STAT 4911 Project 1
Donor Giving Insights Report
Predicting a 3K - 99K Jump

Instructor: Thomas Metzger
Team Number: Team 4
Team Members: Xidan Kou, Songyuan Wu, Caroline Pier, Kat Husar, Troy Stein, Carli Werner

I. Introduction

Through exploratory data analysis of what differentiates a donor who makes the jump versus a donor who has not made the jump we found that particular demographic information, donation histories, donation methods, and alumni relations were influential predictors.

A random forest and a boosting tree was used for exploring the important variables for donation methods. A random forest model was also used to explore predictive accuracy among predictive variables of demographic information, donation history, alumni relations, and donation methods. Logistic regression and Naive Bayes models were used to predict a donation jump to the $3,000 to $99,000 range using demographic information, donation history, alumni relations, and donation methods. K-means clustering was used to help distinguish constituents who made the jump and those who didn't by arbitrarily placing centroids in the data and then iterating from that point to the final solution. The clustering labels generated from the k-means model would then be treated as a new feature to feed into the random forest model to strengthen the performance of the classification model. A Linear SVM was used to help predict whether someone would make the jump or not using the last 12 fiscal years.

II. Exploratory Data Analysis

A. Data Preparation

The constituents dataset and the donor data set were joined under full join conditions, taking the variables from both sides of the join. The variable for the date compare type for donors of $3,000-$99,000 jump status was changed to 0 for no jump and 1 for jump. Here, the donors who made a donation jump to 3K-99K are defined in the "jump" data set. Those who have not are defined in the "no_jump" data set. A data saving process was used to transfer the data for the tableau visualizations.

```
df = full_join(ppl, d3k99k, by=c("ID_CONSTITUENTLOOKUPID"="CONSTITUENTLOOKUPID"))
#rename param_ID_DATECOMPARETYPE to 0 and 1
df["param_ID_DATECOMPARETYPE"][df["param_ID_DATECOMPARETYPE"] == 2] <- 0
#creating a jump data set and no jump data set from the conjoined table
jump = df %>% filter(param_ID_DATECOMPARETYPE == 1)
no_jump = df %>% filter(param_ID_DATECOMPARETYPE == 0)
#saving datasets for tableau
save(jump, file = "donorjumps.RData")
no_jump_tab = sample_n(no_jump, 20000)
save(no_jump_tab, file = "nojumpdonors_xs.RData")
```

B. Composition of a Jump

To make the composition of a jump radar chart the calculations below were made. The variables considered include: OSU alumni status, first donation amount, income, Columbus area high donation zip-code, event donation, recent events, and the age of the largest donation.

I like this plot a lot

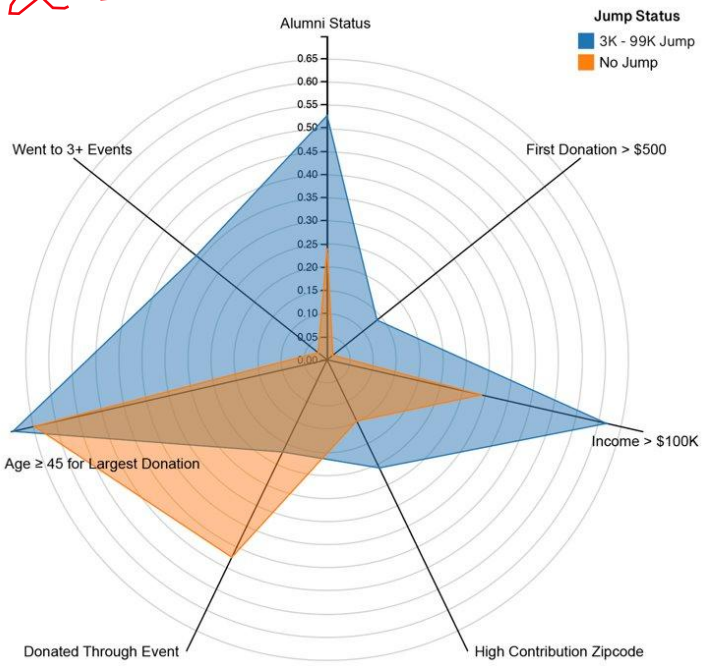


Figure 2.1 The Composition of a Jump

Some notable differences include that the $3,000 - $99,000 donors went to more events, more of them were alumni, their first donations tend to be more than $500, their projected incomes tend to be more than $100,000, and more tend to live in high contribution zip-codes. High contribution zip-codes were defined based on a mapping of donors by zip-code which displayed a high concentration of 3K - 99K donors in the Columbus area. Many donors tend to donate through events, although more non-jump donors tend to donate through this method. The ages of the largest donation are similar for both groups.

```
#radar chart visualization creation calculations
#OSU alumni status
jump_alum = jump %>% filter(!is.na(IS_OSUALUMNI))
no_jump_alum = no_jump %>% filter(!is.na(IS_OSUALUMNI))
table(jump_alum$IS_OSUALUMNI)
table(no_jump_alum$IS_OSUALUMNI)

#first donation > 500
jump_first = jump %>% filter(!is.na(AMT_DONOR_GIFT_FIRST))
no_jump_first = no_jump %>% filter(!is.na(AMT_DONOR_GIFT_FIRST))
table(no_jump_first$AMT_DONOR_GIFT_FIRST > 500)
table(jump_first$AMT_DONOR_GIFT_FIRST > 500)

#income greater than 100,000
jump_income = jump %>% filter(!is.na(AMT_WEALTH_DEMO_INCOME_RAW_TA))
no_jump_income = no_jump %>% filter(!is.na(AMT_WEALTH_DEMO_INCOME_RAW_TA))
table(no_jump_income$AMT_WEALTH_DEMO_INCOME_RAW_TA > 100000)
table(jump_income$AMT_WEALTH_DEMO_INCOME_RAW_TA >100000)

#columbus area
jump_zip = jump %>% filter(!is.na(CAT_ADDRESS_ZIP5))
no_jump_zip = no_jump %>% filter(!is.na(CAT_ADDRESS_ZIP5))
zipcodes = c('43085','43125', '43015', '43230', '43081', '43082',
'43017','43065','43016','43220','43235','43026',
```

```
            '43123', '43214', '43221')
jump_zip$new_bool_zip <- (jump_zip$CAT_ADDRESS_ZIP5 %in% zipcodes)
table(jump_zip$new_bool_zip)
no_jump_zip$new_bool_zip <- (no_jump_zip$CAT_ADDRESS_ZIP5 %in% zipcodes)
table(no_jump_zip$new_bool_zip)

#donated through event
jump_ea = jump %>% filter(!is.na(IS_DONOR_APPEAL_EVENT))
no_jump_ea = no_jump %>% filter(!is.na(IS_DONOR_APPEAL_EVENT))
table(jump_ea$IS_DONOR_APPEAL_EVENT)
table(no_jump_ea$IS_DONOR_APPEAL_EVENT)

#went to 3 or more events
jump_events = jump %>% filter(!is.na(N_EVENTS))
no_jump_events = no_jump %>% filter(!is.na(N_EVENTS))
```

C. Age of Jump Comparisons

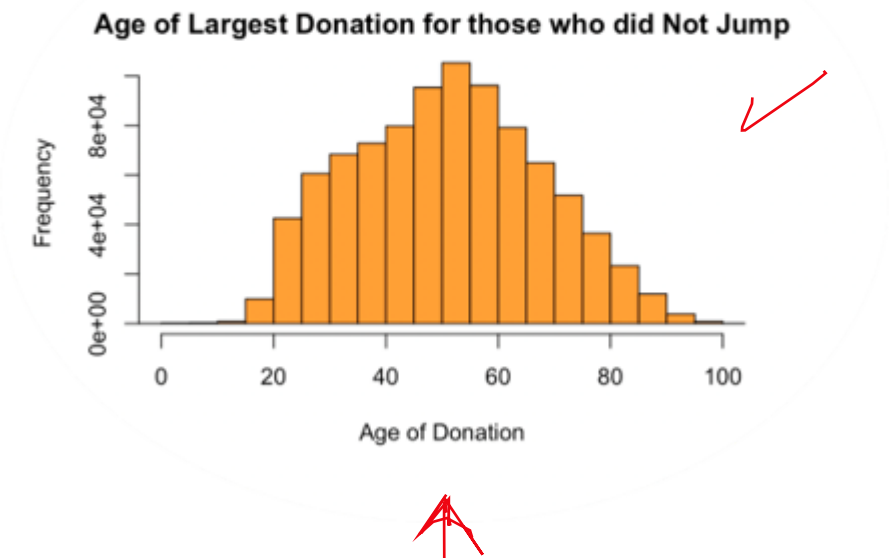


Figure 2.2 Age of Largest Donation for those who did Not Jump

The age of the largest donation for donors who did not make the jump has its distribution slightly more weighted toward the 20 to 40 age range.

These would be clearer as side-by-side boxplots

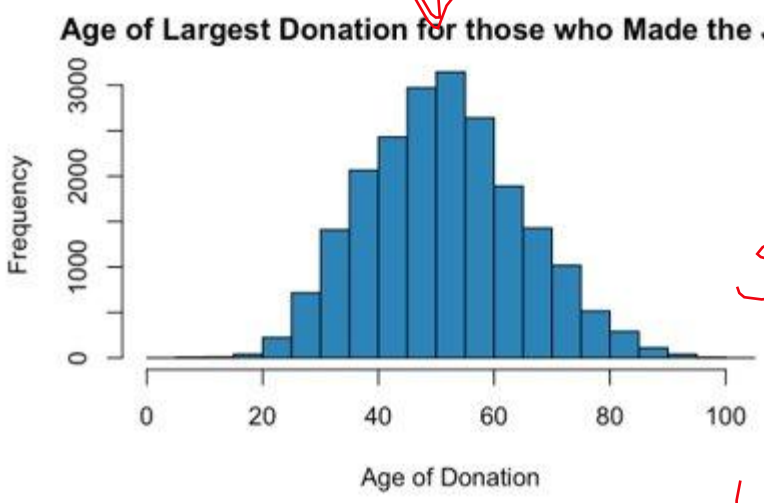


Figure 2.3 Age of Largest Donation for those who Made the Jump

The age of the largest donation for those who did make the jump is centered around 50 and follows a relatively normal distribution.

```
#creating histograms for age of donations
hist(jump_yob$FY_DONOR_FY_LARGEST_LAST - jump_yob$VAL_CONST_YOB_CLEAN, main = "Age of Largest
Donation for those who Made the Jump", xlab = "Age of Donation", col = "steelblue")
hist(no_jump_yob$FY_DONOR_FY_LARGEST_LAST - no_jump_yob$VAL_CONST_YOB_CLEAN, main = "Age of
Largest Donation for those who did Not Jump", xlab = "Age of Donation", xlim = c(0,100), col =
"tan1", breaks = 30)
```

D. First Donation Amount

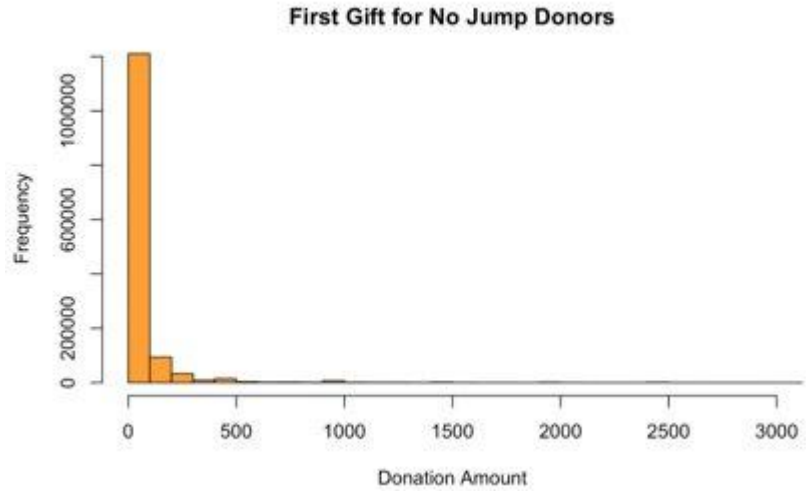


Figure 2.4 First Gift for No Jump Donors

The distribution of first gift donations, $1 to $3,000, for no-jump donors was concentrated toward $1 to $100 dollars.

