting the year based on 2023.
3. From MntWine to MntSweetProducts, which is the amount of spending for each product over the two years, it was added up and changed to 'spending'.
4. I changed the values of marital_status such as divorced, single, absurd, widow, and yolo to 0, meaning not married, and together and married to 1.
5. kidhome and teen home were considered as whether or not there was a child or children, so they were combined into 1 if there was a baby, and 0 if there was no child.
6. I changed the categorical variables in the education column to approximate numbers. Basic has the lowest value of 5 and PhD has the highest value of 21. The different education levels were designated by numbers between 5 and 21.
7. dt_customer calculates how many days the customer has been maintained by calculating the current date.
8. Outliers in age, income, and spending were checked, and only income 666k was judged to be too high than the second maximum income of 113k, so it was decided to exclude it and leave most other values.
9. For missing values, 24 values for income were estimated and filled in using income, age, and education_years.



Exploratory Data Analysis

Descriptive Analysis of Dataset 1

	income	recency	numdealspurchases	numwebpurchases	numcatalogpurchases
count	2239	2239	2239	2239	2239
mean	51964.8	49.12	2.32	4.08	2.66
std	21432.2	28.96	1.93	2.77	2.92
min	1730	0	0	0	0
25%	35428.5	24	1	2	0
50%	51390	49	2	4	2
75%	68277.5	74	3	6	4
max	162397	99	15	27	28

1. Excluding one value whose income exceeds 600k, 2239 rows remain from the first 2400.
2. The mean of income was higher than 50k.
3. The recency, which indicates the number of days since the last purchase, recorded a mean of 49.12.
4. num___purchases represents the number purchased in each environment, with delas being 2.32, web being 4.08, and catalog being 2.66, showing the highest number of web purchases.

Descriptive Analysis of Dataset 2

	numstorepurchases	numwebvisitsmonth	acceptedcmp3	acceptedcmp4	acceptedcmp5
count	2239	2239	2239	2239	2239
mean	5.79	5.31	0.0728	0.0745	0.0728
std	3.25	2.42	0.2598	0.2627	0.2598
min	0	0	0	0	0
25%	3	3	0	0	0
50%	5	6	0	0	0
75%	8	7	0	0	0
max	13	20	1	1	1

Descriptive Analysis of Dataset 3

	complain	z_costcontact	z_revenue	response	age	spending	marital_stat
count	2239	2239	2239	2239	2239	2239	2239
mean	0.009379	3	11	0.1491	54.19	606.04	0.6444
std	0.096412	0	0	0.3563	11.98	602.27	0.4787
min	0	3	11	0	27	5	0
25%	0	3	11	0	46	69	0
50%	0	3	11	0	53	396	1
75%	0	3	11	0	64	1046	1
max	1	3	11	1	130	2525	1

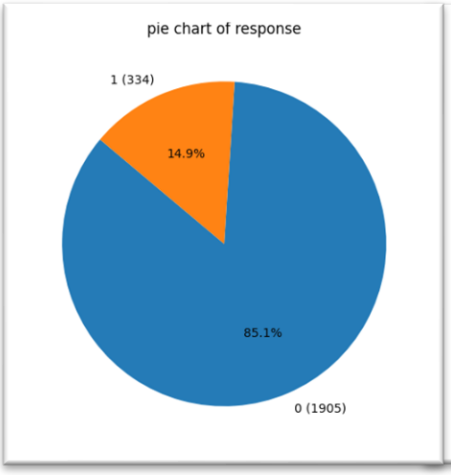
Descriptive Analysis of Dataset 4

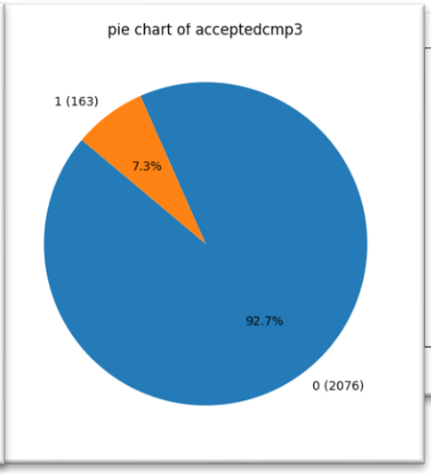
	has_child	education_years	enrollment_days
count	2239	2239	2239
mean	0.7150	14.4135	3746.5649
std	0.4514	4.5107	202.1660
min	0	5	3393
25%	0	12	3573.5
50%	1	12	3748
75%	1	18	3922
max	1	21	4092

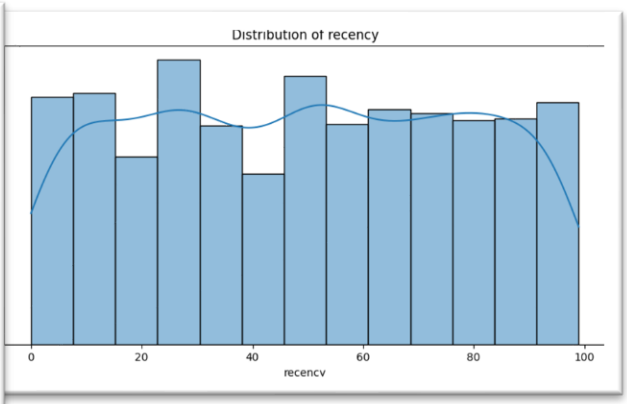
5. Among the above data, z_costcontact had the same value of 3, and z_revenue had the same value of 11. Therefore, it was finally decided to exclude this data.
6. Although it is not specified exactly when the data was created, age and enrollment_days are currently calculated based on 2023, so rather than comparing exact numbers, we will look at trends in large and small numbers.

Data Visualizations

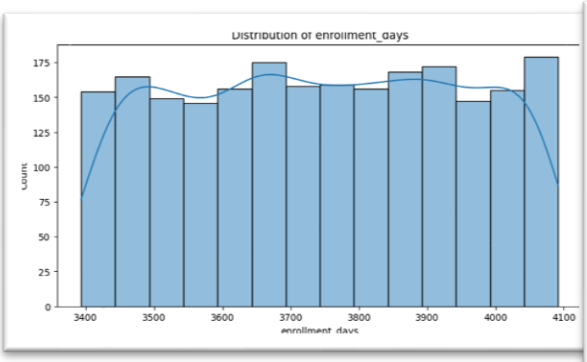
Histograms & Pie chart of primary attribute for logistic regression & SVM

