she codes;

GLOBAL ACTIVISM

Eti Mayer
Final Project – NOV 2021

ETI MAYER

- Junior Data Scientist – Certified DS from Bar Ilan
- DS Skills: ML Models, R, Python (DS packages), SQL Server
- Financial back office in an activist hedge fund – BROSH CAPITAL PARTNERS

<u>Experience:</u>

- She Code Final Project – Global Activism
- Israeli Security Agency Hackathon 2021 – DS Team Lead
- Final project in the DS course – NoShow dataset

GLOBAL ACTIVISM

- Domain Field: Global Capital Market
- Main Project Purpose: Predict Activists' Campaign Succession for an Activism Follower Fund
- Data Source: Purchased Database.

  \+ Edgar SEC database

WHAT IS AN ACTIVIST CAMPAIGN?

- An activist investor buys a significant stake in a public company
- **Strategy**: Influence on how the company is run, e.g.:

1. Obtaining seats on BOD.
2. Taking private
3. Etc.

- **Objective**: Creating value to share holders:

>50% net IRR over ref index, S&P500

HOW TO FIND ACTIVISTS' CAMPAIGNS?

Activist acquires more than 5% of voting class

Disclose purchase on a 13D form

Becomes a beneficial owner

Schedule 13D must be filed with the U.S. SEC to disclosed relevant information

THE DATA

In [3]:
```
con <- dbConnect(odbc(),
                 Driver = "SQL Server",
                 Server = "DESKTOP-AAGNMGA\\SQLEXPRESS",
                 Database = "Activism",
                 Trusted_Connection = "True")
```

In [4]:
```
df <- dbReadTable(con,"activist_holdings_v_SC")
head(df)
```

A data.frame: 6 × 189

	Investor.ID	Activist	ActivistHQ	ActivistRegion	Founded	FirstDateInvestedByActivisit	CurrentHolding	StatusCurrent	StatusExisted	DateExited	...	Seat
	<int>	<chr>	<chr>	<chr>	<int>	<chr>	<dbl>	<int>	<int>	<chr>	...	
1	2	Aberdeen Asset Management PLC	UK	WestEurope	1983	2017-01-03	NA	1	0	NA	...	
2	2	Aberdeen Asset Management PLC	UK	WestEurope	1983	2010-09-10	11	1	0	NA	...	
3	2	Aberdeen Asset Management PLC	UK	WestEurope	1983	2015-09-17	NA	0	1	2016-10-10	...	
		Aberdeen										

A Data.Frame: 9299x189 ; **Target=Succession (Categorial 1/0)**

Target: CASE WHEN (a.[Follower Return Annualised (%)]/ a.[S&P Change Annualised (%)]-1>0.5)

THEN (1) ELSE (0) END AS Succession

EDA – Data Visualization

- Using Pandas-Profiling package for statistical and visualized insights.
  Full Report

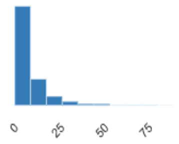
CurrentHolding
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

MISSING

Distinct count	1333
Unique (%)	38.8%
Missing	5863
Missing (%)	63.0%
Infinite	0
Infinite (%)	0.0%

Mean	8.373591327
Minimum	0
Maximum	90.14
Zeros	215
Zeros (%)	2.3%
Memory size	72.8 KiB



Toggle details

Statistics | Histogram(s) | Common values | Extreme values

Quantile statistics

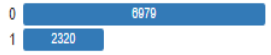
Minimum	0
5-th percentile	0
Q1	1.2875
median	4.99
Q3	10.01
95-th percentile	29.605
Maximum	90.14
Range	90.14
Interquartile range (IQR)	8.7225

Descriptive statistics

Standard deviation	11.12448392
Coefficient of variation (CV)	1.328520044
Kurtosis	11.631764
Mean	8.373591327
Median Absolute Deviation (MAD)	7.263289756
Skewness	2.940309187
Sum	28771.6598
Variance	123.7541425

Succession
Boolean

Distinct count	2
Unique (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	72.8 KiB