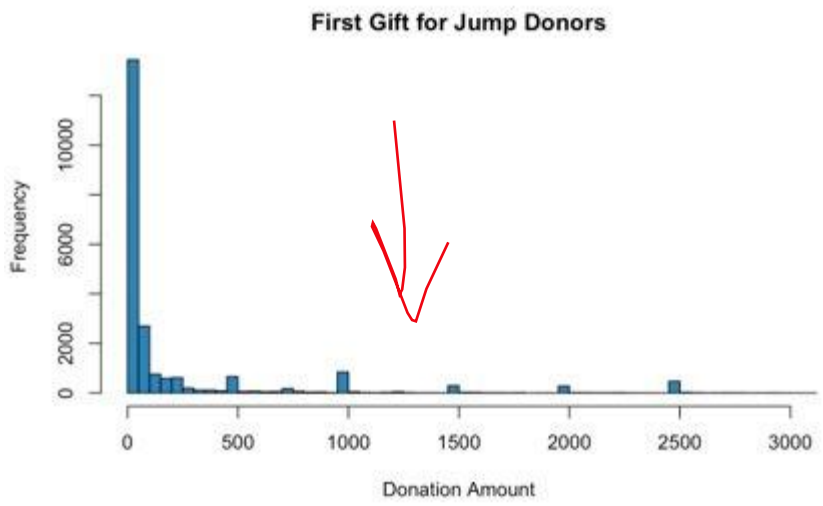


Figure 2.5 First Gift for Jump Donors

The distribution of first gift donations $1 to $3,000 for jump donors had notable peaks at greater quantities such as $1,000, $1,500 and so forth.

```
#visualizing the distributions for the first donations
hist(jump$AMT_DONOR_GIFT_FIRST, xlim = c(0,3000), breaks =100000, main = "First Gift for Jump
Donors", col = "steelblue", xlab = "Donation Amount")
hist(no_jump$AMT_DONOR_GIFT_FIRST, breaks = 100000, xlim = c(0,3000), main= "First Gift for No
Jump Donors", xlab = "Donation Amount", col = "tan1")
```

E. Top Giving Unit by Constituency Category and Gender

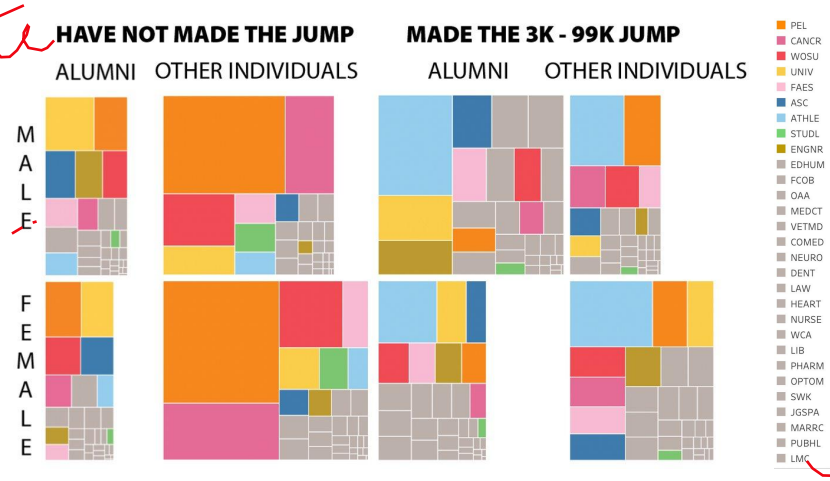


Figure 2.6 Top Giving Unit by Constituency Category and Gender

This visualization was created in Tableau and the code for the random subsets of 20,000 created for the visualization can be found in section 2.A as well as the export of jump donor data. Note that this visualization does not include the gender categories of other and unknown or faculty, staff, parents, or students; it only includes the majority giving categories in order to see differences more clearly. Some

differences between the categories to highlight here are differences among donations to athletics, Pelotonia, and the university.

F. Donor Location

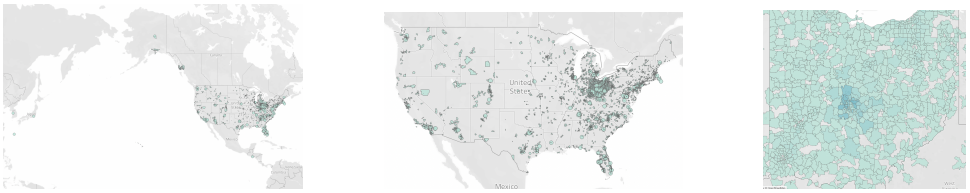


Figure 2.7 Count of Donations by Zip-code for Donors who have not Made the Jump

The Columbus area is notable for the number of donations compared to other places in Ohio, nationally and globally. The concentration in the Columbus area is less for the non-jump donors compared to the jump donors.

Figure 2.8 Count of Donations by Zip-code for Jump Donors

As seen in this figure, there is a higher concentration of jump donors in the Columbus area when compared to other zip-codes and non-jump donors. The graph furthest to the right zooms in on the Columbus area. Particular zip-codes were notable for their number of donations, and a boolean variable was created to mark donors from these zip-codes; the code for this creation can be found in section 2.B.

G. Involvement Before Donation

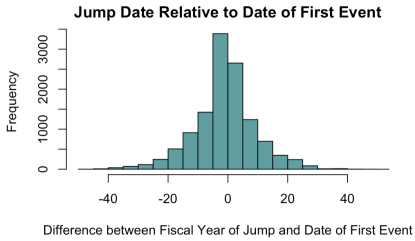


Figure 2.9 Jump Data Relative to Date of First Event

This figure shows a dispersion around zero for the difference between the date recorded for the first event and the date of the jump. Involvement and the donation jump tend to occur around the same period of life.

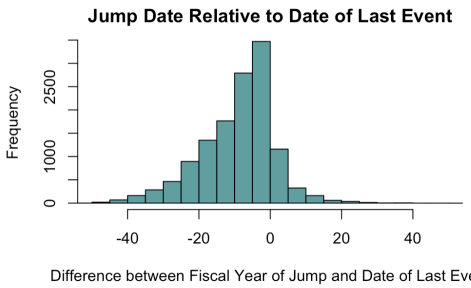


Figure 2.10 Jump Date Relative to Date of Last Event

This histogram shows how donors who jumped years ago tend to continue to go to events after their jump donation.

```
hist(jump$param_ID_FISCALYEAR - jump$FY_EVENTS_FIRST, xlim = c(-50,50), breaks = 300, col = "cadetblue", main = "Jump Date Relative to Date of First Event", xlab = "Difference between Fiscal Year of Jump and Date of First Event")
```

H. Multiple Addresses

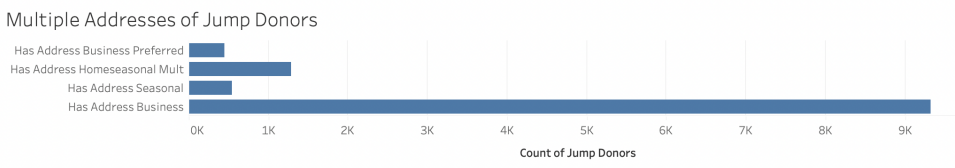


Figure 2.11 Multiple Addresses of Jump Donors

Using 22,335 rows, the whole donor jump dataset, the number of business addresses and multiple homes compared to the non-jump random dataset is notable.

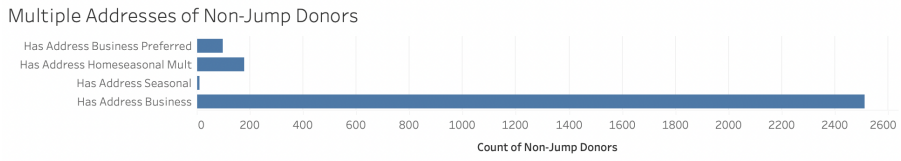


Figure 2.12 Multiple Addresses of Non-Jump Donors

Using a random sample of 20,000 rows, there is a comparably less number of business addresses and multiple homes recorded for non-jump donors.

I. First Donation Mediums

The first donation medium section used the 'DonorGivingAggregate_RelativeDate_3K99K' data set. The variables of interest are the first donation mediums in order to explore whether the donors' first donation medium will contribute to a jump in the future. First, a histogram of jumped and not jumped donors in terms of the first donation method was drawn. From the histogram it was discovered that no jump donors

made a gift through practice credit. Three variables, donation through email, giving day solicitation, and Wexner arts and museum, had little data relative to the jumped category. Thus, these four categories are not in the model. A random and balanced subset with roughly equal rows for jumped and not jumped was created, removing all the rows where the donor's last giving year was after 2000.

```
d3k99k["param_ID_DATECOMPARETYPE"][d3k99k["param_ID_DATECOMPARETYPE"] == 2] <- 0
set.seed(314159)
d3k99k$IS_DONOR_APPEAL_DM_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_DM_FIRST)
d3k99k$IS_DONOR_APPEAL_ONLINE_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_ONLINE_FIRST)
d3k99k$IS_DONOR_APPEAL_EMAIL_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_EMAIL_FIRST)
d3k99k$IS_DONOR_APPEAL_PHONE_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_PHONE_FIRST)
d3k99k$IS_DONOR_APPEAL_GIVINGDAY_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_GIVINGDAY_FIRST)
d3k99k$IS_DONOR_APPEAL_NEWDIGITAL_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_NEWDIGITAL_FIRST)

d3k99k$IS_DONOR_APPEAL_EVENT_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_EVENT_FIRST)
d3k99k$IS_DONOR_APPEAL_OSUEMPLCMPN_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_OSUEMPLCMPN_FIRST)
d3k99k$IS_DONOR_APPEAL_BIG6_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_BIG6_FIRST)
d3k99k$IS_DONOR_APPEAL_AG_FIRST = as.factor(d3k99k$IS_DONOR_APPEAL_AG_FIRST)
d3k99k$IS_DONOR_TRIBUTE_FIRST = as.factor(d3k99k$IS_DONOR_TRIBUTE_FIRST)
d3k99k$IS_DONOR_PAY_THRU_PRACTICECREDIT_FIRST =
as.factor(d3k99k$IS_DONOR_PAY_THRU_PRACTICECREDIT_FIRST)
d3k99k$IS_DONOR_TO_PEL_FIRST = as.factor(d3k99k$IS_DONOR_TO_PEL_FIRST)
d3k99k$IS_DONOR_TO_WCA_FIRST = as.factor(d3k99k$IS_DONOR_TO_WCA_FIRST)
d3k99k$IS_DONOR_TO_WOSU_FIRST = as.factor(d3k99k$IS_DONOR_TO_WOSU_FIRST)
d3k99k$IS_DONOR_TO_LICPLATE_FIRST = as.factor(d3k99k$IS_DONOR_TO_LICPLATE_FIRST)
d3k99k$param_ID_DATECOMPARETYPE = as.factor(d3k99k$param_ID_DATECOMPARETYPE)
set.seed(111)
trainup = upSample(x = d3k99k[,-ncol(d3k99k)], y = d3k99k$param_ID_DATECOMPARETYPE)
rows_to_keep <- sample(1:nrow(trainup), size=20000, replace=FALSE)
d3k99k_subset <- trainup[rows_to_keep,]
set.seed(1)
# Only keep rows where donors' last giving is after 2000
new_data = d3k99k_subset$FY_DONOR_GIFT_LAST > 2000
# select columns I'll use for model
donation_ways_subset = d3k99k_subset[new_data,c(2,91:100,107,110,113,121,124,127)]
# Deleted insignificant variables / only few people ever made donation through this method
donation_ways_subset = donation_ways_subset[,c(-4,-6,-13,-15)]
```