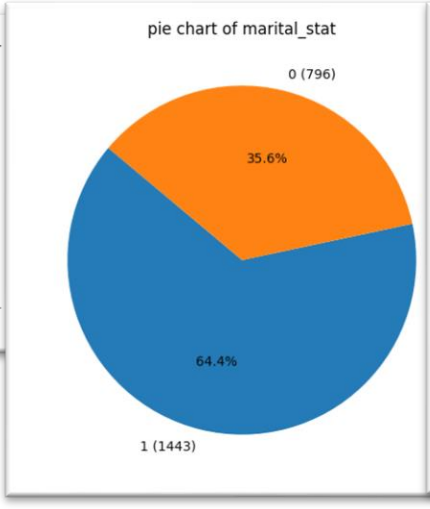


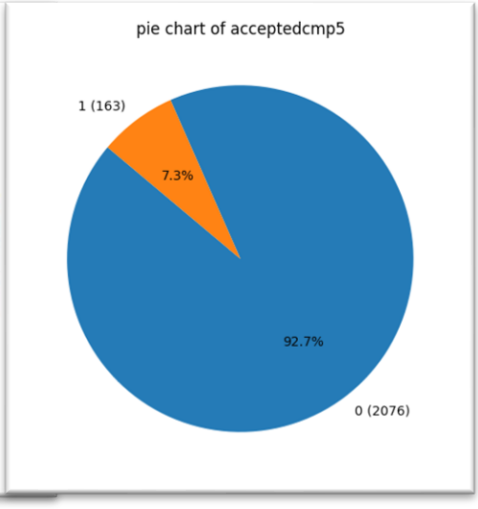




The target variable, response, is imbalanced data with non-response 0 accounting for 85.1% or 1905, and response rate 1 recording 14.9%, or 334 values. The pie chart of acceptedcmp3 also shows that 0, which is the percentage of marketing not accepted, accounts for 92.7%. The distribution of recency is evenly distributed between 0 and 100, and the mode is located between 20 and 30.







Enrollment_days is also evenly distributed from 3140 to 4100. Because it is based on the time from when data was collected to the present, absolute values are not very meaningful, and it is important to understand how the data is distributed. Marital_stat recorded the majority of married people as '1', at 64.4%, and acceptedcmp5 recorded '3', at 7.3%. Accordingly, I looked at other acceptedcmp1, 2, and 4 but since the numbers were not the same, I judged each to be meaningful data.

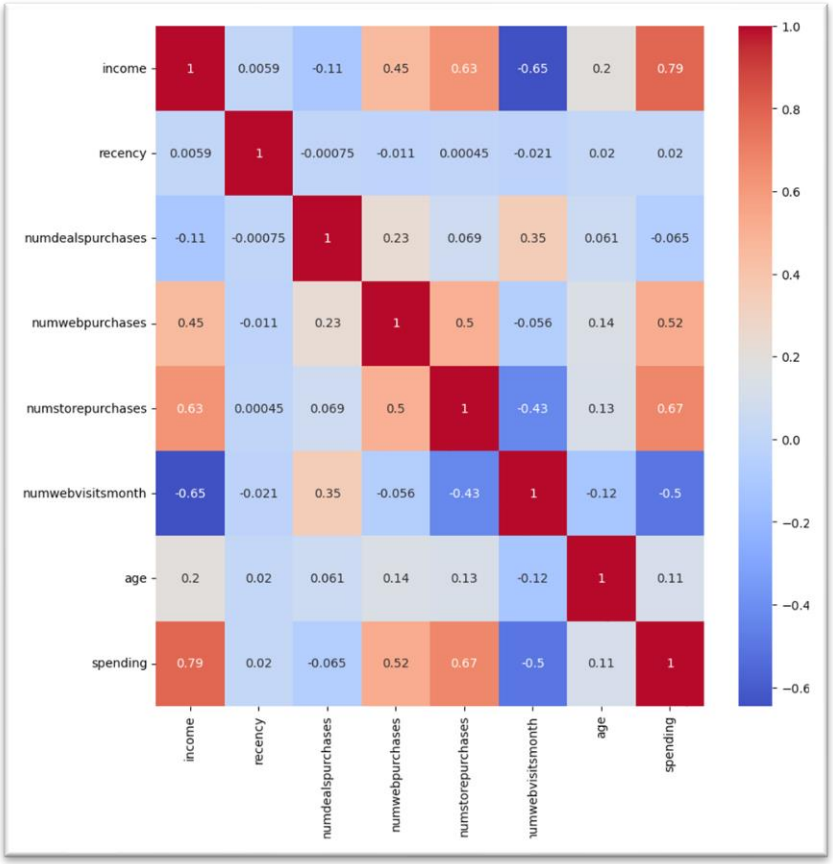
Logistic Regression Assumptions

Before performing logistic regression, we will look at the assumptions and check whether they are satisfied. I set the target variable to be a binary response for A1, below. I will mainly determine whether variables are independent, but this time I will check the correlation matrix first and mainly see the meaning of the column.

Assumptions of Logistic Regression (Lillian, 2018)

A1. Target variable is binary
A2. Predictive features are interval (continuous) or categorical
A3. Features are independent of one another
A4. Sample size is adequate – Rule of thumb: 50 records per predictor

Correlation Matrix



In the table, the correlation between spending and income is the highest at 0.79, and numstorepurchases and income are also high at 0.63. Correlation appears to be high mainly in columns related to spending, income, or purchases. numwebvisitmonth also shows a negative correlation value of -0.65 with income.

There is a high possibility that there is a clear correlation between the above variables. For example, if your income is high, your spending may be high. However, this may not necessarily be the case, and since the two do not necessarily affect each other, we decided to use all of them in the analysis without removing other num___purchases.

	chi-square p-value	correlated or not
acceptedcmp1 and acceptedcmp2	2.36e-15	'correlated'
acceptedcmp1 and has_child	5.06e-39	'correlated'
acceptedcmp1 and complain	0.44	'not-correlated'
acceptedcmp1 and marital_stat	0.76	'not-correlated'
marital_stat and has_child	0.009	'correlated'

This is part of analyzing correlation through chi-square test of binary variables. There are many cases where it is correlated, but in the case of acceptedcmp, it is a matter of whether marketing was accepted through each independent implementation, so even though correlation was found, we will include all of them in the analysis.

It turns out that only has_child is related to accpetedcmp, but since it is difficult to find a direct correlation, we will keep this. Of course, marital_stat and has_child are not completely independent of each other, but semantically, you can have children even if you are married, and you cannot have children even if you are married, so we will include them in the analysis.

Result of Logistic Regression

Result 1: before elimination

Logit Regression Results

Dep. Variable:	response	No. Observations:	1791
Model:	Logit	Df Residuals:	1771
Method:	MLE	Df Model:	19
Date:	Fri, 13 Oct 2023	Pseudo R-squ.:	0.3916
Time:	03:48:24	Log-Likelihood:	-439.36
converged:	True	LL-Null:	-722.12
Covariance Type:	nonrobust	LLR p-value:	9.308e-108

	coef	std err	z	P>|z|	[0.025	0.975]
const	-15.9508	2.046	-7.795	0.000	-19.961	-11.940
income	1.449e-06	8.26e-06	0.175	0.861	-1.47e-05	1.76e-05
recency	-0.0288	0.003	-8.507	0.000	-0.035	-0.022
numdealspurchases	0.0760	0.052	1.474	0.141	-0.025	0.177
numwebpurchases	0.0495	0.034	1.476	0.140	-0.016	0.115
numcatalogpurchases	0.0569	0.044	1.283	0.200	-0.030	0.144
numstorepurchases	-0.1907	0.039	-4.898	0.000	-0.267	-0.114
numwebvisitsmonth	0.0993	0.054	1.837	0.066	-0.007	0.205
acceptedcmp3	2.0983	0.263	7.986	0.000	1.583	2.613
acceptedcmp4	0.8212	0.314	2.614	0.009	0.205	1.437
acceptedcmp5	1.8280	0.321	5.693	0.000	1.199	2.457
acceptedcmp1	1.3825	0.328	4.210	0.000	0.739	2.026
acceptedcmp2	1.4782	0.620	2.384	0.017	0.263	2.693
complain	-0.4565	1.410	-0.324	0.746	-3.221	2.308
age	-0.0064	0.007	-0.867	0.386	-0.021	0.008
spending	0.0005	0.000	1.689	0.091	-7.9e-05	0.001
marital_stat	-1.3487	0.184	-7.335	0.000	-1.709	-0.988
has_child	-1.1296	0.245	-4.605	0.000	-1.610	-0.649
education_years	0.0843	0.020	4.198	0.000	0.045	0.124
enrollment_days	0.0039	0.001	7.406	0.000	0.003	0.005