Toggle details

Frequency Table

Value	Count	Frequency (%)
0	6979	75.1%
1	2320	24.9%

EDA - Correlation

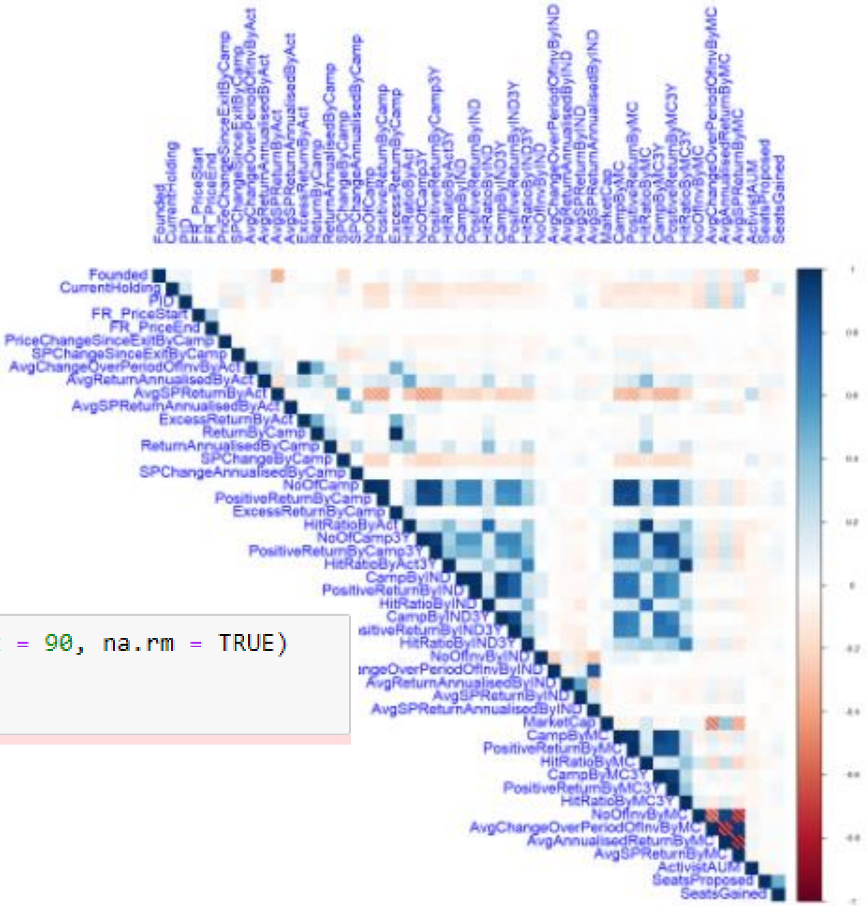
- Correlation matrix between continues variables

```r
res <- NULL

for(i in names(Activism_continuousV)) {
    rw <- NULL
    for(j in names(Activism_continuousV)) {
        rw <- cbind(rw,cor.test(x=df[[i]],y=df[[j]],method="spearman")$estimate)
    }
    res <- rbind(res,rw)
}
row.names(res) <- names(Activism_continuousV)
colnames(res) <- names(Activism_continuousV)
```

```r
corrplot(res, method = "shade",type = "upper", is.corr = TRUE,tl.cex=1.5,tl.col = "Blue", tl.srt = 90, na.rm = TRUE)
options(repr.plot.width = 10, repr.plot.height = 10)