J. Donations in Fiscal Years

The fiscal year donations and how they contributed to each jump through visualizations were explored. In order to aid the analysis, the range of which each individual made the jump was considered; in order to avoid looking at very large gifts, a donation of 3,000 defines a jump. The purpose of the violin plot is to look at what small donation amounts preceded the jump. These remained largely consistent on a year to year basis, and the peaks represent the amounts which people would tend to give. How these donation amount benchmarks would contribute to making the jump was considered. The data values were filtered down to values greater than 100 and less than 200. The reason this is done is so we can look at the giving behaviors of donors who donate small, yet do it consistently and see how their giving behaviors pan out on at each contribution amount per year basis. What was observed is that both people who jumped and didn't jump exhibited very similar giving behaviors with regards to donation amounts in a year. This could be used to identify certain amounts to be used when targeting people who are donating to the university.

```
don_agg  = dt.fread("~\\Downloads\\202201 Data Capstone SP2022 - Donor Giving Aggregate - as
```

```
of today.csv").to_pandas()
const_abbr  = dt.fread("~\\Downloads\\202201 Data Capstone SP2022 - Constituent Table
Abbreviated.csv").to_pandas()
don_agg_3_99 = dt.fread("~\\Downloads\\SP2022 Capstone - 202201 -
DonorGivingAggregate_RelativeDate_3K99K.csv").to_pandas()

filt_don_agg_2day = don_agg[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY12_TOTAL',
'AMT_DONOR_LFY11_TOTAL',
       'AMT_DONOR_LFY10_TOTAL', 'AMT_DONOR_LFY09_TOTAL',
       'AMT_DONOR_LFY08_TOTAL', 'AMT_DONOR_LFY07_TOTAL',
       'AMT_DONOR_LFY06_TOTAL', 'AMT_DONOR_LFY05_TOTAL',
       'AMT_DONOR_LFY04_TOTAL', 'AMT_DONOR_LFY03_TOTAL',
       'AMT_DONOR_LFY02_TOTAL', 'AMT_DONOR_LFY01_TOTAL','AMT_DONOR_CFY00_TOTAL']]

filt_don_agg_3_99 =
don_agg_3_99[["CONSTITUENTLOOKUPID","IS_DONOR_APPEAL_DM","IS_DONOR_APPEAL_ONLINE","IS_DONOR_AP
PEAL_EMAIL","IS_DONOR_APPEAL_PHONE","IS_DONOR_APPEAL_GIVINGDAY","IS_DONOR_APPEAL_NEWDIGITAL","
IS_DONOR_APPEAL_EVENT","IS_DONOR_APPEAL_BIG6","IS_DONOR_APPEAL_AG","param_ID_DATECOMPARETYPE",
"param_ID_DATEDIMID"]]

filt_data_2 = filt_don_agg_3_99.merge(filt_don_agg_2day,
left_on="CONSTITUENTLOOKUPID",right_on="CONSTITUENTLOOKUPID")

donations_by_years = [filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY12_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY11_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY10_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY09_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY08_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY07_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY06_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY05_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY04_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY03_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY02_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_LFY01_TOTAL']],
filt_data_2[['CONSTITUENTLOOKUPID','AMT_DONOR_CFY00_TOTAL']]]

person_type = filt_data_2[['CONSTITUENTLOOKUPID','param_ID_DATECOMPARETYPE']]
names = ['AMT_DONOR_LFY12_TOTAL', 'AMT_DONOR_LFY11_TOTAL',
         'AMT_DONOR_LFY10_TOTAL', 'AMT_DONOR_LFY09_TOTAL',
          'AMT_DONOR_LFY08_TOTAL', 'AMT_DONOR_LFY07_TOTAL',
           'AMT_DONOR_LFY06_TOTAL', 'AMT_DONOR_LFY05_TOTAL',
           'AMT_DONOR_LFY04_TOTAL', 'AMT_DONOR_LFY03_TOTAL',
   'AMT_DONOR_LFY02_TOTAL', 'AMT_DONOR_LFY01_TOTAL','AMT_DONOR_CFY00_TOTAL']
for i in range(0,13):
    donations_by_years[i]["years ago"] = 12-i
    donations_by_years[i].rename(columns={names[i]:"donation"},inplace=True)
big_df_for_violin_plot = pd.concat(donations_by_years, ignore_index = True)
big_df_for_violin_plot_final =
pd.merge(big_df_for_violin_plot,person_type,how='inner',on="CONSTITUENTLOOKUPID",validate="man
y_to_many")

plt.figure(figsize=(20,10))
sns.violinplot(x="years ago", y="donation", hue="param_ID_DATECOMPARETYPE",

data=big_df_for_violin_plot_final[big_df_for_violin_plot_final.donation.isin(range(100,200))],
palette="husl", split=True)
```

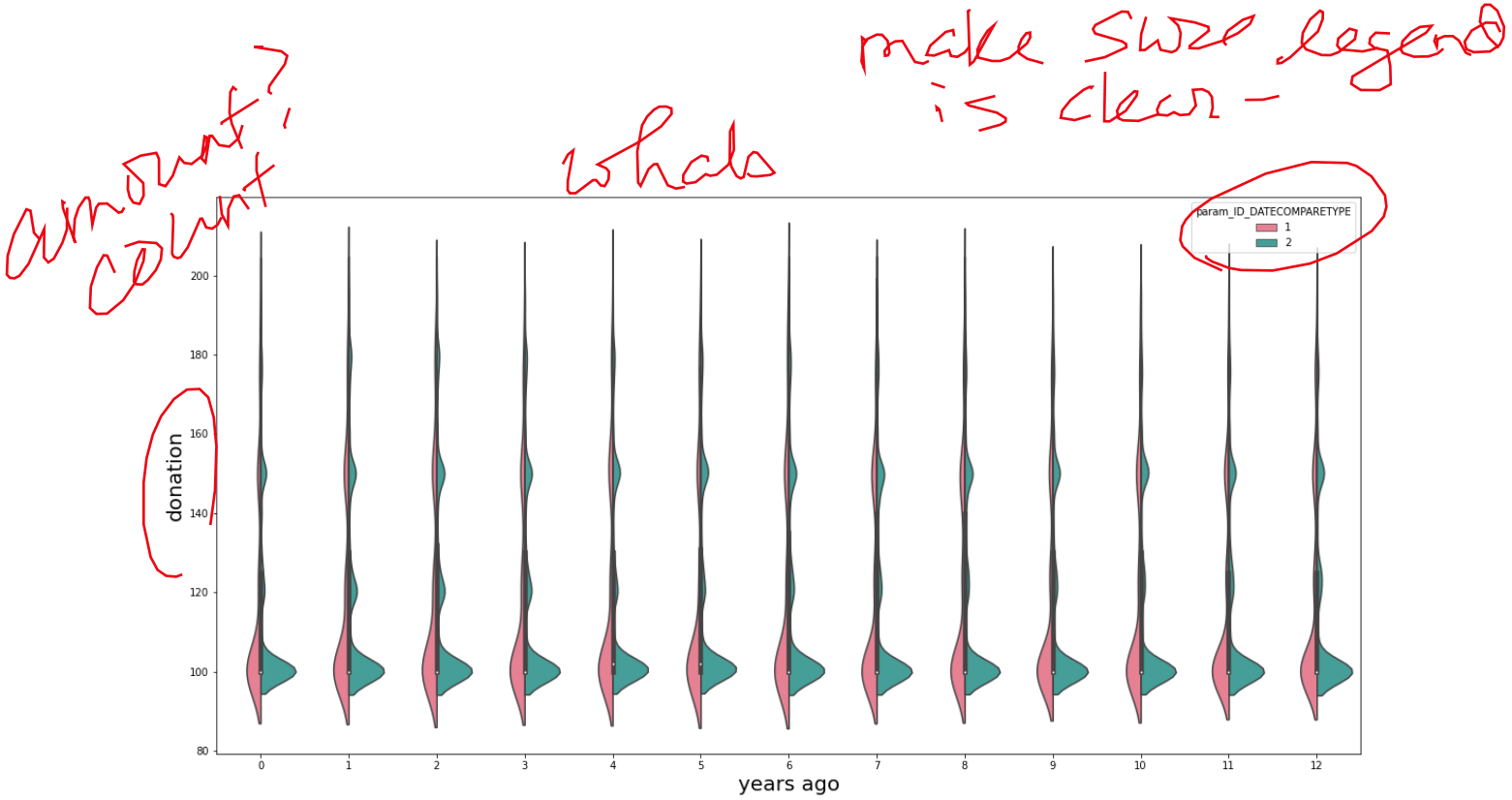


Figure 2.13 Donation Amount Over Years

Donations for people who donated within the range of 100 to 200 on a year to year basis consistently. In pink are the donors who made the jump and in green are the donors who did not make the jump.

K. First and Largest Donations

A random subset of the data provided by Dr. Metzger was used for this part of data exploration. The motivation was to explore whether the constituents that made the jump have previously made donations close to the $3,000.00 amount. Another point of interest was the increase in the amount donated between the first and largest donation. In order to analyze a relationship between the first donation made by a constituent and the largest donation made, a plot was created. To decrease the overlap between the points and better see the patterns present, noise was added (note, because of this, some points had negative values).

```
#adding noise
mu1, sigma1 = 0, 100
mu2, sigma2 = 0, 50
noise1 = np.random.normal(mu1, sigma1, [d3k99k_sub.shape[0]])
noise2 = np.random.normal(mu2, sigma2, [d3k99k_sub.shape[0]])
d3k99k_sub["AMT_DONOR_GIFT_LARGEST_Noise"] = d3k99k_sub["AMT_DONOR_GIFT_LARGEST"]+noise2
d3k99k_sub["AMT_DONOR_GIFT_FIRST_Noise"] = d3k99k_sub["AMT_DONOR_GIFT_FIRST"]+noise1
# plot first vs largest donations
sns.set_style("whitegrid")
plt.rcParams["figure.figsize"] = [13.00, 6.0]
plt.rcParams["figure.autolayout"] = True
fig, ax =plt.subplots(1,2)
g1 = sns.scatterplot(data=d3k99k_sub.loc[d3k99k_sub['param_ID_DATECOMPARETYPE'] ==1],
x="AMT_DONOR_GIFT_LARGEST_Noise", y="AMT_DONOR_GIFT_FIRST_Noise",
                hue= "param_ID_DATECOMPARETYPE", style="param_ID_DATECOMPARETYPE",
                    palette = "Set1", alpha = 0.25, legend = False, ax=ax[1])
g1.set_title("Largest notation vs. First donation for people \n who moved to next stage",
fontsize=20)
g1.set_xlabel("Largest gift amount (with noise)", fontsize = 15)
```

```
g1.set_ylabel("First gift amount (with noise)", fontsize = 15)
g1.set_ylim(-500, 3100)
g1.set_xlim(-300, 3100)

g2 = sns.scatterplot(data=d3k99k_sub.loc[d3k99k_sub['param_ID_DATECOMPARETYPE'] ==2],
x="AMT_DONOR_GIFT_LARGEST_Noise", y="AMT_DONOR_GIFT_FIRST_Noise",
               hue= "param_ID_DATECOMPARETYPE", style="param_ID_DATECOMPARETYPE",
                   palette = "Set2", alpha = 0.25, legend = False, ax=ax[0])
g2.set_title("Largest notation vs. First donation for people \n who did not move to next
stage", fontsize = 20)
g2.set_xlabel("Largest gift amount (with noise)", fontsize = 15)
g2.set_ylabel("First gift amount (with noise)", fontsize = 15)
g2.set_ylim(-500, 3100)
g2.set_xlim(-300, 3100)
fig.show()
jhighlar = d3k99k_sub.loc[(d3k99k_sub["AMT_DONOR_GIFT_LARGEST"] > 2000) &
(d3k99k_sub["AMT_DONOR_GIFT_FIRST"] < 500) & (d3k99k_sub['param_ID_DATECOMPARETYPE'] ==1)]
#jumped and high largest
njhighlar = d3k99k_sub.loc[(d3k99k_sub["AMT_DONOR_GIFT_LARGEST"] > 2000) &
(d3k99k_sub["AMT_DONOR_GIFT_FIRST"] < 500) & (d3k99k_sub['param_ID_DATECOMPARETYPE'] ==2)] #no
jump and high largest
j = d3k99k_sub.loc[d3k99k_sub['param_ID_DATECOMPARETYPE'] ==1] #jumped
nj = d3k99k_sub.loc[d3k99k_sub['param_ID_DATECOMPARETYPE'] ==2] #did not jump
#proportions
p1 = jhighlar.shape[0]/j.shape[0]
p2 = njhighlar.shape[0]/nj.shape[0]
```