I first constructed a logistic regression including all variables. Among these, income (p-value: 0.861), complain (0.746), and age (0.386) were excluded. In the excluded analysis, numcatalog purchases were also found to be higher than 0.25, so they were also removed, and a final logistic regression was performed.

Result 2: after elimination of 'response','income', 'age', 'complain', and 'numcatalogpurchases'

Logit Regression Results

Dep. Variable:	response	No. Observations:	1791
Model:	Logit	Df Residuals:	1775
Method:	MLE	Df Model:	15
Date:	Fri, 13 Oct 2023	Pseudo R-squ.:	0.3950
Time:	03:51:29	Log-Likelihood:	-455.27
converged:	True	LL-Null:	-752.46
Covariance Type:	nonrobust	LLR p-value:	5.545e-117

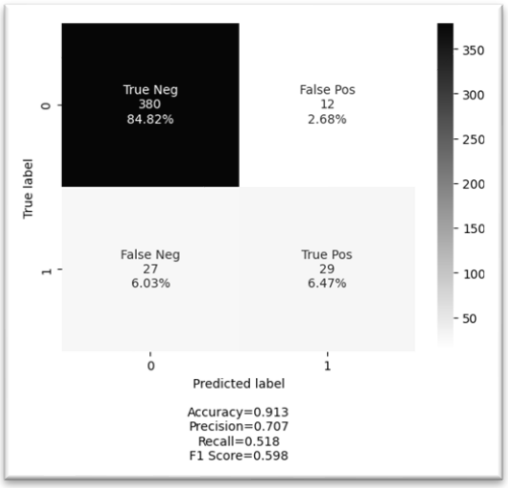
	coef	std err	z	P>|z|	[0.025	0.975]
const	-16.6974	1.916	-8.714	0.000	-20.453	-12.942
recency	-0.0296	0.003	-8.963	0.000	-0.036	-0.023
numdealspurchases	0.1188	0.049	2.428	0.015	0.023	0.215
numwebpurchases	0.0425	0.032	1.319	0.187	-0.021	0.106
numstorepurchases	-0.2152	0.038	-5.612	0.000	-0.290	-0.140
numwebvisitsmonth	0.0670	0.047	1.422	0.155	-0.025	0.159
acceptedcmp3	2.2276	0.259	8.608	0.000	1.720	2.735
acceptedcmp4	0.7325	0.303	2.415	0.016	0.138	1.327
acceptedcmp5	1.7684	0.308	5.748	0.000	1.165	2.371
acceptedcmp1	1.3651	0.318	4.292	0.000	0.742	1.988
acceptedcmp2	1.4947	0.606	2.465	0.014	0.306	2.683
spending	0.0008	0.000	3.448	0.001	0.000	0.001
marital_stat	-1.3159	0.180	-7.316	0.000	-1.668	-0.963
has_child	-1.1617	0.241	-4.830	0.000	-1.633	-0.690
education_years	0.0889	0.020	4.535	0.000	0.050	0.127
enrollment_days	0.0041	0.001	7.914	0.000	0.003	0.005

Attribute	p-value
recency	9.53e-20
accepetedcmp3	7.57e-19
enrollment_days	9.08e-16
marital_stat	4.29e-13
accetedcmp5	4.12e-09

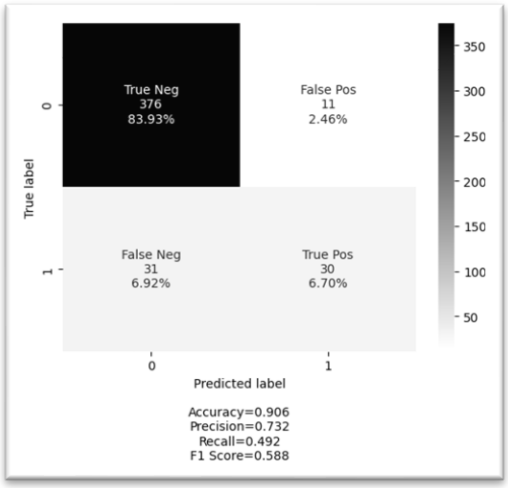
The recency, which represents 'Number of days since the last purchase', had a negative coefficient, indicating significance for the largest response. The longer the ‘Number of days since the last purchase’, the lower the probability of response.

Marital_status also had a negative effect and was recorded as the fourth most significant attribute. acceptedcmp3, enrollment_days, and acceptedcmp5 were found to sequentially exert positive significance on the response.

Before Feature selection



After Feature Selection

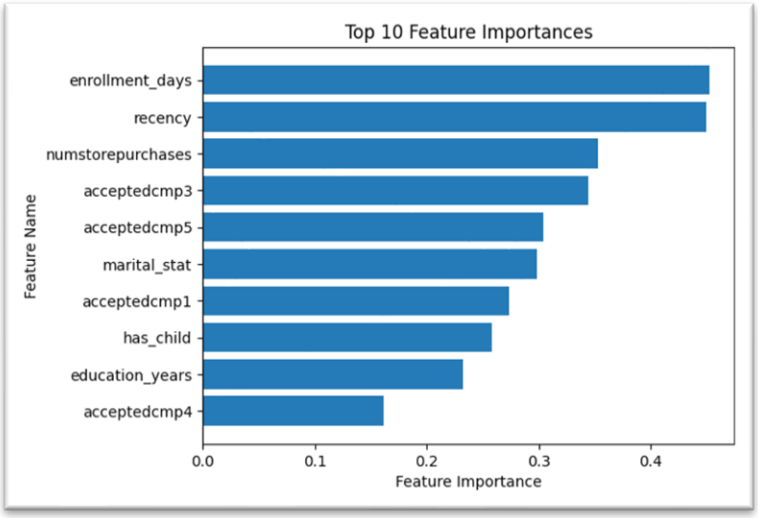


On the left, the results are shown when all variables are used. The accuracy is 0.913 on the left, which is 0.007 higher. However, after removing variables with high p-value, the results on the right show that precision is higher than before selecting the variables. This means that noise was reduced in the process of removing variables with low involvement.

Ultimately, accuracy was 0.906, precision was 0.732, recall was 0.492, and F1-score was 0.588. In this data, you can also see that the recall value was low because the '0' data accounted for a significantly higher value than the '1' data.