```



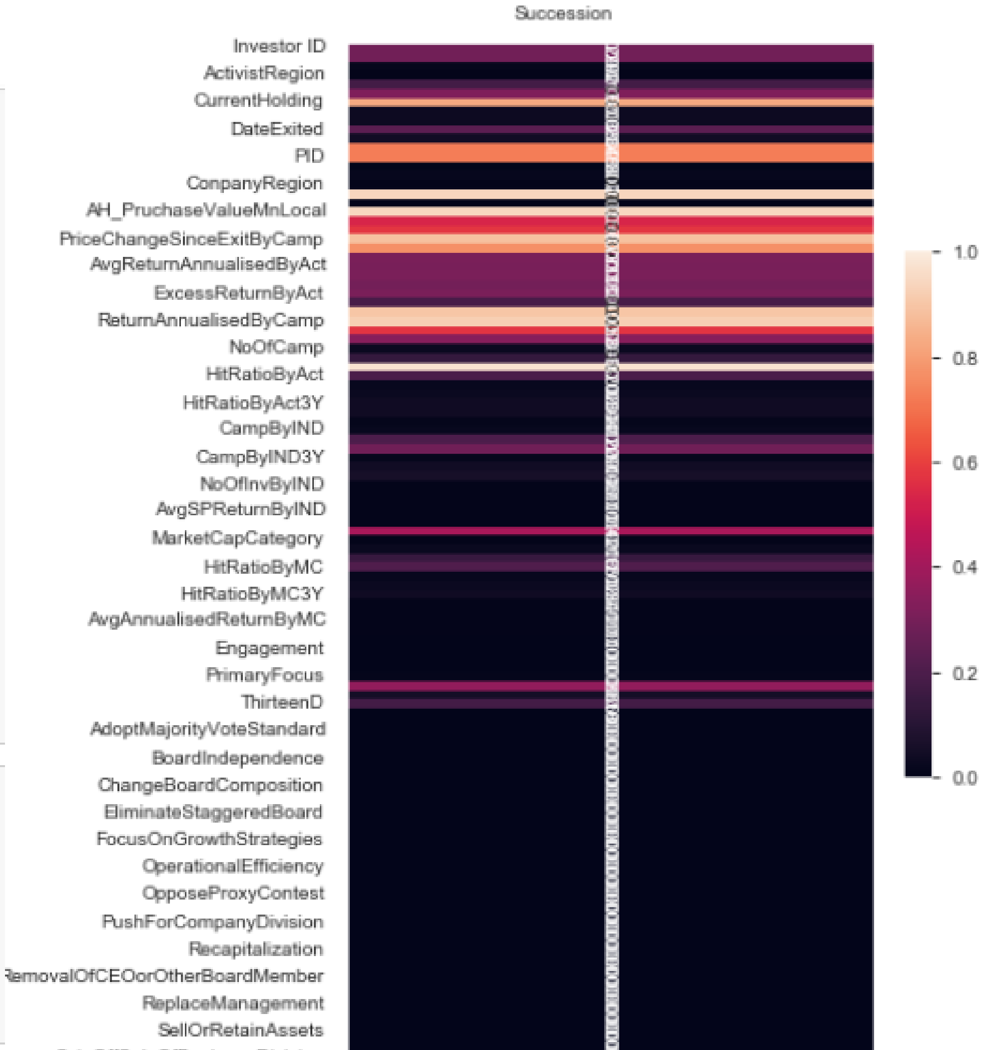
EDA - Correlation

- Correlation between target ‘Succession’ to categorial variables

```python
def conditional_entropy(x,y):
    # entropy of x given y
    y_counter = Counter(y)
    xy_counter = Counter(list(zip(x,y)))
    total_occurrences = sum(y_counter.values())
    entropy = 0
    for xy in xy_counter.keys():
        p_xy = xy_counter[xy] / total_occurrences
        p_y = y_counter[xy[1]] / total_occurrences
        entropy += p_xy * math.log(p_y/p_xy)
    return entropy

def theil_u(x,y):
    s_xy = conditional_entropy(x,y)
    x_counter = Counter(x)
    total_occurrences = sum(x_counter.values())
    p_x = list(map(lambda n: n/total_occurrences, x_counter.values()))
    s_x = ss.entropy(p_x)
    if s_x == 0:
        return 1
    else:
        return (s_x - s_xy) / s_x
```

```python
theilu = pd.DataFrame(index=['Succession'],columns=df.columns)
columns = df.columns
for j in range(0,len(columns)):
    u = theil_u(df['Succession'].tolist(),df[columns[j]].tolist())
    theilu.loc[:,columns[j]] = u
theilu.fillna(value=np.nan,inplace=True)
plt.figure(figsize=(20,5))
sns.heatmap(theilu,annot=True,fmt='.2f')
plt.show()
```



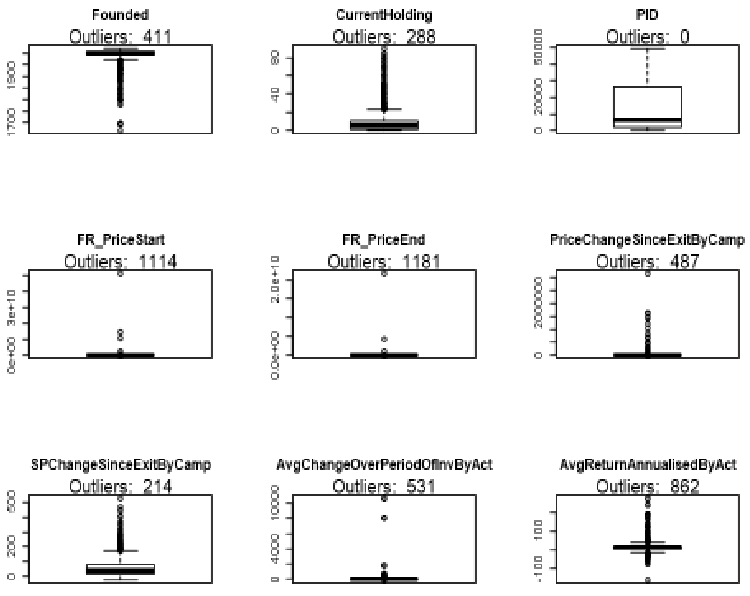
EDA – Data Cleaning - Outliers

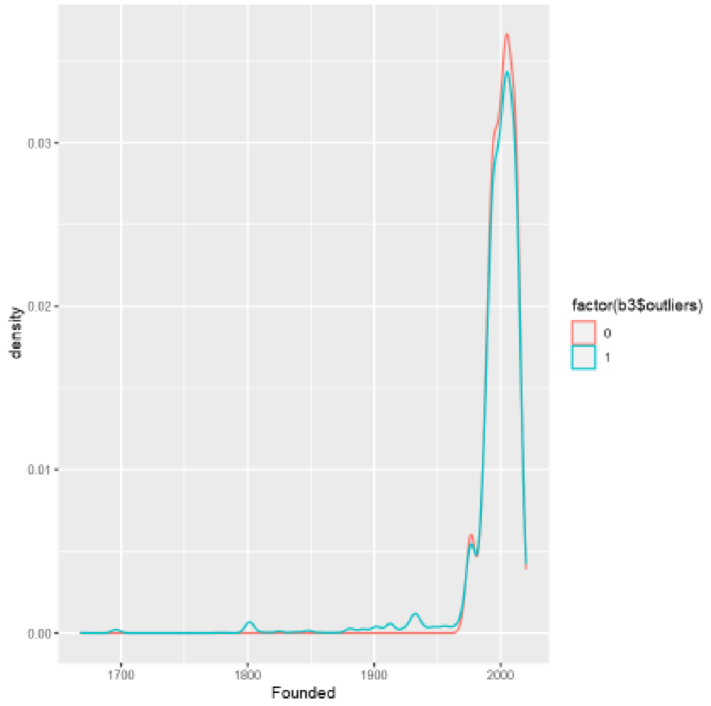
Detecting Outliers using outliers' matrix