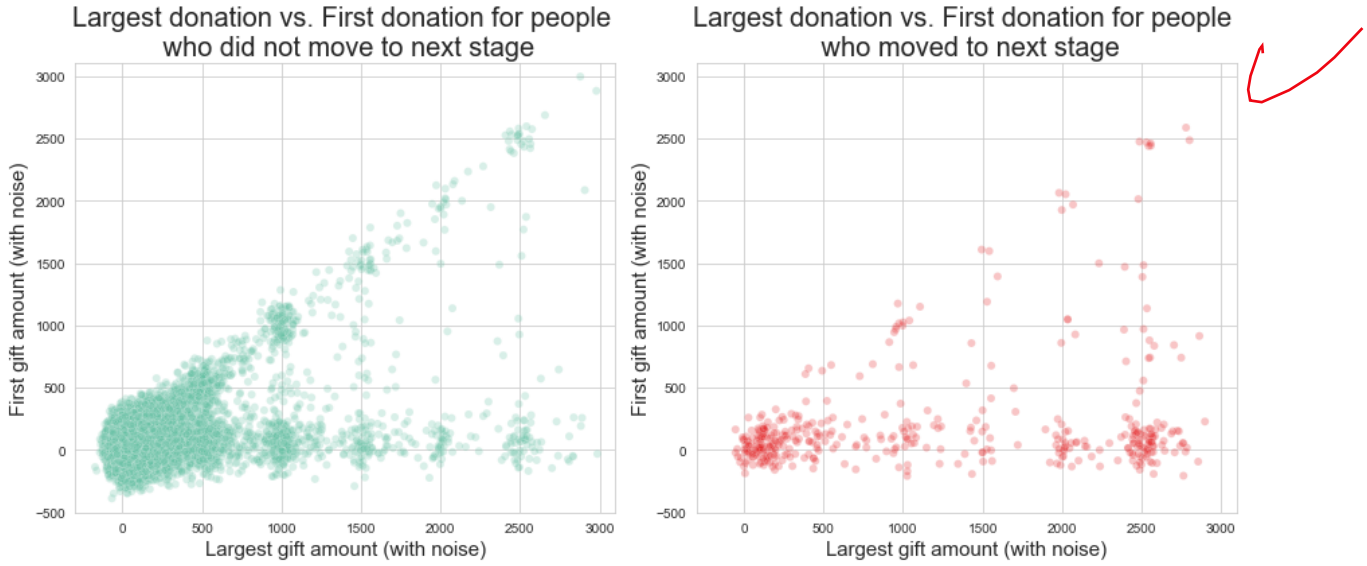


Figure 2.14 Largest vs. First Donations for Two Groups

It was observed that given the first donation was under $500.00, 0.3% of people who have not made the jump donated above $2,000.00. In contrast, out of people who did make the jump, given the first donation was under $500.00, 23.5% made at least one donation over $2,000.00. Therefore, the variables for the first and largest donations were used in the classification models.

L. Alumni Variables

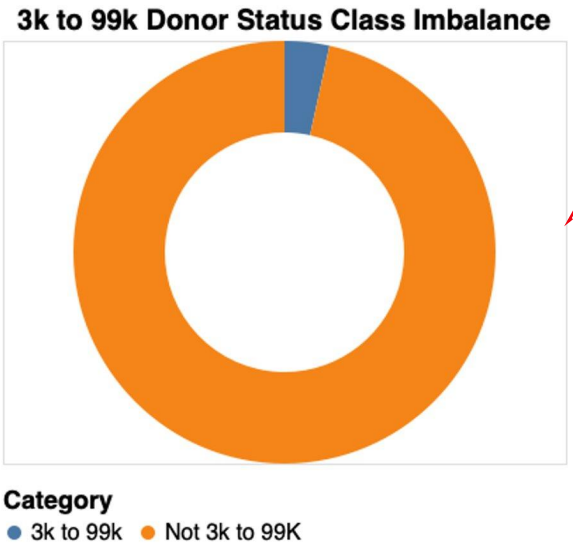
It is logical to assume that alumni may be more likely to give because of the experiences, friendships, and opportunities OSU provided them. Thus, the subset of only OSU alumni was considered to investigate the impact of these experiences on 3k to 99k donor transitions. Any variables describing an alumnus' current and previous involvement in the university were isolated to use as the first set of predictor variables. Involvement score variables were not included to preserve easy interpretability.

```
# Load Relevant Libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import altair as alt
```

The file named 'alumni_reduced_variables' was created by joining the constituents data on the transitions data via the 'CONSTITUENTLOOKUPID' column, and isolating the alumni-related variables. Bucketing was done on the variables 'IS_DEGREE_COLLEGECDO_XXXX' to combine categories with low representation (<10%). Professional colleges were grouped together, and the remaining colleges of low representation went to an 'other' category. OSU degree level variables (ex. IS_DEGREE_UGRAD_ANY) were reduced to the variables 'JUST_UNDER', 'JUST_GRAD', and 'UNDERANDGRAD' for those posessing only an undergraduate degree (from any college), only a graduate degree, or both, respectively. An individual was placed in only one of these categories. In addition, non-binary variables were included in the predictor set over binary counterparts if such existed.

```
# Read data and store in Data Frame
df = pd.read_csv('alumni_reduced_variables2.csv')
```

Investigation of the 3k to 99k donor classes in alumni revealed a class imbalance. Only 4% of alumni are 3k to 99k donors.



mosaic plot of alumni / non alumni?

Figure 2.15 Alumni Donor Imbalance

```
# Class Imbalance Visualization
```

```
# Calculate number of alumni who did and did not jump
df = pd.read_csv('alumni_reduced_variables2.csv')
not_jumped = sum(df['param_ID_DATECOMPARETYPE'] ==2)
jumped = sum(df['param_ID_DATECOMPARETYPE'] == 1)
# Store in data frame
df2 = pd.DataFrame({'Jumped': [jumped], 'Not Jumped': [not_jumped]}).reset_index()
df2
# Reformat df2 to work with Altair graphing functions
df2 = pd.DataFrame({'Category':['3k to 99k', 'Not 3k to 99K'], 'Value': [11779, 331879]})
# Calculate Proportions
df2['prop'] = df2['Value'].divide(11779 + 331879)
df2['prop'] = df2['prop'].multiply(100)
# Create Donut Plot
alt.Chart(df2).mark_arc(innerRadius=85).encode(
theta=alt.Theta(field="Value", type="quantitative"),
color=alt.Color(field="Category", type="nominal"),
).configure_legend(
titleFontSize=18,
labelFontSize=16,
orient = 'bottom',
direction = 'horizontal'
).configure_title(
fontSize = 20,
).properties(title = "3k to 99k Donor Status Class Imbalance")
```

Models were fitted using all alumni. Models were trained on data with a down-sampled majority class (3:1 majority to minority call ratio), and tested on regular data. A random forest was built using the full predictor list and evaluated using the f1 and recall scores of the 3k to 99k class. Variables of low importance according to change in gini impurity were removed and the model was rebuilt using the new subset of predictors. Performance of the two models were compared, and the model with the highest f1 score was chosen for future use. If the two models had very similar f1 scores, the model with fewer predictors was chosen. This process was repeated until removing any features caused a significant decrease in performance.

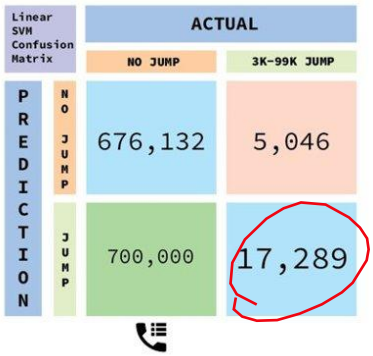
```
# import sklearn random forest libraries
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel

# Where filtering of variables occurrs - remove variables of low importance here
# Note: param_ID_DATECOMPARETYPE must stay
# This is final set of predictors
cols = ['IS_TWOALUMHH',
'IS_RELATEDTO_FUND',
'N_RELATEDTO_ALUM',
'IS_DEGREE_RESFELLOW_ANY',
'param_ID_DATECOMPARETYPE',
'N_DEGREE_LEVEL',
'IS_OSUPARENT_EVER']
# isolate relevant columns
features = df[cols]
# Get one hot encodings for categorical variables (required)
features = pd.get_dummies(features)
# Turn true and false into 1s and 0s (required)
features = features*1
```

```
def random_forest(features):
# Under Sampling ------------------------------------------------------------------------
# Notes: it seems crazy to train on 90%, but much of this will be removed
# to achieve 3:1 class ratio
train = features.sample(frac = 0.90)
test = features.drop(train.index)
major = train[train['param_ID_DATECOMPARETYPE'] == 2]
minor = train[train['param_ID_DATECOMPARETYPE'] == 1]
num_to_remove = len(major) - (3*len(minor))
temp = major.sample(n = num_to_remove)
major.drop(temp.index, inplace = True)
train = pd.concat([major, minor], ignore_index=True)
# Shuffle the DataFrame rows
train = train.sample(frac = 1)
# Final training data sets
y_train = train['param_ID_DATECOMPARETYPE']
X_train = train.drop(columns = ['param_ID_DATECOMPARETYPE'])
# Final testing data sets
y_test = test['param_ID_DATECOMPARETYPE']
X_test = test.drop(columns = ['param_ID_DATECOMPARETYPE'])
# Model Fitting -------------------------------------------------------------------------
clf = RandomForestClassifier(n_estimators=100, bootstrap = True, oob_score = True)
clf.fit(X_train, y_train)
pred_labels = X_test.columns
# Calculate and print feature importances-----------------------------------------
importances = list()
var = list()
val = list()
print('Feature Importances: ' )
for feature in zip(pred_labels, clf.feature_importances_):
var.append(feature[0])
val.append(feature[1])
importances.append(feature)
print(feature)
# Graph feature importance--------------------------------------------------------
temp = pd.DataFrame({'Predictor': var, 'Importance': val})
temp.sort_values(by ='Importance', inplace = True)
plt.bar('Predictor', 'Importance', data = temp)
plt.xticks(rotation = 90)
plt.show()
# Print performance metrics -------------------------------------------------------
print('Confusion Matrix: ')
# Test (predict)
y_test_pred = clf.predict(X_test)
# Confusion matrix
cm = metrics.confusion_matrix(y_test, y_test_pred, labels = [1,2])
print(cm)
# Classification report - calculates f1 score, recall, accuracy, precision, etc.
print('Classification Report: ')
print(metrics.classification_report(y_test, y_test_pred))
# Calculate train and test error
y_train_pred = clf.predict(X_train)
mse_train = metrics.mean_squared_error(y_train, y_train_pred)
mse_test = metrics.mean_squared_error(y_test, y_test_pred)
print("Train MSE: {} Test MSE: {}".format(mse_train, mse_test))
return cm  #Returning confusion matrix for use later
# Suppress harmless warning in random_forest call
```

III. Model Analysis

A. Linear SVM

The Linear SVM created was trained on a small subset with a 3:1 jump to no jump ratio, and the data used was trimmed down to those who donated $0 to $5,000 in a single year. The 3:1 ratio was made to handle the imbalance of classes; a model that had used the natural balance may have gotten a 99% accuracy, but that was only because it identified nearly all observations as having not made the jump. The purpose of this was to train small, and identify a more long run trend of people who consistently donate a certain amount per year instead of a lot of money up front. The model is high risk high reward. The purpose of the trimmed down subset was to see how the donors, who donate in small increments, have the potential to make the jump.



make these percentages so it's easier to

Figure 3.1 Linear SVM Confusion Matrix

The lower accuracy (18%) allows for leniency for the high-risk high-reward model. The target rate was over 50%, i.e., about 700,000 candidates who have not made the jump yet have been identified. A possibility is to perhaps market for smaller donation amounts to then lead up to a jump. Overall, consider prior giving history of small amount increments.