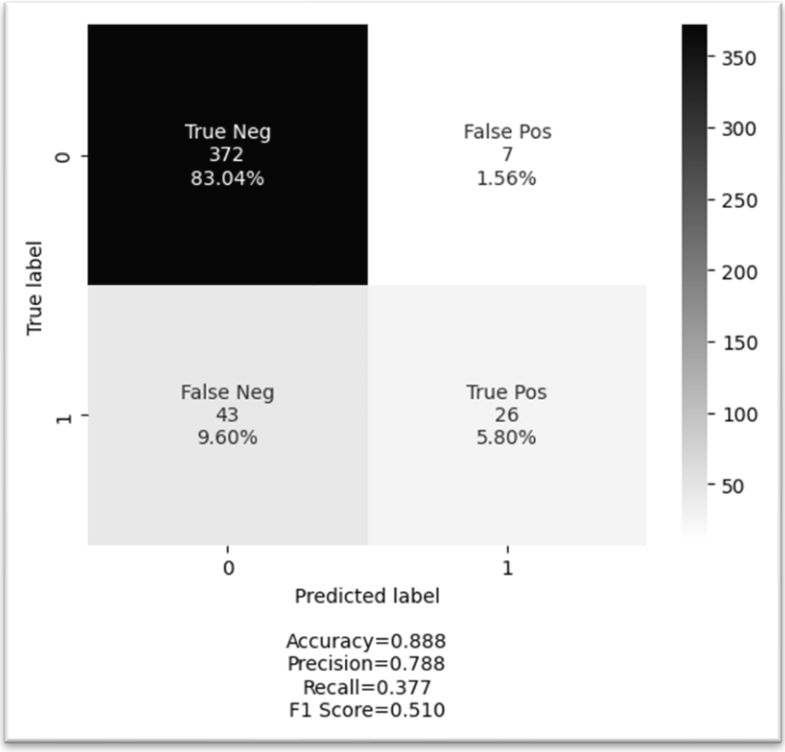
Result of SVM

Result: after elimination of 'response','income', 'age', 'complain', and 'numcatalogpurchases'



The SVM model's analysis of the top 10 feature imports revealed the following significant influences in a sequential order: enrollment_days, recency, numstorepurchases, and acceptedcmp3 and 5 consecutively. The SVM model identified key factors influencing customer response:

Enrollment Days: The duration since customer enrollment had the most significant impact, emphasizing the importance of customer loyalty.
Recency: The number of days since the last purchase was the second most influential factor, indicating the customer's recent engagement.
NumStorePurchases: Direct in-store purchases strongly affected responses, highlighting the impact of physical store visits.
Marital Status and Accepted Campaigns: These attributes, previously significant in logistic regression, remained influential, underlining their ongoing importance in shaping customer behavior.



Looking at the confusion matrix of the SVM model, the accuracy was 0.888, which was lower than logistic regression, the precision was 0.788, which was 0.56 higher, and the recall was 0.377, which was more than 0.115 lower. Ultimately, the F1-score recorded 0.510, which is 0.078 lower than logistic regression. Accuracy is not a good metric to use when you have class imbalance. One way to solve class imbalance problems is to work on your sample. Another way to solve class imbalance problems is to use better accuracy metrics like the F1 score (Joos, 2001).

Conclusion

I conducted 'response' analysis, as target variable, using marketing and consumption patterns and customer information. Through rigorous data preprocessing and cleaning, I found out the importance of accurately identifying variables and tried to increase model accuracy. Logistic regression and SVM were performed together to compare significant variables, and finally the results were compared in a confusion matrix. I thought about the values of Accuracy, Precision, and Recall and learned that the judging parameters should be different depending on the given problem and context.

Reference

Joos, Korstanje. (2021, August 31). The F1 score. Medium. Retrieved from https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

sefidian. (n.d.). Measure the correlation between numerical and categorical variables and the correlation between two categorical variables in Python: Chi-Square and ANOVA. Retrieved from http://www.sefidian.com/2020/08/02/measure-the-correlation-between-numerical-and-categoricalvariables-and-the-correlation-between-two-categorical-variables-in-python-chi-square-andanova/#:~:text=The%20ANOVA%20test%20is%20used,variables%20for%20each%20categorical%20value.

ML Explained. (2023). Calculate correlation among categorical variables in Python. YouTube. Retrieved from https://www.youtube.com/watch?v=fzzUfa0-VsE

Lillian Pierson, P.E. (2018). Logistic regression example in Python. DATA MANIA. Retrieved from https://www.data-mania.com/blog/logistic-regression-example-in-python/

utdallas. (n.d.). Building and applying logistic regression models. Retrieved from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://personal.utdallas.edu/~pkc022000/6390/SP06/NOTES/Logistic_Regression_4.pdf

Kenneth, Leung. (2021, October 4). Assumptions of logistic regression, clearly explained. Medium. Retrieved from https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290#:~:text=Logistic%20regression%20does%20not%20require,but%20not%20for%20logistic%20regression.

Data-Centric-AI-Community. (n.d.). ydata_profiling_demo. github. Retrieved from githttps://colab.research.google.com/github/Data-Centric-AI-Community/awesome-data-centric-ai/blob/master/medium/data-profiling-tools/notebooks/ydata_profiling_demo.ipynb#scrollTo=Sr4EoQ4UuVZW

RODOLFO, SALDANHA. (2020). Marketing campaign. Kaggle. Retrieved from https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign

RAPHAEL. (2020). Data prep, visual EDA and statistical hypothesis. Kaggle. Retrieved from https://www.kaggle.com/code/raphael2711/data-prep-visual-eda-and-statistical-hypothesis

Smitha, Dinesh Semwal. (2023, July 19). Python program to find number of days between two given dates. geeksforgeeks. Retrieved from https://www.geeksforgeeks.org/python-program-to-find-number-of-days-between-two-given-dates/

Demetri, Pananos. (2019, December 10). Statsmodels logistic regression: adding intercept?. stackexchange. Retrieved from https://stats.stackexchange.com/questions/440242/statsmodels-logistic-regression-adding-intercept#:~:text=It%20is%20almost%20always%20necessary,interpretation%20of%20the%20other%20coefficients.