Checking the distribution with(blue) / without outliers(red)

Correlation matrix - Y + X(with outliers) & X(without outliers)

Result DF





EDA – Data Cleaning - Outliers

Detecting Outliers using outliers' matrix

Checking the distribution with(blue) / without outliers(red)

Correlation matrix - Y + X(with outliers) & X(without outliers)

Result DF

	with	pv_w	without	pv_wo	diff	cor.drop	dis.drop	drop
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<fct>
Founded	0.02447008	3.397179e-02	-0.002885733	8.078509e-01	0.88207095	No	Yes	Drop
CurrentHolding	-0.20498188	9.742106e-34	-0.183220686	4.694388e-25	0.10616154	Yes	Yes	Drop
FR_PriceStart	0.08316077	2.335499e-15	0.107911859	5.052674e-22	-0.29762941	Yes	Yes	Drop
FR_PriceEnd	0.38664775	2.146221e-320	0.414456884	0.000000e+00	-0.07192369	Yes	Yes	Drop
PriceChangeSinceExitByCamp	0.02817042	6.829174e-02	0.113201991	4.438121e-12	-3.01847070	No	Yes	Drop
SPChangeSinceExitByCamp	0.08762148	2.183701e-07	0.077937526	7.503105e-06	0.11052034	Yes	Yes	Drop
AvgChangeOverPeriodOfInvByAct	0.09537587	8.421008e-15	0.060024133	2.834695e-06	0.37065707	No	No	Leave
AvgReturnAnnualisedByAct	0.25382696	1.808149e-97	0.194388531	4.861968e-50	0.23416910	Yes	No	Drop
AvgSPReturnByAct	-0.20973284	1.891091e-66	-0.195151021	6.636532e-56	0.06952569	Yes	No	Drop

EDA – Data Cleaning – Missing Values

- Summarizing missing values after outliers' treatment

```
1  getMissingness(df_noout)
```

$missingness

A data.frame: 116 × 3

var	na_count	rate
<fct>	<dbl>	<dbl>
AH_PruchaseValueMnLocal	8385	90.2
AvgAnnualisedReturnByMC	8329	89.6
AH_PricePerShareLocal	8220	88.4
Buyer	7866	84.6
ThirteenD	6845	73.6
SeatsProposed	6751	72.6
SeatsGained	6725	72.3
CurrentHolding	6151	66.1

DelayShareholderMeeting	7	0.1
CloseFund	7	0.1
ClosedAGM	7	0.1
ActivistIssuesPublicLetter	7	0.1
ActivistLetterToRegulatoryBodies	7	0.1
LitigationInitiated	7	0.1
ConsentSolicitationInitiated	7	0.1
SECFiling	7	0.1
AvgSPReturnByIND	2	0.0

$message

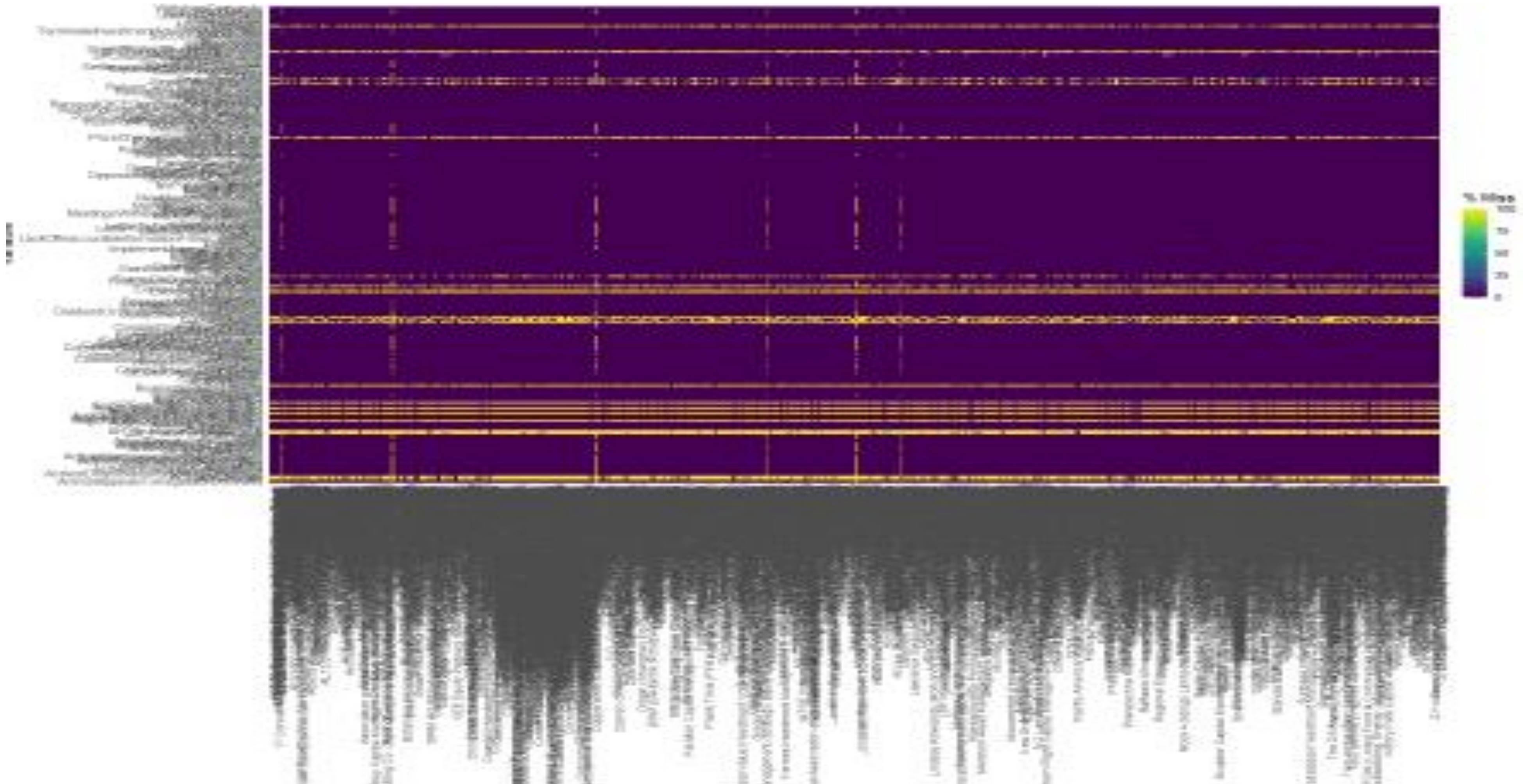
'This dataset has 0 (0%) complete rows. Original data has 9299 rows.'