B. K-means & Hierarchical

The selected variables of interest for the K-means & Hierarchical models were as follows:
VAL_DONOR_UPORDOWN: Count the number of fiscal years with increased giving amount for each constituent.
N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD: Also called the BlackJack score which was used to measure the fluctuation of the donation amount during the past 12 fiscal years.

```
load("/Users/wusongyuan/Desktop/Downloads/donors (5).RData")

df_const <- data.frame(constituents)
df_3k99k <- data.frame(X3k99k_new)

#Merging the constituents and 3k99k data frames into one called df2
df2 <- merge(x=df_const, y=df_3k99k, by.x="ID_CONSTITUENTLOOKUPID",
by.y="CONSTITUENTLOOKUPID",all=FALSE)

#Distinguishing the constituents who made the jump and didn't
```

```
jump = df2 %>%
   filter(param_ID_DATECOMPARETYPE==1)
no_jump = df2 %>%
   filter(param_ID_DATECOMPARETYPE==2)


#Sampling the data frame
jump_sample = jump[sample(1:nrow(jump), 3000),]
no_jump_sample = no_jump[sample(1:nrow(no_jump), 9000),]

#Selecting features which I'm interested in
jump_sample = jump_sample %>%
  select(VAL_DONOR_UPORDOWN,N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD)
no_jump_sample = no_jump_sample %>%
  select(VAL_DONOR_UPORDOWN,N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD)

jump_sample$label = 1
no_jump_sample$label = 2
data1 = rbind(jump_sample,no_jump_sample)

##Clustering Analysis-Here we use Distance-based methods: Find groups that minimize the
distance between members within the group while maximizing the distance from members of other
groups. First we use Hierarchical clustering and then we use k-means.

distances1 = round(dist(data1,method = "euclidean"),2)
clust1 = hclust(distances1,method = "ward.D2")
clusters1 = cutree(clust1,k = 2)
data1c = cbind(data1,clusters1)
ggplot(data=data1c,aes(x=VAL_DONOR_UPORDOWN,y=N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD,color=fa
ctor(clusters1)))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F,color=F)+
  geom_text(aes(label=rownames(data1c)),,hjust=-1.5,vjust=0.5)+
  ggtitle("Hierarchical Clustering (With Jump)")

library(psych)
temp = data.frame(cluster = factor(clusters1),
           factor1 = fa(data1,nfactors = 2,rotate = 'varimax')$scores[,1],
           factor2 = fa(data1,nfactors = 2,rotate = 'varimax')$scores[,2])
ggplot(temp,aes(x=factor1,y=factor2,col=cluster))+
  ggtitle("Hierarchical Scatterplot")+
  geom_point()
library(cluster)
clusplot(data1,
         clusters1,
         color=T,shade=T,labels=4,lines=0,main='Hierarchical Cluster Plot')
```

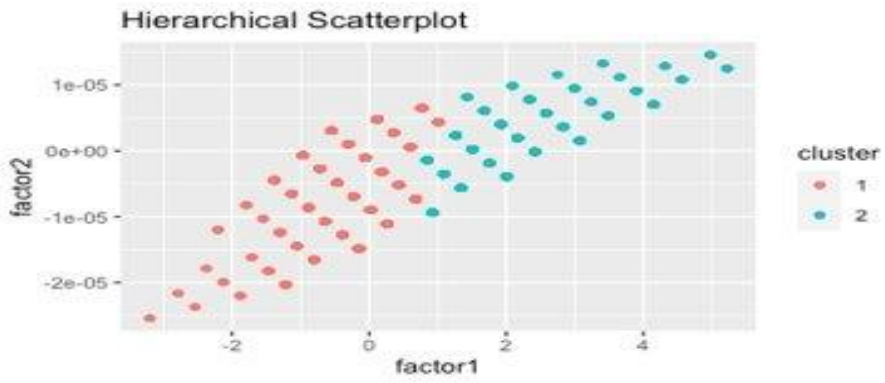


Figure 3.2 Hierarchical Scatterplot

To express the clusters on a scatter plot, the data was flattened into 2 dimensions by conducting a factor analysis with varimax rotation. By feeding two variables VAL_DONOR_UPORDOWN and N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD into the hierarchical clustering model, it could divide the original data into two different groups which was marked by red and blue in the plot.

```
##Then we use k-means - This disorganized approach to clustering produces similar quality of
clusters to hierarchical clustering but much faster
set.seed(617)
km1 = kmeans(x = data1,centers = 2,iter.max=10000,nstart=25)
km1$totss == km1$betweenss + km1$tot.withinss
k_segments = km1$cluster
library(psych)
temp = data.frame(cluster = factor(k_segments),
           factor1 = fa(data1,nfactors = 2,rotate = 'varimax')$scores[,1],
           factor2 = fa(data1,nfactors = 2,rotate = 'varimax')$scores[,2])
ggplot(temp,aes(x=factor1,y=factor2,col=cluster))+
  geom_point()+
  ggtitle("K-means Scatterplot")
```

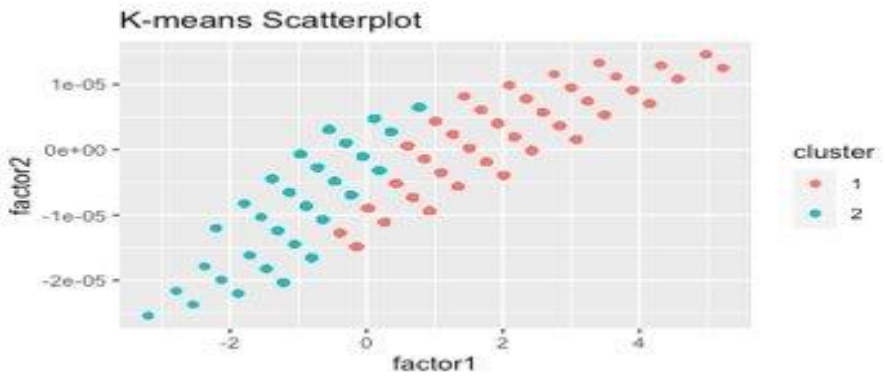


Figure 3.3 K-means Scatterplot

K-means Scatterplot finds groups that are most compact, in terms of the mean sum-of-squares deviation of each observation from the multivariate center of its assigned group. K-means clustering begins by arbitrarily placing centroids in the data and then iterating from that point to the final solution. Then, another clear boundary was drawn successfully in the plot shown above.

Introduction to cluster-then-predict approach:

After performing these clustering algorithms on our original dataset with two features, data points were successfully divided into two clusters. The clear boundary in the current K-means scatter plot shows that current selected variables VAL_DONOR_UPORDOWN and N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD work well to distinguish constituents who made the jump and those who didn't. However, the clustering model is an unsupervised learning model which means that it couldn't display the detailed numeric measurement or metric, such as recall rate, F1 score, accuracy, and precision. Therefore, it was necessary to connect the current cluster label results from the clustering model with the random forest model by pasting a new feature called cluster label of each data point, and then feeding this into the logistic model.

```
# Python Code Version
drive.mount('/content/drive')
train_path = '/content/drive/MyDrive/data1.csv'
df1 = pd.read_csv(train_path)
X = df1.drop('Unnamed: 0', axis=1)
X = X.drop('label',axis=1)
y = df1['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=90210)

# Running the k-means model again and getting the cluster labels for each data point in
whatever training and testing dataset.
def get_clusters(X_train: pd.DataFrame, X_test: pd.DataFrame, n_clusters: int) ->
Tuple[pd.DataFrame, pd.DataFrame]:
   clustering = KMeans(n_clusters=n_clusters, random_state=8675309)
   clustering.fit(X_train)
   # apply the labels
   train_labels = clustering.labels_
   X_train_clstrs = X_train.copy()
   X_train_clstrs['clusters'] = train_labels

   # predict labels on the test set
   test_labels = clustering.predict(X_test)
   X_test_clstrs = X_test.copy()
   X_test_clstrs['clusters'] = test_labels
   return X_train_clstrs, X_test_clstrs

X_train_clstrs, X_test_clstrs = get_clusters(X_train, X_test, 2)

#Scaling the features chosen by us before fitting into the model to make sure that all
features are on the same numerical scale.
def scale_features(X_train: pd.DataFrame, X_test: pd.DataFrame) -> Tuple[pd.DataFrame,
pd.DataFrame]:
   scaler = StandardScaler()
   to_scale = [col for col in X_train.columns.values]
   scaler.fit(X_train[to_scale])
   X_train[to_scale] = scaler.transform(X_train[to_scale])

   # predict z-scores on the test set
   X_test[to_scale] = scaler.transform(X_test[to_scale])

   return X_train, X_test
X_train_scaled, X_test_scaled = scale_features(X_train_clstrs, X_test_clstrs)

# to divide the df by cluster, we need to ensure we use the correct class labels, we'll use
pandas to do that
```

```
train_clusters = X_train_scaled.copy()
test_clusters = X_test_scaled.copy()
train_clusters['y'] = y_train
test_clusters['y'] = y_test
# locate the "0" cluster
train_0 = train_clusters.loc[train_clusters.clusters < 0] # after scaling, 0 went negtive
test_0 = test_clusters.loc[test_clusters.clusters < 0]
y_train_0 = train_0.y.values
y_test_0 = test_0.y.values
# locate the "1" cluster
train_1 = train_clusters.loc[train_clusters.clusters > 0] # after scaling, 1 dropped slightly
test_1 = test_clusters.loc[test_clusters.clusters > 0]
y_train_1 = train_1.y.values
y_test_1 = test_1.y.values
# the base dataset has no "clusters" feature
X_train_base = X_train_scaled.drop(columns=['clusters'])
X_test_base = X_test_scaled.drop(columns=['clusters'])
# drop the targets from the training set
X_train_0 = train_0.drop(columns=['y'])
X_test_0 = test_0.drop(columns=['y'])
X_train_1 = train_1.drop(columns=['y'])
X_test_1 = test_1.drop(columns=['y'])
datasets = {
   'base': (X_train_base, y_train, X_test_base, y_test),
   'cluster feature': (X_train_scaled, y_train, X_test_scaled, y_test),
   'cluster0': (X_train_0, y_train_0, X_test_0, y_test_0),
   'cluster1': (X_train_1, y_train_1, X_test_1, y_test_1),
}
#Built a random forest model on these 4 datasets and store 4 testing metrics of each dataset
into the data frame.

def run_exps(datasets: dict) -> pd.DataFrame:
   '''
   runs experiments on a dict of datasets
   '''
   # initialize a Random Forest classifier
     model = RandomForestClassifier(n_estimators=100)

   dfs = []
   results = []
   conditions = []
   scoring = ['accuracy','precision_weighted','recall_weighted','f1_weighted']

   for condition, splits in datasets.items():
     X_train = splits[0]
     y_train = splits[1]
     X_test = splits[2]
     y_test = splits[3]
     kfold = model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
     cv_results = model_selection.cross_validate(model, X_train, y_train, cv=kfold,
scoring=scoring)
     clf = model.fit(X_train, y_train)
     y_pred = clf.predict(X_test)
     print(condition)
     print(classification_report(y_test, y_pred))
     results.append(cv_results)
     conditions.append(condition)

     this_df = pd.DataFrame(cv_results)
     this_df['condition'] = condition
     dfs.append(this_df)
```

```
    final = pd.concat(dfs, ignore_index=True)
    print("final",final)
    results_long = pd.melt(final,id_vars=['condition'],var_name='metrics', value_name='values')
    print("results_Long",results_long)
    time_metrics = ['fit_time','score_time']
    results = results_long[~results_long['metrics'].isin(time_metrics)]
    results = results.sort_values(by='values')
    return results

df = run_exps(datasets)
plt.figure(figsize=(20, 12))
sns.set(font_scale=2.5)
g = sns.boxplot(x="condition", y="values", hue="metrics", data=df, palette="Set3")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
L = plt.legend()
L.get_texts()[0].set_text('Precision')
L.get_texts()[1].set_text('F1 Score')
L.get_texts()[2].set_text('Accuracy')
L.get_texts()[3].set_text('Recall')
plt.title('Comparison of Dataset by Classification Metric')
```

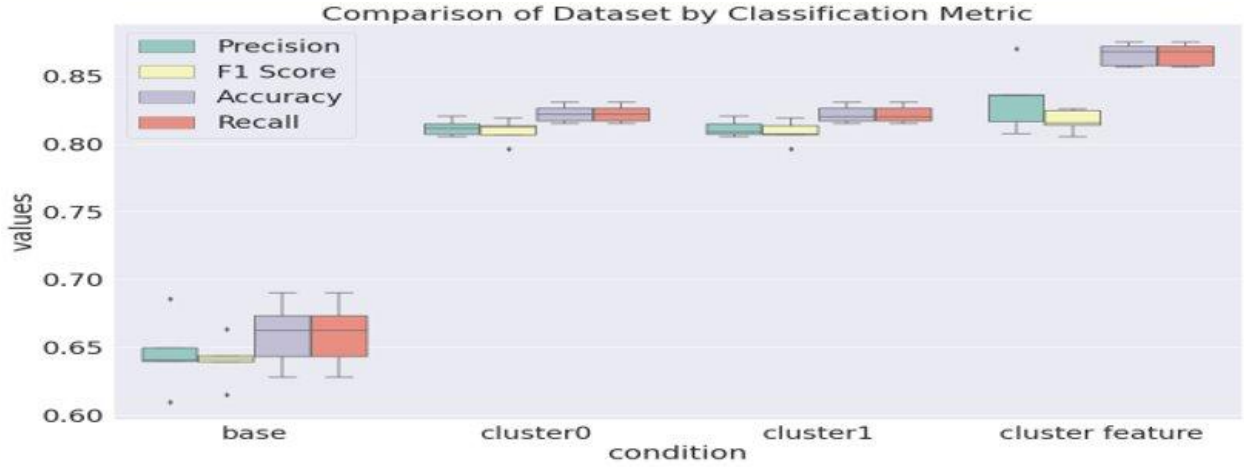


Figure 3.4 Comparison of Dataset by Classification Metric

The cluster-then-predict process divided the original dataset into 4 different datasets and then fed them into the random forest model respectively to get the precision, accuracy, F1 and recall rate of each dataset. By comparison, it could be concluded that a dataset with the clustering label as a new feature achieves the highest overall metric, and there is a huge jump from the base dataset without clustering labels to the featured dataset with clustering labels. Therefore, adding the clustering labels into the original feature set would increase the performance of the classification model.