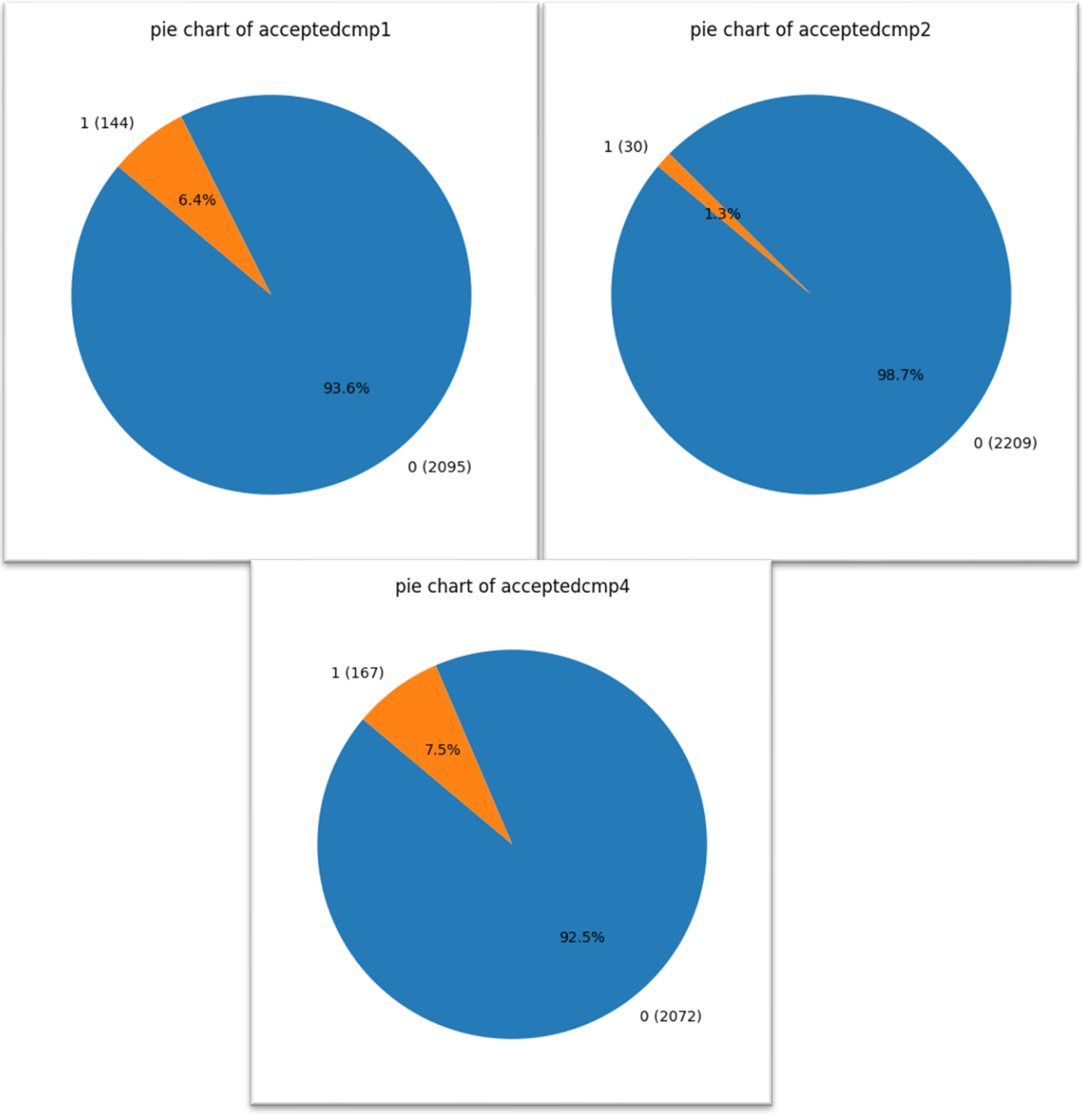
Chaithanya, Kumar. (2017). What is the correlation coefficient threshold above which a predictor should be considered for a predictive model?. Quora. Retrieved from

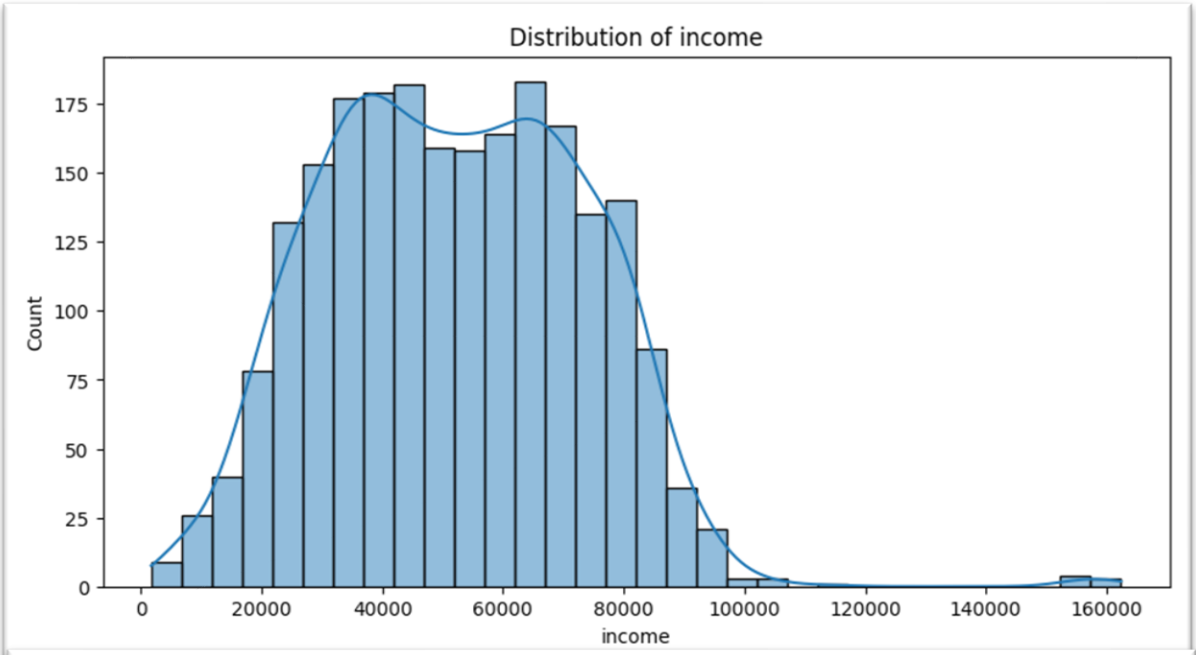
https://www.quora.com/What-is-the-correlation-coefficient-threshold-above-which-a-predictor-should-be-considered-for-a-predictive-model