$rows

NULL

EDA – Data Cleaning – Missing Values

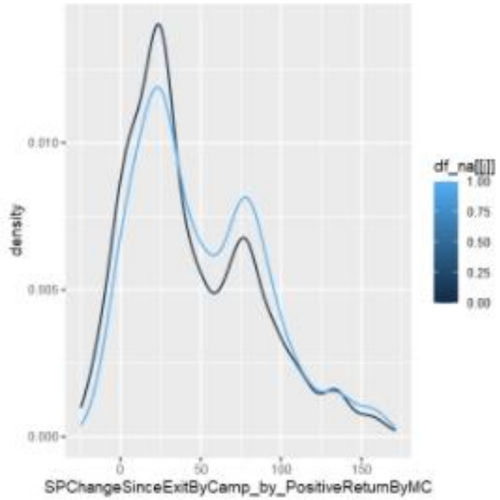
- Heatmap for missing values



EDA – Data Cleaning – Missing Values

- Determinate the Missingness Generation Mechanism

By the distribution and the t-test I assume that the missing mechanism is MNAR




```
[1] "SPChangeSinceExitByCamp by NA PositiveReturnByIND"

        Welch Two Sample t-test

data:  b1[[i]] and b2[[i]]
t = -0.4457, df = 1153.3, p-value = 0.6559
```

EDA – Data Cleaning – Missing Values

- Treatment:
  1. Quantile numeric variable and adding 'miss' category

```r
# quantile numeric variable
for (i in intVar){
   df.na[[i]] <-  as.factor(ifelse (is.na(df.na[[i]]),'miss',quantile(df.na[[i]], probs = seq(0, 1, 0.25), na.rm = TRUE,
}
getMissingness(df.na)
```

  2. For categorical vars with >10% missing – add 'miss'

```
   var  na_count    rate
 <fct>     <dbl>   <dbl>
$message
'This dataset has 9299 (100%) complete rows. Original data has 9299 rows.'
$rows
NULL
```

Re-Check for Outliers and missing values

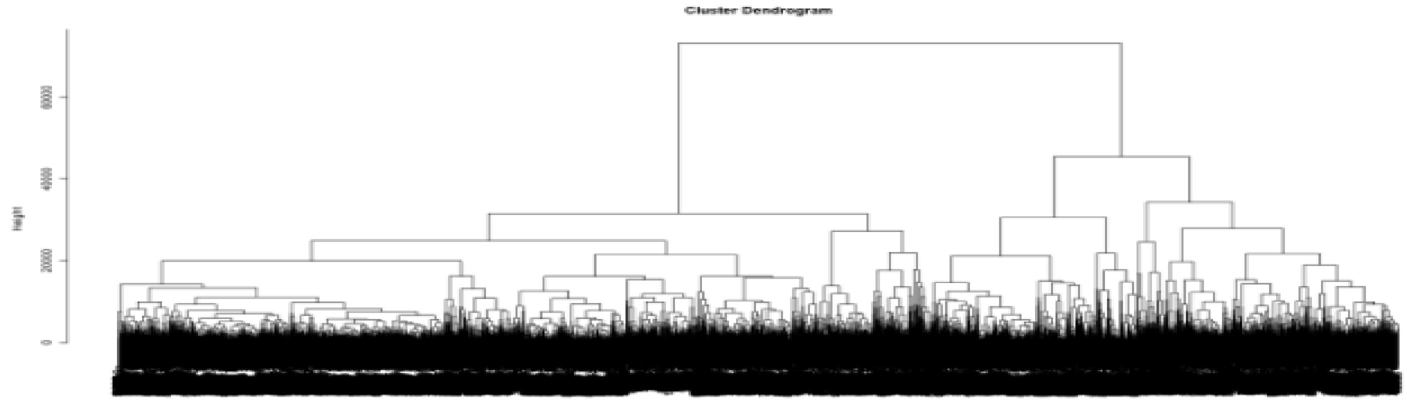
EDA – Notes and challenges

- A wrong ‘ifelse’ transcript converted big part of the data to NA.
- Outliers’ treatment