C. Alumni Random Forest

```
# Suppress harmless warning in random_forest call
import warnings
warnings.filterwarnings('ignore')
# Create random forest
cm = random_forest(features)
```

The variables IS_TWOALUMHH, IS_RELATEDTO_FUND, N_RELATEDTO_ALUM, IS_DEGREE_RESFELLOW_ANY, N_DEGREE_LEVEL, and IS_OSUPARENT_EVER are the predictors that had the highest importance and were thus included in the final model.

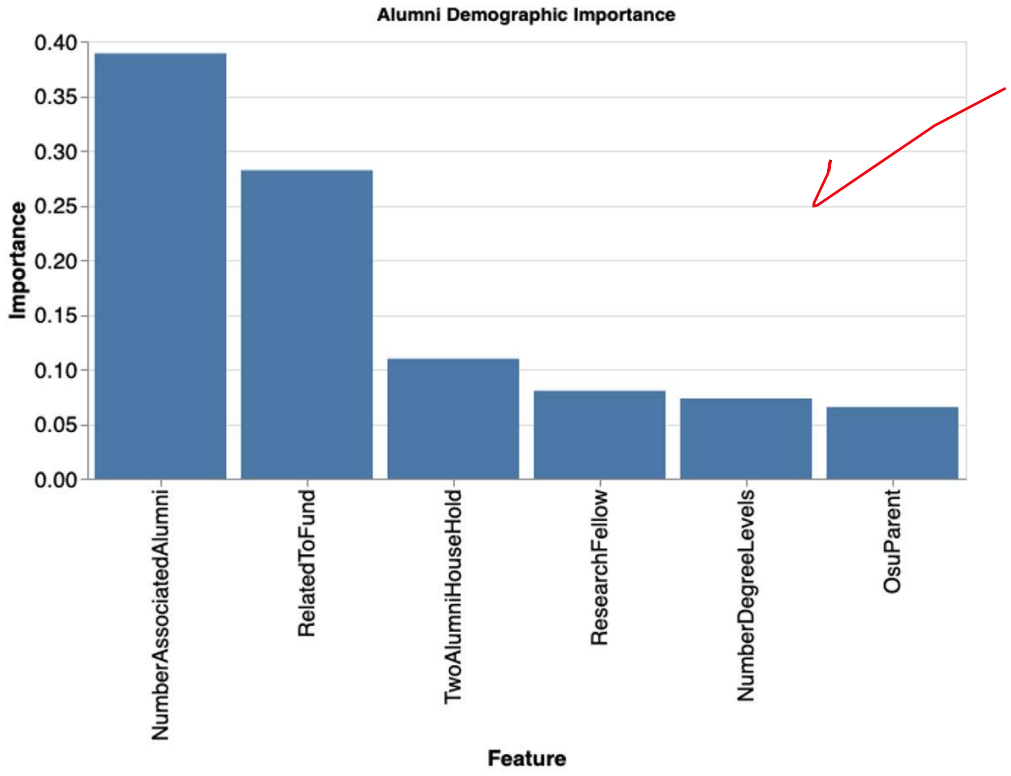


Figure 3.5 Alumni Variable Importances

```
# FeatureImportance.xlsx is a file containing feature importances based on one run of
# of the model. It is from the same run that produced figure 3.4 .
df = pd.read_excel('FeatureImportance.xlsx')
df.sort_values(by = ['Importance'], inplace = True)
alt.Chart(df).mark_bar().encode(
alt.X('Feature:N',sort=alt.EncodingSortField(field="Importance", order = 'descending')),
alt.Y('Importance:Q')
).configure_axis(
labelFontSize = 15,
titleFontSize = 15
).configure_axisX(
).properties(width = 600, title = 'Alumni Demographic Importance')
```

The final random forest produced a minority f1 score of 0.25 on average. Based on one run, this model classified 3.09% of alumni as 3k to 99k prospects. This model also misclassfied 2.54% of alumni who actually are 3k to 99k donors. Note: The confusion matrix below was created in adobe llustrator based on one run of the model.

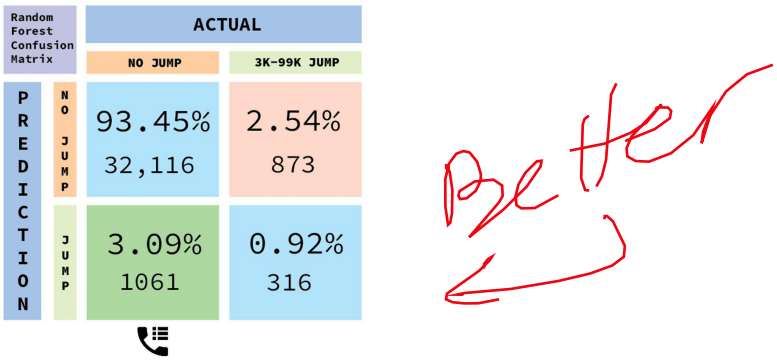


Figure 3.6 Alumni Random Forest Confusion Matrix

Although the performance of this model isn't great it is still useful. When the first random forest model was created with 18 variables its performance was poor. When I limited the model down to just 6 important predictor variables, its performance barely changed. Considering this, if someone wishes to use alumni features to classify 3k to 99k donors, he/she should focus on the variables that had highest importance and therefore contributed the most to correct classifications. Based on figure 3.4, the office of advancement should target alumni who are married to alumni, OSU parents, associated with or in contact with several alumni, related to funds, research fellows, and have more than one degree level. If I could do this analysis again, I would create a new variable called 'IS_OSU_FAMILY', which is true when an alumnus is or was an OSU parent, is married to an OSU alumnus, and has a N_RELATEDTO_ALUM value that is above a certain threshold.

D. Random Forest for Notable Predictors

A random forest was used to measure the predictive accuracy of notable predictors in our analysis. The variables included are the income, amount of the first gift, number of events attended, blackjack score, number of fiscal years with increased giving, count of alumni related to record, OSU alumni status, donates through digital channel first, related to a fund, donates through event first, and high contribution zip-code location.

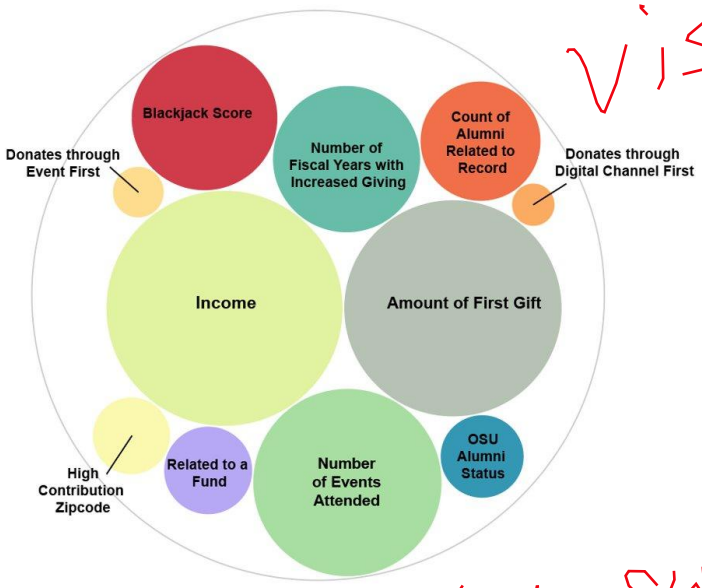


Figure 3.7 MeanDecreaseGini for Random Forest

This figure visualizes the MeanDecreaseGini variable from the random forest variable importance output. For example, income had the largest mean decrease in gini. The higher the variance the more mis-classification there is; thus, lower values of the gini index yield better classification. The mean decrease in gini relates to the decrease in model accuracy if the variable was removed; thus, larger values hold more predictive influence or accuracy in the random forest model.

```
#random forest model predictors
#creating a new zipcode boolean for 1 (true) if in a high contribution zipcodes
df$new_bool_zip <- (df$CAT_ADDRESS_ZIP5 %in% zipcodes)
rf.data = df %>% select(IS_DONOR_APPEAL_NEWDIGITAL_FIRST, IS_DONOR_APPEAL_EVENT_FIRST,
VAL_DONOR_UPORDOWN, N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD, N_RELATEDTO_ALUM,
IS_RELATEDTO_FUND, N_EVENTS, IS_OSUALUMNI,AMT_WEALTH_DEMO_INCOME_RAW_TA, AMT_DONOR_GIFT_FIRST,
new_bool_zip, param_ID_DATECOMPARETYPE)
#removing NA values
rf.data.clean = rf.data %>% drop_na()
#using a smaller subset for random forest analysis
rf.data.xs = sample_n(rf.data.clean, 20000)
#donor status for param_ID_DATECOMPARETYPE is made to be a factor
rf.data.xs$param_ID_DATECOMPARETYPE = as.factor(rf.data.xs$param_ID_DATECOMPARETYPE)
rf = randomForest(param_ID_DATECOMPARETYPE~., data = rf.data.xs, mtry = 3, importance = TRUE)
#random forest R outputs
importance(rf)
varImpPlot(rf)
```

```
> importance(rf)
                                          0        1
IS_DONOR_APPEAL_NEWDIGITAL_FIRST          5.088704  7.949396
IS_DONOR_APPEAL_EVENT_FIRST               6.636835 17.917481
VAL_DONOR_UPORDOWN                       10.110770 17.753383
N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD   8.662819 19.929289
N_RELATEDTO_ALUM                        -16.242608 29.584910
IS_RELATEDTO_FUND                         6.066990 31.900365
N_EVENTS                                 -7.986076 73.952061
IS_OSUALUMNI                             15.622044 -8.060214
AMT_WEALTH_DEMO_INCOME_RAW_TA             4.572162 14.392266
AMT_DONOR_GIFT_FIRST                     12.604228 26.506548
new_bool_zip                              8.350969  2.747155
                                         MeanDecreaseAccuracy
IS_DONOR_APPEAL_NEWDIGITAL_FIRST                     6.345325
IS_DONOR_APPEAL_EVENT_FIRST                         10.410791
VAL_DONOR_UPORDOWN                                  14.325471
N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD             15.614361
N_RELATEDTO_ALUM                                    -2.175093
IS_RELATEDTO_FUND                                   20.926877
N_EVENTS                                            16.614919
IS_OSUALUMNI                                        12.642780
AMT_WEALTH_DEMO_INCOME_RAW_TA                       10.338307
AMT_DONOR_GIFT_FIRST                                21.574061
new_bool_zip                                         8.813020

                                         MeanDecreaseGini
IS_DONOR_APPEAL_NEWDIGITAL_FIRST                 5.449863
IS_DONOR_APPEAL_EVENT_FIRST                      9.227571
VAL_DONOR_UPORDOWN                              72.783033
N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD         82.689842
N_RELATEDTO_ALUM                                52.697388
IS_RELATEDTO_FUND                               27.945935
N_EVENTS                                       133.394125
IS_OSUALUMNI                                    23.859225
AMT_WEALTH_DEMO_INCOME_RAW_TA                  196.753059
AMT_DONOR_GIFT_FIRST                           167.187453
new_bool_zip                                    23.760025
```

E. Generalized Logistic Regression for Notable Predictors

The model was trained on a 3:1 ratio of donors who had not made the jump to donors who have, creating a 30,000 to 10,000 random sample. The model was also tested on a 3:1 ratio, creating a 9,000 to 3,000 random sample of test subjects. A threshold of 0.5 probability was used to categorize a donor as having made the jump or not.