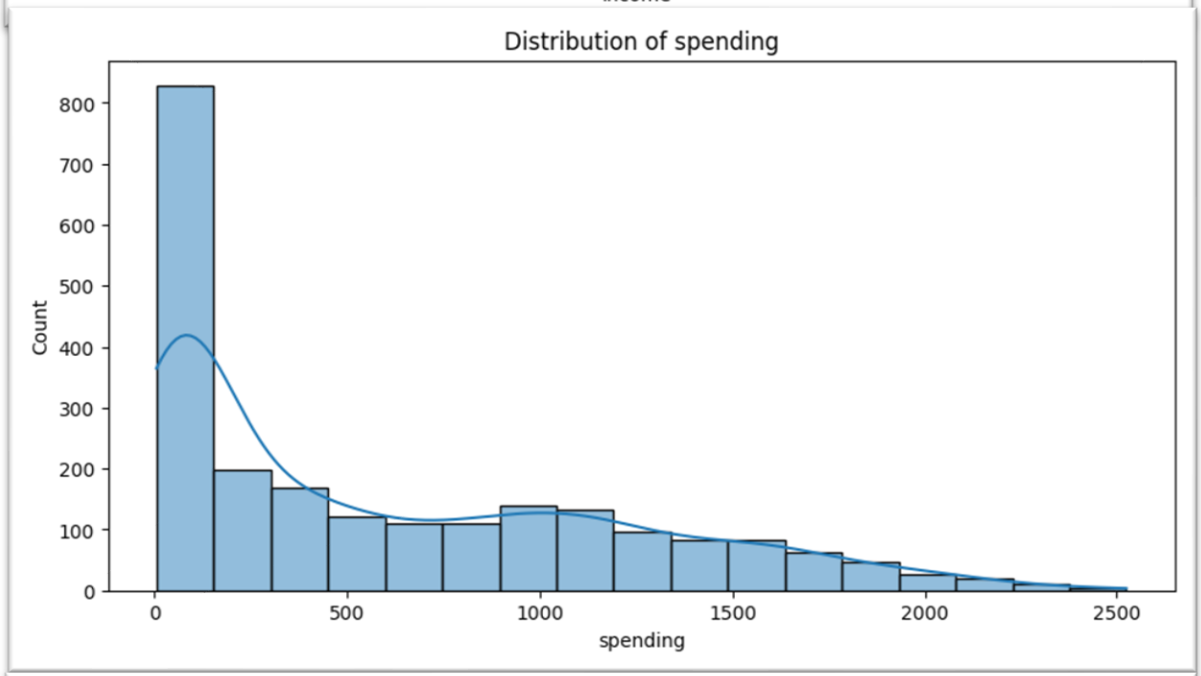
mltut. (n.d.). SVM implementation in python from scratch- step by step guide. Retrieved from https://www.mltut.com/svm-implementation-in-python-from-scratch/

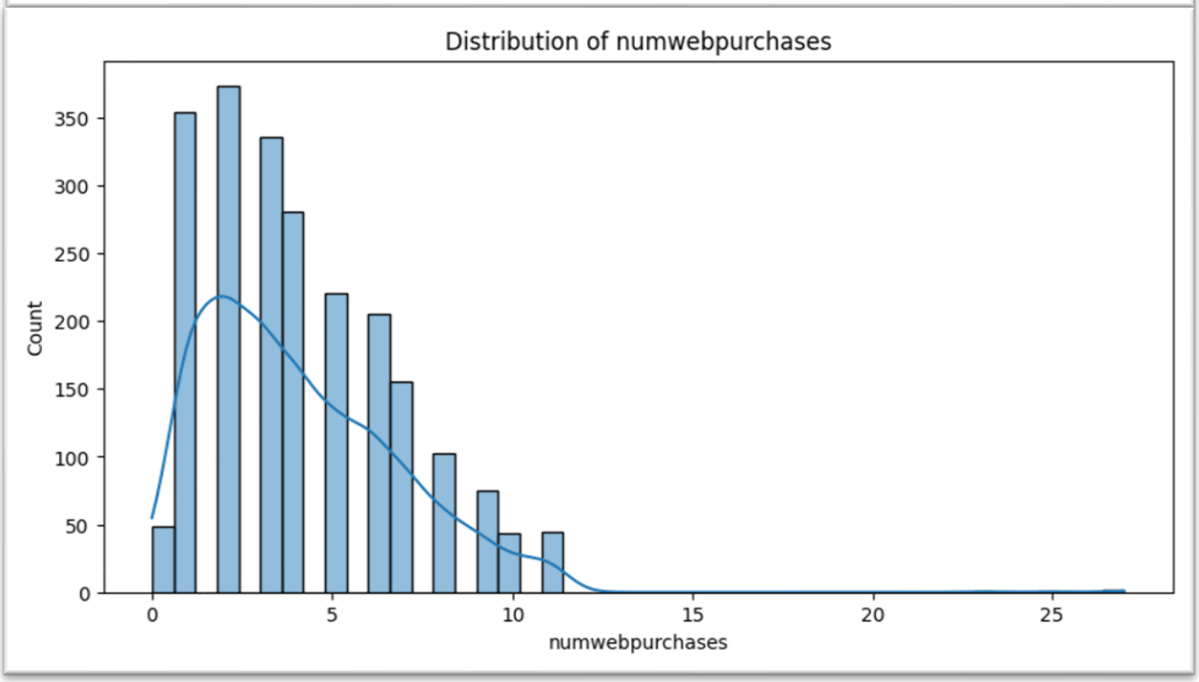
saturncloud. (n.d.). How to increase model accuracy of logistic regression. Retrieved from https://saturncloud.io/#:~:text=Data%20preprocessing%20is%20a%20crucial,values%2C%20and%20normalizing%20the%20data.

Appendix (Graphs):




Distribution of income
Count
income
0
20000
40000
60000
80000
100000
120000
140000
160000
0
25
50
75
100
125
150
175


Distribution of spending
Count
spending
0
500
1000
1500
2000
2500
0
100
200
300
400
500
600
700
800


Distribution of numwebpurchases
Count
numwebpurchases
0
5
10
15
20
25
0
50
100
150
200
250
300
350