CLUSTER ANALYSIS

- Adding 2 type of clusters to the data, Threshold 3 and 4 :

1. Hierarchical clustering – hclust.



2. Gaussian mixture models – mclust.

```
library(mclust)

### McLust: implementation of gaussian mixture model

mcl_model3 <- Mclust(df, 3)

summary(mcl_model3)

mcl_model4 <- Mclust(df, 4)

summary(mcl_model4)
```

FEATURE SELECTION STRATEGY

- 2 methods for feature selection:
  1. Univariable Analysis – Selecting features with p_value≤0.05
  2. Multivariable Analysis – Using Lasso, Ridge, Naïve Bayse, Random forest, etc.
- Summarization and Selection of Variables – Threshold=1

```
varSel['Sum'] =  np.sum(varSel,axis=1)
varSel
```

	Variable	Univariable	Lasso	Ridge	ComplementNB	RandomForest	CART	GradientBoost	ADAboost	SVM	Sum
0	ActivistHQ_1	0	0	0	0	1	1	0	0	0	2
1	ActivistHQ_2	1	0	0	0	1	1	1	0	0	4
2	ActivistHQ_3	1	0	0	0	1	0	0	0	0	2
3	ActivistHQ_4	0	0	1	0	0	0	0	0	0	1
4	ActivistHQ_5	1	0	0	0	1	0	0	0	0	2
...	...	...	...	...	...	...	...	...	...	...	...
3280	Unresolved_2	1	0	0	0	1	1	0	0	0	3
3281	hclust3	1	0	0	0	1	1	0	0	0	3
3282	hclust4	1	0	0	0	1	1	0	0	1	4
3283	Mclust3	1	0	0	0	1	1	1	0	0	4
3284	Mclust4	1	0	0	0	1	1	1	0	1	5

3285 rows × 11 columns

FEATURE SELECTION – Notes and challenges

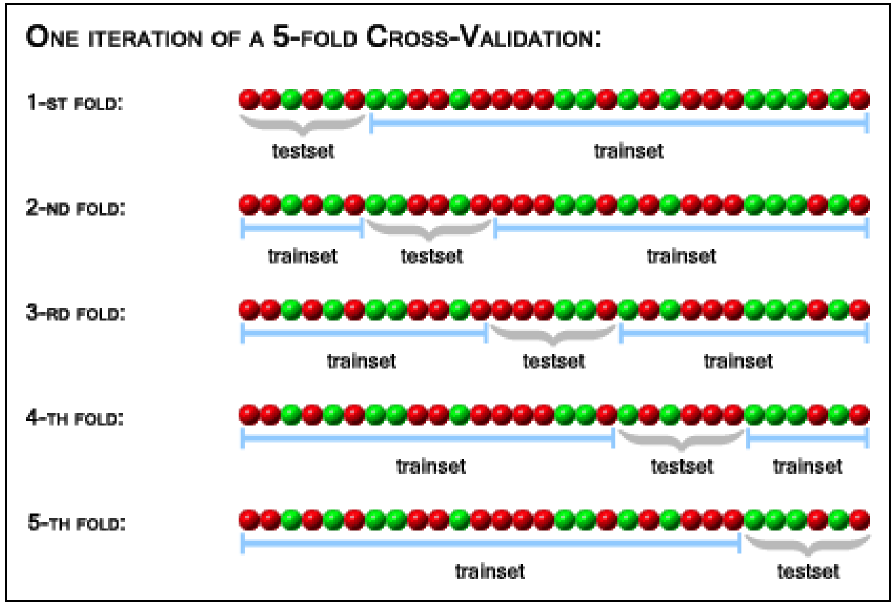
- Memory error due to high cardinality on ‘one hot encoding’.
- Using data batches for the encoder gave me different features which I couldn’t join back to DF

DATA PRE-PROCCING – Preparing the data

- Splitting data to train-dev-test

```
You got a perfectly balanced training and test datasets
```