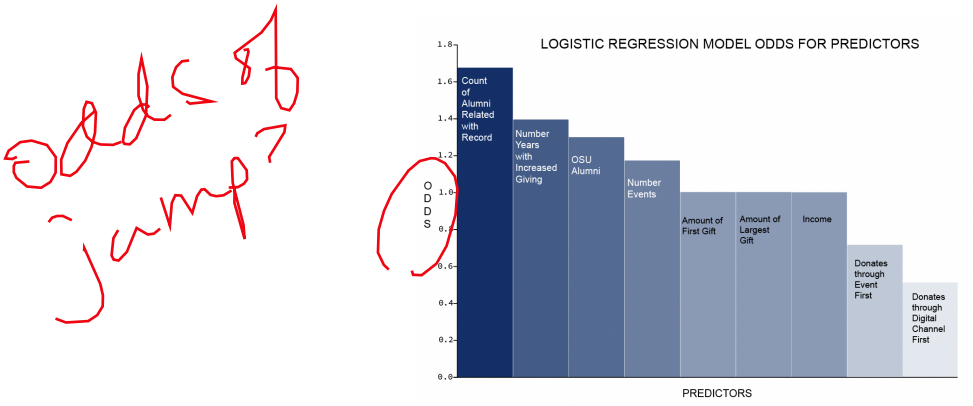


Figure 3.8 Logistic Regression Model Odds

The figure above depicts the odds of the predictor coefficients from the logit model with an output of 1 being the constituent jumped vs a 0 meaning the donor did not jump.

Note that this figure includes Blackjack Score; however, the Blackjack Score was ultimately removed from the logistic regression model due to insignificance (at alpha level 0.10). It is kept in the figure due to its significance as a predictor in the random forest model and clustering to visualize it in comparison to other variables.

A logistic regression model was created using the variables "DigitalFirst", "OSUAlumniStatus", "NumberEvents", "EventAppealFirst", "NumberofIncreasedGivingYears", "NumberRelatedAlumni", "RelatedtoFund", "Income", "FirstGiftAmount", and "Amount Largest" to predict “DateCompareType”.

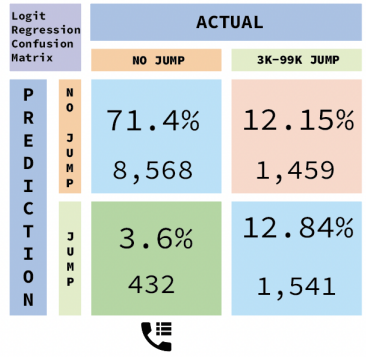


Figure 3.9 Logistic Regression Model Confusion Matrix

This is the confusion matrix based on the code output found below. In green are the potential donors to contact.

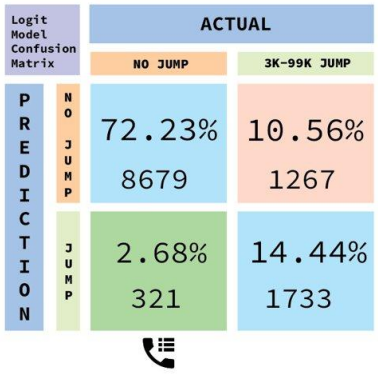


Figure 3.10 Logistic Regression Model Confusion Matrix including Blackjack Score

This figure is included to show the increase in model accuracy when the Blackjack Score is included despite its coefficient having a p-value greater than 0.10, the significance level.

```
#train and test set up in a 3:1 ratio
jump_sample = jump[sample(1:nrow(jump), 10000),]
no_jump_sample = no_jump[sample(1:nrow(no_jump), 30000),]

train = sample(1:nrow(jump_sample),nrow(jump_sample)*.7)
jump.train = jump_sample[train,]
jump.test = jump_sample[-train,]
```

```
train = sample(1:nrow(no_jump_sample),nrow(no_jump_sample)*.7)
no.jump.train = no_jump_sample[train,]
no.jump.test = no_jump_sample[-train,]
#combing the jump and no jump donors in the ratios defined above for the train and test data
final_train_data = rbind(jump.train, no.jump.train)
final_test_data = rbind(jump.test, no.jump.test)
#using significant variables
test = cbind(final_test_data$param_ID_DATECOMPARETYPE,
final_test_data$IS_DONOR_APPEAL_NEWDIGITAL_FIRST, final_test_data$IS_OSUALUMNI,
final_test_data$N_EVENTS, final_test_data$IS_DONOR_APPEAL_EVENT_FIRST,
final_test_data$N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD, final_test_data$N_RELATEDTO_ALUM,
final_test_data$AMT_WEALTH_DEMO_INCOME_RAW_TA, final_test_data$AMT_DONOR_GIFT_FIRST,
final_test_data$AMT_DONOR_GIFT_LARGEST)

colnames(test) = c("DateCompareType", "DigitalFirst", "OSUAlumniStatus","NumberEvents",
"EventAppealFirst","NumberofIncreasedGivingYears", "NumberRelatedAlumni", "Income",
"FirstGiftAmount", "Amount Largest")
test = data.frame(test)
#train set up
train = cbind(final_train_data$param_ID_DATECOMPARETYPE,
final_train_data$IS_DONOR_APPEAL_NEWDIGITAL_FIRST, final_train_data$IS_OSUALUMNI,
final_train_data$N_EVENTS, final_train_data$IS_DONOR_APPEAL_EVENT_FIRST,
final_train_data$N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD, final_train_data$N_RELATEDTO_ALUM,
final_train_data$AMT_WEALTH_DEMO_INCOME_RAW_TA, final_train_data$AMT_DONOR_GIFT_FIRST,
final_train_data$AMT_DONOR_GIFT_LARGEST)

colnames(train) = c("DateCompareType", "DigitalFirst", "OSUAlumniStatus","NumberEvents",
"EventAppealFirst","NumberofIncreasedGivingYears", "NumberRelatedAlumni", "Income",
"FirstGiftAmount", "Amount Largest")

train = data.frame(train)
#model summary
train$DateCompareType = as.factor(train$DateCompareType)
test$DateCompareType = as.factor(test$DateCompareType)
train_model = glm(DateCompareType ~., data = train, family = "binomial")
summary(train_model)
#table(train$DateCompareType)
```

```
Call:
glm(formula = DateCompareType ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.4852  -0.6204  -0.3996   0.5791   2.5813

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                -2.872e+00  4.950e-02 -58.019  < 2e-16
DigitalFirst               -8.705e-01  1.613e-01  -5.396 6.83e-08
OSUAlumniStatus             7.561e-02  4.367e-02   1.731  0.08337
NumberEvents                1.628e-01  8.495e-03  19.162  < 2e-16
EventAppealFirst           -3.081e-01  1.451e-01  -2.123  0.03377
NumberofIncreasedGivingYears  4.595e-01  1.114e-02  41.233  < 2e-16
NumberRelatedAlumni           5.430e-01  3.123e-02  17.386  < 2e-16
Income                        4.677e-06  1.942e-07  24.091  < 2e-16
FirstGiftAmount               3.630e-05  1.265e-05   2.868  0.00412
Amount.Largest                4.605e-06  1.743e-06   2.642  0.00823
```

```
(Intercept)                  ***
DigitalFirst                 ***
OSUAlumniStatus              .
NumberEvents                 ***
EventAppealFirst             *
NumberofIncreasedGivingYears ***
NumberRelatedAlumni          ***
Income                       ***
FirstGiftAmount              **
Amount.Largest               **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 20761  on 16007  degrees of freedom
Residual deviance: 13899  on 15998  degrees of freedom
  (11992 observations deleted due to missingness)
AIC: 13919

Number of Fisher Scoring iterations: 8

predict.glm.compareType    0    1
                       0 8568 1459
                       1  432 1541
```

F. Naive Bayes Model on Categorical and Quantitative Variables

Good method

Using the same variables as in logistic regression, a Naive Bayes model was then fit to predict which category (did not make a jump or made a jump to 3k99k level) the data most likely came from. A 3:1 ratio with oversampling was used for training data to account for class imbalance. The model was run and evaluated 5,000 times and the mean recall and accuracy were calculated to be 67% and 95% respectively.

```
df1 = ppl_sub[['ID_CONSTITUENTLOOKUPID','N_EVENTS', 'CAT_ADDRESS_ZIP5', 'IS_OSUALUMNI',
'N_RELATEDTO_ALUM',
           'IS_RELATEDTO_FUND', 'AMT_WEALTH_DEMO_INCOME_RAW_TA']]
d3k99k_sub = pd.merge(d3k99k_sub, df1, left_on="CONSTITUENTLOOKUPID",
right_on="ID_CONSTITUENTLOOKUPID")

# zip code boolean
zips = ['43085','43125', '43015', '43230', '43081', '43082',
'43017','43065','43016','43220','43235','43026',
             '43123', '43214', '43221']
d3k99k_sub['CLOSE_ZIP']  = d3k99k_sub.apply(lambda x: True if x['CAT_ADDRESS_ZIP5'] in zips
else False,axis=1)

#select columns for the model
d3k99k_mod = d3k99k_sub[['param_ID_DATECOMPARETYPE', 'IS_DONOR_APPEAL_NEWDIGITAL_FIRST',
'IS_OSUALUMNI',
                         'N_EVENTS', 'IS_DONOR_APPEAL_EVENT_FIRST',
'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'IS_RELATEDTO_FUND',
'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST',
'VAL_DONOR_UPORDOWN', 'CLOSE_ZIP']]
```

```
d3k99k_mod['AMT_WEALTH_DEMO_INCOME_RAW_TA'] =
d3k99k_mod['AMT_WEALTH_DEMO_INCOME_RAW_TA'].fillna(0)


attr = d3k99k_mod.iloc[:, 1:]
cls = d3k99k_mod.iloc[:, 0]

#run cross validation
acc = []
rec = []
warnings.filterwarnings('ignore')
for i in range(1000):
    x_train, x_test, y_train, y_test = train_test_split(attr, cls, test_size = 0.2)

    oversample = RandomOverSampler(sampling_strategy=0.33)

    x_train_over, y_train_over = oversample.fit_resample(x_train, y_train)

    scaler = StandardScaler()
    scaler.fit(x_train_over[['N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST' ]])
    x_train_over[['N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST' ]] =
scaler.transform(x_train_over[[
                       'N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST']])
    x_test[['N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST']] =
scaler.transform(x_test[[
                       'N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST']])

    #Split train into categorical and ratio
    x_train_categ = x_train_over[['IS_DONOR_APPEAL_NEWDIGITAL_FIRST', 'IS_OSUALUMNI',
'IS_DONOR_APPEAL_EVENT_FIRST',
                              'IS_RELATEDTO_FUND' ,  'CLOSE_ZIP']]
    x_train_ratio = x_train_over[['N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST']]
    #Split test into categorical and ratio
    x_test_categ = x_test[['IS_DONOR_APPEAL_NEWDIGITAL_FIRST', 'IS_OSUALUMNI',
'IS_DONOR_APPEAL_EVENT_FIRST',
                              'IS_RELATEDTO_FUND' , 'CLOSE_ZIP']]
    x_test_ratio = x_test[['N_EVENTS', 'N_DONOR_INCREASEDGIVINGYEARS_TIMEPERIOD',
                         'N_RELATEDTO_ALUM', 'AMT_WEALTH_DEMO_INCOME_RAW_TA',
                         'AMT_DONOR_GIFT_FIRST', 'AMT_DONOR_GIFT_LARGEST']]
    sz = y_test.shape[0] # how many values we are predicting
    pred_y = [None]*sz # this will keep track of prediction

    mnb = MultinomialNB()
    prob_cat = mnb.fit(x_train_categ, y_train_over).predict_proba(x_test_categ) # n by 2 array
    gnb = GaussianNB()
    prob_ratio = gnb.fit(x_train_ratio , y_train_over).predict_proba(x_test_ratio) # n by 2
array
```

```
    prob = np.multiply(prob_cat, prob_ratio) #probability of each class
    #predict class based on maximum probability
    for i in range(sz):
        prob_i = prob[i, :]
        if np.amax(prob_i) == prob_i[0]:
            pred_y[i] = 1
        if np.amax(prob_i) == prob_i[1]:
            pred_y[i] = 2
    predict_y = np.array(pred_y)
    acc.append(accuracy_score(y_test, predict_y))
    rec.append(recall_score(y_test, predict_y, average='binary'))


#single run
cm = confusion_matrix(y_test, predict_y)
print(cm)

#average recall and accuracy
np.mean(acc)
np.mean(rec)
```

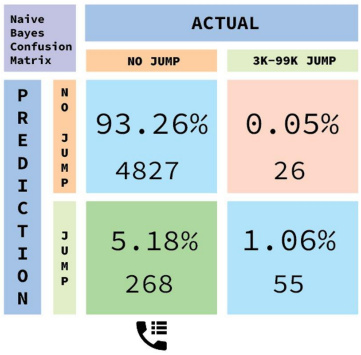


Figure 3.11 Naive Bayes Model Confusion Matrix

The confusion matrix from running the model one time is presented above. The proportion of false negatives (red cell) has decreased in comparison to other models.