Appendix (Python code):

```
import pandas as pd
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from collections import Counter

"""### DATA Import"""

df = pd.read_excel('marketing_campaign.xlsx')

df.head()

df.tail()

"""## Visualization & Understanding Dataset"""

def missing_values(df):
    missing_number = df.isnull().sum().sort_values(ascending=False)
    missing_percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)
    missing_values = pd.concat([missing_number, missing_percent], axis=1, keys=['Missing_Number',
'Missing_Percent'])
    return missing_values[missing_values['Missing_Number']>0]

def first_looking(df):
    print(colored("Shape:", attrs=['bold']), df.shape,'\n',
          colored('-'*79, 'red', attrs=['bold']),
          colored("\nInfo:\n", attrs=['bold']), sep='')
    print(df.info(), '\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("Number of Uniques:\n", attrs=['bold']), df.nunique(),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("Missing Values:\n", attrs=['bold']), missing_values(df),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("All Columns:", attrs=['bold']), list(df.columns),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')

    df.columns= df.columns.str.lower().str.replace('&', '_').str.replace(' ', '_')

    print(colored("Columns after rename:", attrs=['bold']), list(df.columns),'\n',
              colored('-'*79, 'red', attrs=['bold']), sep='')

pip install colorama

import colorama
from colorama import Fore, Style  # maakes strings colored
from termcolor import colored

missing_values(df)

first_looking(df)

df.describe()

!pip install -U ydata-profiling[notebook]==4.0.0 matplotlib==3.5.1

from ydata_profiling import ProfileReport

df.profile_report()

"""## Data Preperation

"""

from datetime import datetime
```

```
df1=df.copy()

df['age']=2023-df['year_birth']
df['spending']=df['mntwines']+df['mntfruits']+df['mntmeatproducts']+df['mntfishproducts']+df['mntsweetproducts']
+df['mntgoldprods']
df['marital_stat']=df['marital_status'].replace({'Divorced':'Alone','Single':'Alone','Married':'In
couple','Together':'In couple','Absurd':'Alone','Widow':'Alone','YOLO':'Alone'})
df['has_child'] = np.where(df.kidhome+df.teenhome > 0, 1, 0)
df['education_years']=df['education'].replace({'Basic':5,'2n Cycle':8,'Graduation':12,'Master':18,'PhD':21})
print(df[['age','income','spending','marital_stat','has_child','education_years']])

df['dt_customer'] = df['dt_customer'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))

df['enrollment_days'] = datetime.now() - df['dt_customer']

df=df.drop(['year_birth','mntwines', 'mntfruits', 'mntmeatproducts',
'mntfishproducts','mntsweetproducts','mntgoldprods', 'marital_status', 'kidhome', 'teenhome', 'education',
'dt_customer', 'id'], axis=1)

df['marital_stat']=df['marital_stat'].replace({'Alone':0,'In couple':1})

df

df2=df.copy()

import plotly.express as px

df = df[['age','income','spending','education_years']]

fig = px.box(df.melt(), y="value", facet_col="variable",facet_col_wrap=2, boxmode="overlay",
color="variable",height=1000, width=900)
fig.update_yaxes(matches=None)

for i in range(len(fig["data"])):
    yaxis_name = 'yaxis' if i == 0 else f'yaxis{i + 1}'
    fig.layout[yaxis_name].showticklabels = True

fig.update_layout(showlegend=False)
fig.update_xaxes(showline=True, linewidth=2, linecolor='grey')
fig.update_yaxes(showline=True, linewidth=2, linecolor='grey')

fig.show()

df = df2.drop(df2[df2['income']> 600000].index).reset_index(drop=True)
df

df.describe()

df.describe()

df3=df.copy()

"""#### Handling Missing values with KNN"""

from sklearn.impute import KNNImputer

count_nan = len(df3) - df3.count()
print(count_nan)

imputer = KNNImputer()
imputer = KNNImputer(n_neighbors=5,metric='nan_euclidean')
# fit on the dataset
imputer.fit(df[['income','age','education_years']])
# transform the dataset
X = imputer.transform(df[['income','age','education_years']])
Income_impute=pd.DataFrame(X,columns=['income','age','education_years'])
df['income']=Income_impute['income'].reset_index(drop=True)
count_nan = len(df) - df.count()
print(count_nan)
```

```
df.nunique()

df.info()

df['enrollment_days'] = df['enrollment_days'].dt.days.astype(int)

"""## Data Visualization"""

import seaborn as sns

df1 = df.copy()

plt.figure(figsize=(10, 5))
sns.histplot(x=df1['recency'], kde=True)

plt.title("Distribution of recency")

counts = df1['acceptedcmp3'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of acceptedcmp3')
plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(x=df1['enrollment_days'], kde=True)

plt.title("Distribution of enrollment_days")

counts = df1['marital_stat'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of marital_stat')
plt.show()

counts = df1['acceptedcmp5'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of acceptedcmp5')
plt.show()

counts = df1['acceptedcmp1'].value_counts()

# Extract labels and sizes for the pie chart
```

```
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of acceptedcmp1')
plt.show()

counts = df1['acceptedcmp2'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of acceptedcmp2')
plt.show()

counts = df1['acceptedcmp4'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of acceptedcmp4')
plt.show()

counts = df1['response'].value_counts()

# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values

# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)

# Display the pie chart
plt.title('pie chart of response')
plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(x=df1['income'], kde=True)

plt.title("Distribution of income")

plt.figure(figsize=(10, 5))
sns.histplot(x=df1['spending'], kde=True)

plt.title("Distribution of spending")

plt.figure(figsize=(10, 5))
sns.histplot(x=df1['numwebpurchases'], kde=True)
```

```
plt.title("Distribution of numwebpurchases")

"""#### Correlation matrix"""

df4=df.copy()

df=df.drop(['z_costcontact', 'z_revenue'], axis=1)

df.info()

df.nunique()

df_num=df[['income', 'recency', 'numdealspurchases', 'numwebpurchases', 'numstorepurchases',
'numwebvisitsmonth', 'age', 'spending', 'enrollment_days']]

car_corr_matrix=df_num.corr()

plt.figure(figsize=(10, 10))
sns.heatmap(car_corr_matrix, cmap='coolwarm', annot=True)

"""#### Correlation between binary variables"""

from scipy.stats import chi2_contingency

cross_tab = pd.crosstab(index=df['acceptedcmp1'], columns=df['acceptedcmp2'])
cross_tab

chi_sq_result = chi2_contingency(cross_tab,)

p, x = chi_sq_result[1], "reject" if chi_sq_result[1] < 0.05 else "accept"

print(f"The p-value is {chi_sq_result[1]} and hence we {x} the null hypothesis with {chi_sq_result[2]} degrees
of freedom")

bi_var=df[['acceptedcmp1', 'acceptedcmp2', 'acceptedcmp3', 'acceptedcmp4', 'acceptedcmp5', 'complain',
'marital_stat', 'has_child']]

def is_correlated(x, y):
    ct = pd.crosstab(index=x, columns=y)
    chi_sq_result = chi2_contingency(ct)
    p, correlation_status = chi_sq_result[1], "correlated" if chi_sq_result[1] < 0.05 else "not-correlated"
    return p, correlation_status

for i in range(len(bi_var.columns)):
    for j in range(i + 1, len(bi_var.columns)):
        x1 = bi_var.columns[i]
        x2 = bi_var.columns[j]
        correlation_result = is_correlated(bi_var[x1], bi_var[x2])
        print(f"Correlation between {x1} and {x2}: {correlation_result}")

"""## Analysis

### Saving before changing df as df1
"""

df5 = df.copy()

"""### Logistics regression model fitting"""

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import permutation_importance
from sklearn import metrics
from scipy.stats import norm
import statsmodels.api as sm

df

y=df[['response']]
x=df.drop(['response'], axis=1)
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

Y=y_train
X=x_train
x1=sm.add_constant(X)
model = sm.Logit(endog=Y, exog=x1).fit()

print(model.summary())

# Create a logistic regression model
model = sm.Logit(endog=Y, exog=x1).fit()
p_values = model.pvalues
print(p_values)

ranked_features = p_values.sort_values(ascending=True)
print(ranked_features)

from sklearn.metrics._plot.confusion_matrix import confusion_matrix

x_test = sm.add_constant(x_test)

pred = model.predict(x_test)

binary_predictions = round(pred)

confusion = confusion_matrix(y_test, binary_predictions)

def make_confusion_matrix(cf,
                          group_names=None,