- Using cross validation for Train-Dev



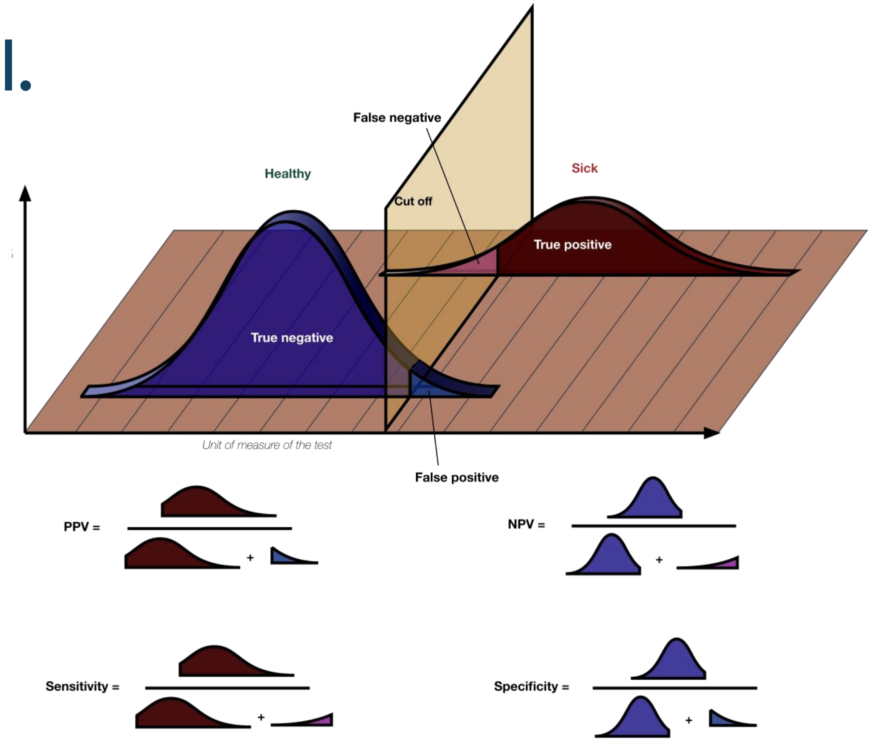
CLASSIFICATION MODELS – Metric Selection

- I chose 'Sensitivity' score with FPR – The Gain driven approach

The false positive rate is FPR= $\frac{FP}{FP+TN} = 1 - \text{Specificity}$

Because I wanted to make sure I won't miss any campaign that might be successful.

Sensitivity vs. Specificity



CLASSIFICATION MODELS - CV strategy Selection

- **StratifiedKFold** and **StratifiedShuffleSplit** for imbalance data

Category	#
0	6,979
1	2,320

Imbalance Ratio: 1/4

CLASSIFICATION MODELS - Results

- Model with the highest sensitivity score and the lowest FPR

Results

CLASSIFICATION MODELS

	Name	Model	AUC-train	AUC-test	AUC_diff	F1-train	F1-test	F1_diff	Sensitivity-train	Sensitivity-test	sens_diff	fpr-train	fpr-test	fpr_diff
2	SVC	mod3	0.847208	0.818621	0.028588	0.857845	0.857757	0.000087	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000
0	SVC	mod3	0.841789	0.812042	0.029747	0.857801	0.857845	0.000044	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000
1	SVC	mod3	0.845187	0.824536	0.020651	0.857801	0.857845	0.000044	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000
1	Gradient Boosting Classifier	mod8	0.897174	0.849202	0.047971	0.894374	0.879195	0.015180	0.953455	0.937724	0.015731	0.538877	0.589633	0.050756
2	Gradient Boosting Classifier	mod8	0.895010	0.851578	0.043432	0.891282	0.876722	0.014561	0.947745	0.934814	0.012931	0.539957	0.596112	0.056156
2	Logistic Regression	mod1	0.897094	0.848336	0.048758	0.898873	0.874362	0.024511	0.941661	0.919771	0.021890	0.463283	0.555076	0.091793
1	Logistic Regression	mod1	0.897233	0.844349	0.052885	0.896694	0.877349	0.019345	0.936985	0.919112	0.017873	0.461123	0.531317	0.070194
0	Gradient Boosting Classifier	mod8	0.898423	0.846752	0.051671	0.891411	0.869654	0.021757	0.941998	0.916965	0.025033	0.517279	0.578834	0.061555
0	Logistic Regression	mod1	0.898274	0.841868	0.056406	0.899092	0.873326	0.025766	0.939492	0.910523	0.028969	0.453564	0.526998	0.073434
1	ADAboost	mod6	0.871375	0.841471	0.029905	0.879447	0.867238	0.012209	0.911565	0.904796	0.006769	0.487041	0.548596	0.061555
2	Random Forest Classifier	mod5	0.999095	0.760225	0.238869	0.992342	0.852991	0.139351	0.997137	0.904011	0.093125	0.037797	0.650108	0.612311
1	XGboost	mod7	0.981824	0.833186	0.148637	0.956293	0.864902	0.091391	0.975295	0.900501	0.074794	0.194384	0.548596	0.354212
0	ADAboost	mod6	0.870145	0.826714	0.043431	0.880318	0.864436	0.015882	0.911207	0.896922	0.014285	0.479482	0.537797	0.058315
0	XGboost	mod7	0.985828	0.833010	0.152818	0.963237	0.868330	0.094907	0.980308	0.896922	0.083386	0.166307	0.509719	0.343413
2	ADAboost	mod6	0.871031	0.836379	0.034652	0.879227	0.866136	0.013091	0.911954	0.896848	0.015106	0.490281	0.524838	0.034557
1	Random Forest Classifier	mod5	0.999260	0.752747	0.246513	0.993022	0.849437	0.143585	0.993555	0.890480	0.103076	0.022678	0.622030	0.599352
0	Random Forest Classifier	mod5	0.999175	0.773250	0.225925	0.992123	0.849551	0.142572	0.992123	0.881174	0.110949	0.023758	0.583153	0.559395
2	XGboost	mod7	0.982256	0.834246	0.148009	0.960434	0.856745	0.103690	0.981747	0.880372	0.101374	0.188985	0.526998	0.338013
2	Decision Tree Classifier	mod4	0.999988	0.693839	0.306149	0.998566	0.853901	0.144666	0.997137	0.862464	0.134673	0.000000	0.475162	0.475162
0	Decision Tree Classifier	mod4	0.999984	0.678691	0.321294	0.998386	0.842781	0.155605	0.996778	0.846099	0.150679	0.000000	0.488121	0.488121
1	Decision Tree Classifier	mod4	0.999993	0.662455	0.337537	0.998925	0.832200	0.166725	0.998210	0.832498	0.165712	0.001080	0.507559	0.506479
1	ComplementNB	mod2	0.757936	0.681560	0.076376	0.789504	0.760886	0.028618	0.732546	0.712956	0.019589	0.371490	0.485961	0.114471
2	ComplementNB	mod2	0.759615	0.717470	0.042045	0.794563	0.766847	0.027715	0.742663	0.709169	0.033494	0.382289	0.423326	0.041037
0	ComplementNB	mod2	0.761277	0.695748	0.065529	0.796791	0.765163	0.031628	0.746867	0.704366	0.042501	0.385529	0.412527	0.026998