G. Random Forest and Boosting for First Donation Mediums

The libraries of 'randomforest' and 'gbm' to build a random forest, boosting trees were imported and used to discover the variables of importance and the influence of each variable on the response. Random forests and boosting algorithms measured the variable of importance differently, but the results were similar, where 'IS_DONOR_APPEAL_NEWDIGITAL_FIRST,' 'IS_DONOR_APPEAL_EVENT_FIRST,' 'IS_DONOR_TRIBUTE_FIRST' had strong influences on the response. However, the direction of effect, positive or negative, is uncertain.

```
rf.donation = randomForest(param_ID_DATECOMPARETYPE~. , data = donation_ways_subset, mtry = 4,
importance = TRUE)
rf.donation
importance(rf.donation)
varImpPlot(rf.donation,n.var = 5,cex = 0.5)
```

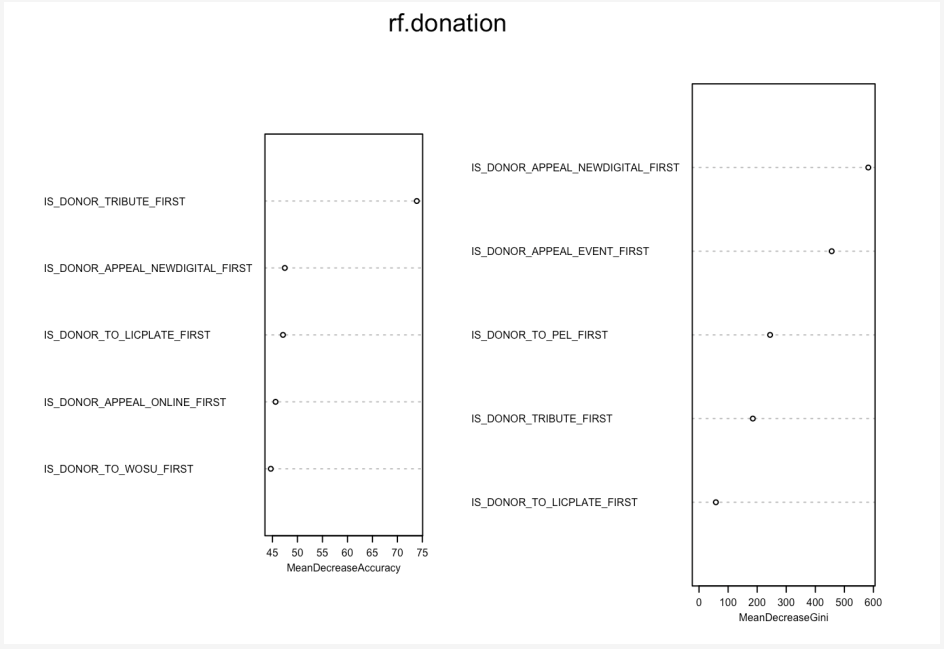


Figure 3.12 Random Forests Output from R

This figure is the visualization of Mean Decrease Accuracy and Mean Decrease Gini for all the variables in the random forest.

```
set.seed(1)
boot.donate.all = gbm(param_ID_DATECOMPARETYPE==1~., data = donation_ways_subset, distribution
= "bernoulli", n.trees = 50, interaction.depth = 3)
summary.gbm(boot.donate.all)
```

```
##                                                           var     rel.inf
## IS_DONOR_APPEAL_NEWDIGITAL_FIRST   IS_DONOR_APPEAL_NEWDIGITAL_FIRST 51.94935971
## IS_DONOR_APPEAL_EVENT_FIRST             IS_DONOR_APPEAL_EVENT_FIRST 20.92600930
## IS_DONOR_TRIBUTE_FIRST                       IS_DONOR_TRIBUTE_FIRST 11.92417168
## IS_DONOR_TO_LICPLATE_FIRST                IS_DONOR_TO_LICPLATE_FIRST  4.30724638
## IS_DONOR_TO_WOSU_FIRST                       IS_DONOR_TO_WOSU_FIRST  4.17049115
## IS_DONOR_APPEAL_ONLINE_FIRST           IS_DONOR_APPEAL_ONLINE_FIRST  2.35241231
## IS_DONOR_APPEAL_AG_FIRST                  IS_DONOR_APPEAL_AG_FIRST  1.76873108
## IS_DONOR_APPEAL_PHONE_FIRST             IS_DONOR_APPEAL_PHONE_FIRST  1.40978662
## IS_DONOR_APPEAL_DM_FIRST                  IS_DONOR_APPEAL_DM_FIRST  0.64732200
## IS_DONOR_APPEAL_BIG6_FIRST                IS_DONOR_APPEAL_BIG6_FIRST  0.37573688
## IS_DONOR_TO_PEL_FIRST                        IS_DONOR_TO_PEL_FIRST  0.10885321
## IS_DONOR_APPEAL_OSUEMPLCMPN_FIRST IS_DONOR_APPEAL_OSUEMPLCMPN_FIRST  0.05987968
```

Barplot would be clearer

Figure 3.13 Boosting Summary Output from R

This figure contains the measures of the relative influence on all the variables in the boosting tree algorithm.

H. Logistic Regression for First Donation Mediums

Logistic regression was chosen to discover the relationship between dependent and response variables. The result differs from the random forest and boosting. Although 'IS_DONOR_APPEAL_NEWDIGITAL_FIRST,' 'IS_DONOR_APPEAL_EVENT_FIRST,' 'IS_DONOR_TRIBUTE_FIRST' had a strong predictive power on the response, they do not have a positive correlation with the response. For example, if the donor made their first donation through an event, no evidence indicates they are likely to jump. From the logistic model, 'IS_DONOR_TO_PEL_FIRST' and 'IS_DONOR_APPEAL_AG_FIRST' have positive coefficients relating to the response, meaning there is evidence that donors who made their first donation through Pelotonia and Annual giving are likely to jump in respect to all the considered variables. There is no substantial evidence that donors who made their first donation through the rest of methods have the potential to jump.

34% of people who made the first donation through pelotonia and 33% who made the first donation through the Annual Giving appeal code have not made the jump. A possible suggestion for the Office of Advancement is to send them a follow-up email about the donation options or invite them to the next event. Pelotonia is a typical good way to raise funds and enables donors to gain better donation experiences through the event. Pelotonia raises funds for cancer research that needs a large amount of money; thus, it provides the donor with options for donating large amounts. At the same time, donating through events does not have large donation options. They may donate $100 every month but find it difficult to make a significant donation at once. It would also be helpful to invite potential donors to events like pelotonia to let them know other donation options and promote large donation amounts.

```
glm.fit.whole = glm(param_ID_DATECOMPARETYPE == 1~ ., data = donation_ways_subset,family =
"binomial")
summary(glm.fit.whole)
```

```
Call:
glm(formula = param_ID_DATECOMPARETYPE == 1 ~ ., family = "binomial",
    data = donation_ways_subset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7172  -0.9378  -0.5026   0.9485   2.3996

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                        0.89831    0.03474  25.856  < 2e-16 ***
IS_DONOR_APPEAL_DM_FIRST1         -0.86029    0.09805  -8.774  < 2e-16 ***
IS_DONOR_APPEAL_ONLINE_FIRST1     -1.26127    0.10811 -11.667  < 2e-16 ***
IS_DONOR_APPEAL_PHONE_FIRST1      -0.87427    0.08905  -9.818  < 2e-16 ***
IS_DONOR_APPEAL_NEWDIGITAL_FIRST1 -1.74922    0.09935 -17.606  < 2e-16 ***
IS_DONOR_APPEAL_EVENT_FIRST1      -2.28640    0.16939 -13.498  < 2e-16 ***
IS_DONOR_APPEAL_OSUEMPLCMPN_FIRST1 -0.64878    0.11405  -5.689 1.28e-08 ***
IS_DONOR_APPEAL_BIG6_FIRST1       -0.92612    0.27668  -3.347 0.000816 ***
IS_DONOR_APPEAL_AG_FIRST1          0.31603    0.08377   3.772 0.000162 ***
IS_DONOR_TRIBUTE_FIRST1           -1.77984    0.07976 -22.314  < 2e-16 ***
IS_DONOR_TO_PEL_FIRST1             1.13197    0.17073   6.630 3.35e-11 ***
IS_DONOR_TO_WOSU_FIRST1           -0.85252    0.06530 -13.056  < 2e-16 ***
IS_DONOR_TO_LICPLATE_FIRST1       -1.49207    0.10413 -14.329  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 21085  on 15324  degrees of freedom
Residual deviance: 17181  on 15312  degrees of freedom
AIC: 17207

Number of Fisher Scoring iterations: 5
```

Figure 3.14 Logistic Regression Summary from R

This figure gives the output from logistic regression showing the coefficients, standard error and p-value for all the variables in the model.

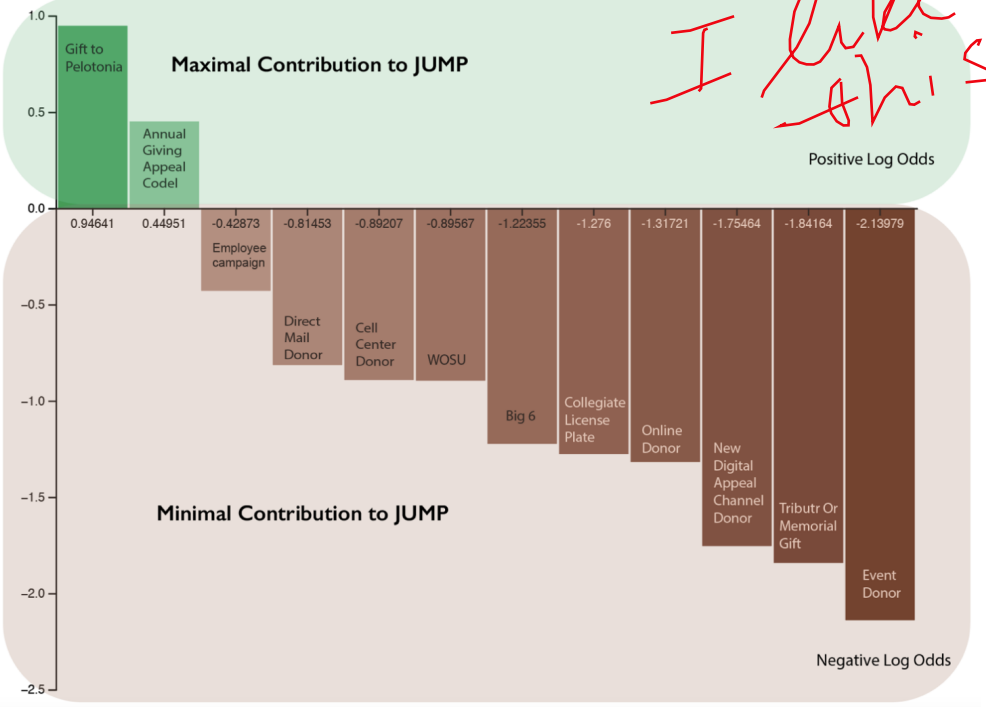


Figure 3.15 Logistic Regression Coefficient Visualization from Adobe Illustrator

This visualization was created in Adobe Illustrator to give a better view of the coefficients for all the variables in logistic regression.

IV. Results and Conclusions

In conclusion, we found predictive power for jump potential in event frequentists, alumni, those with large first donations and donations above $2,000, zip-code, income, first donation experience, and clustering-then-predict for blackjack score and number of years of increased giving. Also, jump donors tend to make the jump when they have attended an event recently. Donors who made their first donation through pelotonia are more likely to make the jump compared to other donation mediums.

V. Code Appendix and Further Explanations

A. Loading R Packages

```
#Loading libraries
library(tidyverse)
library(readr)
library(randomForest)
library(dplyr)
library(caret)
```

B. Loading Python Packages

```
from datatable import dt, f
import numpy as np
import pandas as pd
import warnings

# data visualization
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from statsmodels.graphics.mosaicplot import mosaic
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
# For Naive Bayes model
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from imblearn.over_sampling import RandomOverSampler
# evaluation of models
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay,
recall_score, accuracy_score
from pandasql import sqldf
from sklearn.datasets import make_classification
from google.colab import drive
from sklearn.cluster import KMeans
from typing import Tuple
from sklearn.ensemble import RandomForestClassifier
```