                          categories='auto',
                          count=True,
                          percent=True,
                          cbar=True,
                          xyticks=True,
                          xyplotlabels=True,
                          sum_stats=True,
                          figsize=None,
                          cmap='Blues',
                          title=None):
    '''
    This function will make a pretty plot of an sklearn Confusion Matrix cm using a Seaborn heatmap
visualization.

    Arguments
    ---------
    cf:            confusion matrix to be passed in

    group_names:   List of strings that represent the labels row by row to be shown in each square.

    categories:    List of strings containing the categories to be displayed on the x,y axis. Default is 'auto'

    count:         If True, show the raw number in the confusion matrix. Default is True.

    normalize:     If True, show the proportions for each category. Default is True.

    cbar:          If True, show the color bar. The cbar values are based off the values in the confusion
matrix.
                   Default is True.

    xyticks:       If True, show x and y ticks. Default is True.

    xyplotlabels:  If True, show 'True Label' and 'Predicted Label' on the figure. Default is True.

    sum_stats:     If True, display summary statistics below the figure. Default is True.

    figsize:       Tuple representing the figure size. Default will be the matplotlib rcParams value.

    cmap:          Colormap of the values displayed from matplotlib.pyplot.cm. Default is 'Blues'
                   See http://matplotlib.org/examples/color/colormaps_reference.html
```

```
    title:         Title for the heatmap. Default is None.

    '''


    # CODE TO GENERATE TEXT INSIDE EACH SQUARE
    blanks = ['' for i in range(cf.size)]

    if group_names and len(group_names)==cf.size:
        group_labels = ["{}\n".format(value) for value in group_names]
    else:
        group_labels = blanks

    if count:
        group_counts = ["{0:0.0f}\n".format(value) for value in cf.flatten()]
    else:
        group_counts = blanks

    if percent:
        group_percentages = ["{0:.2%}".format(value) for value in cf.flatten()/np.sum(cf)]
    else:
        group_percentages = blanks

    box_labels = [f"{v1}{v2}{v3}".strip() for v1, v2, v3 in zip(group_labels,group_counts,group_percentages)]
    box_labels = np.asarray(box_labels).reshape(cf.shape[0],cf.shape[1])


    # CODE TO GENERATE SUMMARY STATISTICS & TEXT FOR SUMMARY STATS
    if sum_stats:
        #Accuracy is sum of diagonal divided by total observations
        accuracy  = np.trace(cf) / float(np.sum(cf))

        #if it is a binary confusion matrix, show some more stats
        if len(cf)==2:
            #Metrics for Binary Confusion Matrices
            precision = cf[1,1] / sum(cf[:,1])
            recall    = cf[1,1] / sum(cf[1,:])
            f1_score  = 2*precision*recall / (precision + recall)
            stats_text = "\n\nAccuracy={:0.3f}\nPrecision={:0.3f}\nRecall={:0.3f}\nF1 Score={:0.3f}".format(
                accuracy,precision,recall,f1_score)
        else:
            stats_text = "\n\nAccuracy={:0.3f}".format(accuracy)
    else:
        stats_text = ""


    # SET FIGURE PARAMETERS ACCORDING TO OTHER ARGUMENTS
    if figsize==None:
        #Get default figure size if not set
        figsize = plt.rcParams.get('figure.figsize')

    if xyticks==False:
        #Do not show categories if xyticks is False
        categories=False


    # MAKE THE HEATMAP VISUALIZATION
    plt.figure(figsize=figsize)
    sns.heatmap(cf,annot=box_labels,fmt="",cmap=cmap,cbar=cbar,xticklabels=categories,yticklabels=categories)

    if xyplotlabels:
        plt.ylabel('True label')
        plt.xlabel('Predicted label' + stats_text)
    else:
        plt.xlabel(stats_text)

    if title:
        plt.title(title)

labels = ['True Neg','False Pos','False Neg','True Pos']
```

```
categories = ['0', '1']
make_confusion_matrix(confusion,
                      group_names=labels,
                      categories=categories,
                      cmap='binary')

"""#### Select with p-value"""

y=df[['response']]
x=df.drop(['response','income', 'age', 'complain', 'numcatalogpurchases'], axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

Y=y_train
X=x_train
x1=sm.add_constant(X)
model = sm.Logit(endog=Y, exog=x1).fit()

print(model.summary())

# Create a logistic regression model
model = sm.Logit(endog=Y, exog=x1).fit()
p_values = model.pvalues
print(p_values)

ranked_features = p_values.sort_values(ascending=True)
print(ranked_features)

x_test = sm.add_constant(x_test)

pred = model.predict(x_test)

binary_predictions = round(pred)

confusion = confusion_matrix(y_test, binary_predictions)

labels = ['True Neg','False Pos','False Neg','True Pos']
categories = ['0', '1']
make_confusion_matrix(confusion,
                      group_names=labels,
                      categories=categories,
                      cmap='binary')

"""#### ROC curve"""

pip install stat

from sklearn.metrics import roc_curve, auc, roc_auc_score

x_test_with_const = sm.add_constant(x_test)
pred2 = model.predict(x_test_with_const)
binary_predictions = round(pred2)

fpr, tpr, thresholds = roc_curve(y_true=y_test, y_score=pred2)
auc = roc_auc_score(y_true=y_test, y_score=pred2)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

"""## SVM"""

from sklearn import svm
from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)

classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

labels = ['True Neg','False Pos','False Neg','True Pos']
categories = ['0', '1']
make_confusion_matrix(cm,
                      group_names=labels,
                      categories=categories,
                      cmap='binary')

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)

classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

"""#### determining most important variables"""

def f_importances(coef, names, top=-1):
    imp = coef
    sorted_idx = np.argsort(imp)
    if top == -1:
        top = len(names)

    plt.barh(range(top), imp[sorted_idx][-top:], align='center')
    plt.yticks(range(top), np.array(names)[sorted_idx][-top:])
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature Name')
    plt.title('Top {} Feature Importances'.format(top))
    plt.show()

features_names = ['recency', 'numdealspurchases', 'numwebpurchases',
       'numstorepurchases', 'numwebvisitsmonth',
       'acceptedcmp3', 'acceptedcmp4', 'acceptedcmp5', 'acceptedcmp1',
       'acceptedcmp2', 'spending',
       'marital_stat', 'has_child', 'education_years', 'enrollment_days']
svm = svm.SVC(kernel='linear')
svm.fit(X_train, y_train)

f_importances(abs(classifier.coef_[0]), feature_names, top=10)

from matplotlib.colors import ListedColormap
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Assuming X_train has 15 features
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)

# Initialize and train your SVM classifier using X_train_pca and y_train
classifier = SVC(kernel='linear')
classifier.fit(X_train_pca, y_train)

# Transform X_test using the same PCA object used during training
X_test_pca = pca.transform(X_test)
```

```
# Predict using the reduced 2D features
predictions = classifier.predict(X_test_pca)

X_set, y_set = X_test, y_test

y_set = y_set.values.flatten()

y_set = y_set.flatten()

X_set_pca = pca.transform(X_set)

# Create meshgrid using PCA-transformed features
X1, X2 = np.meshgrid(np.arange(start=X_set_pca[:, 0].min() - 1, stop=X_set_pca[:, 0].max() + 1, step=0.01),
                     np.arange(start=X_set_pca[:, 1].min() - 1, stop=X_set_pca[:, 1].max() + 1, step=0.01))

# Predict using the reduced 2D features
predictions = classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape)

# Plot the contour plot with predictions
plt.contourf(X1, X2, predictions, alpha=0.75, cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

# Scatter plot the PCA-transformed test data points
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set_pca[y_set == j, 0], X_set_pca[y_set == j, 1],
                c=ListedColormap(('red', 'green'))(i), label=j)

plt.title('SVM (Test set)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()

print("X_set_pca shape:", X_set_pca.shape)
print("y_set shape:", y_set.shape)
```