CLASSIFICATION MODELS – Model selected

Decision Tree Classifier is the best considering both parameters

	Name	Model	AUC-train	AUC-test	AUC_diff	F1-train	F1-test	F1_diff	Sensitivity-train	Sensitivity-test	sens_diff	fpr-train	fpr-test	fpr_diff
2	Decision Tree Classifier	mod4	0.999988	0.693839	0.306149	0.998566	0.853901	0.144666	0.997137	0.862464	0.134673	0.000000	0.475162	0.475162
0	Decision Tree Classifier	mod4	0.999984	0.678691	0.321294	0.998386	0.842781	0.155605	0.996778	0.846099	0.150679	0.000000	0.488121	0.488121
1	Decision Tree Classifier	mod4	0.999993	0.662455	0.337537	0.998925	0.832200	0.166725	0.998210	0.832498	0.165712	0.001080	0.507559	0.506479

CLASSIFICATION MODELS – Fine Tuning

- Random search and Grid Search for hyperparameter optimization

Random search

```
# The function to measure the quality of a split
criterion = ['gini', 'entropy']
# The strategy used to choose the split at each node.
# Supported strategies are “best” to choose the best split and “random” to choose the best random split
splitter = ['best', 'random']
# The minimum number of samples required to split an internal node
min_samples_split = [2,4,8,16,32,40]
# The minimum number of samples required to be at a leaf node
min_samples_leaf = [2,4,6,8,10,12,14,16,18,20]

random_grid = {'criterion': criterion,
               'splitter': splitter,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'min_samples_split': [2, 4, 8, 16, 32, 40], 'min_samples_lea
f': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]}
```

```
DTC_random.best_params_
```

```
{'splitter': 'best',
 'min_samples_split': 40,
 'min_samples_leaf': 18,
 'criterion': 'gini'}
```

CLASSIFICATION MODELS – Fine Tuning

- Grid search - We decide which parameters and how (not randomly)

```
# Create the parameter grid
param_grid = {'splitter': ['best'],
              'min_samples_split': [35,40,45],
              'min_samples_leaf': [17,18,19],
              'criterion': ['gini']
}

grid_search = GridSearchCV(estimator = DTC, param_grid = param_grid, cv = sss,
                           verbose=2, n_jobs=-1)
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'min_samples_leaf': 18,
 'min_samples_split': 40,
 'splitter': 'best'}
```

	Base Score	Random Score	Grid Score
0	0.847222	0.891762	0.966696
1	0.875000	0.902778	0.961368
2	0.863027	0.867816	0.962256

We can see the improvement between base, random and grid fine tuning

PREDICTIONS

```
# make a prediction
y_pred = best_grid.predict(X_test)

```

```
pd.crosstab(y_test,y_pred)
```

```
C:\Users\Mayer\Anaconda3\lib\site-packag
tegorical object instead of an ndarray i
  vec = libmissing.isnaobj_old(values.ra
```

col_0	0	1
Succession		
0	754	635
1	322	3868

```
yprob = best_grid.predict_proba(X_test)
yprob = pd.DataFrame(yprob)
yprob
```

	0	1
0	0.289474	0.710526
1	0.090909	0.909091
2	0.000000	1.000000
3	0.000000	1.000000
4	0.000000	1.000000
...	...	...
5574	0.000000	1.000000
5575	0.217391	0.782609
5576	0.000000	1.000000
5577	0.916667	0.083333

FUTURE DEVELOPMENT

- Training day0 set (features from 13D)
- Selecting model by a conservative approach (Specificity)
- Production

THANK YOU