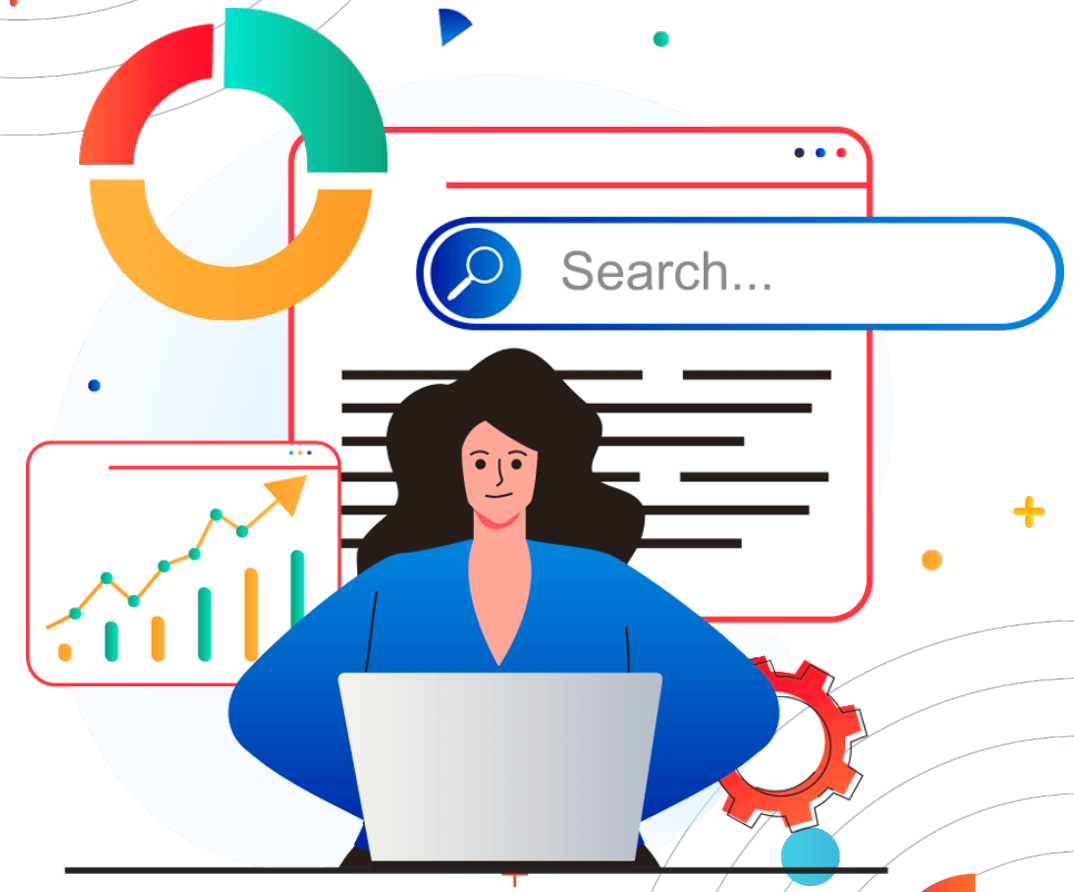
# Data Analysis For Shark Tank India Pitches

## Using Python, R, Excel and Tableau

**By: Shipra Mehandru**




**IITD BAO'21 Course – Feb 2022**

# What the Project Covers

## *Dataset understanding and Preprocessing on Python and Excel*

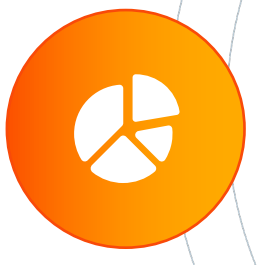
- I have **eliminated the non correlated** variables using Excel regression runs
- Also **standardized the variables** before applying any ML model

## *Predicting classes using ML models on Python*

- All the **Supervised learning ML Classification** runs used have been showed in detail along with an explanation for why it was done
- And **links to output notebooks** are included on every page instead of copy pasting all screen shots

## *Bettering analysis through R Studio, Clustering*

- In order to see **improved prediction accuracies** I have analyzed 2 more datasets, which includes:
- A **bigger sample**;
- A better **correlation** analysis through **R Studio**;
- Finding better clusters through **unsupervised K Means**;
- While avoiding the overfit model trap



## *Data Visualizations using Tableau & Python*

- The deck also shows **charts for every run on Python** and R Studio using inbuilt visualization libraries
- A **Tableau Dashboard** has been added as a screen shot at the end, which can be access on the Public Tableau link. It analysis data further

# Understanding the Dataset

# The dataset is from Kaggle:

**Where a user compiled 121 pitches from Season 1 using Wikipedia**

kaggle [Link]

## This is the Description of Variables and the Meta Data of the File:

1. Episode_number - Number of the episode
2. Pitch_number - Number of the Pitch
3. Brand_name - Name of the brand Idea
4. Idea - behind the brand building Deal
5. Deal - done or not ; 1 - YES, 0 - NO
6. Pitcheraskamount - Amount asked by the pitchers
7. Ask_equity - Equity offered by the pitchers
8. Ask_valuation - Valuation asked by pitchers
9. Deal_amount - Final Deal Amount
10. Deal_equity - Final Deal equity percentage
11. Deal_valuation - Final Valuation of Company after Deal

**+ 17 other Data points on the Investors or Sharks who were present and who invested in the pitches**

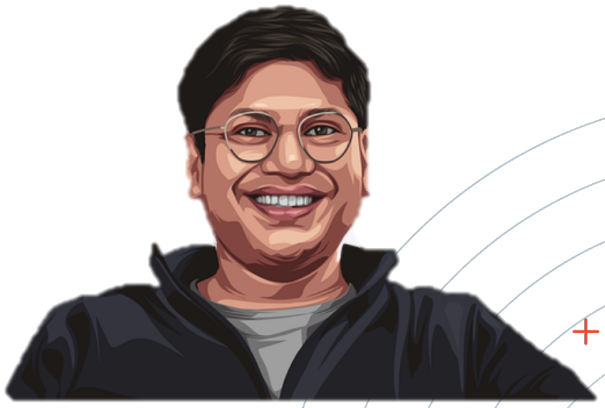
**Ghazal Alagh**
**Co-Founder & CIO of Mamaearth**

**Anupam Mittal**
**Founder & CEO of People Group**

**Namita Thapar**
**Executive Director at Emcure Pharma**

**Vineeta Singh**
**CEO & Co-Founder of SUGAR Cosmetics**

**Aman Gupta**
**Co-Founder & CMO of boAt**



**Peyush Bansal**
**Founder & CEO of Lenskart**

**Ashneer Grover**
**Co-Founder & MD of BharatPe**

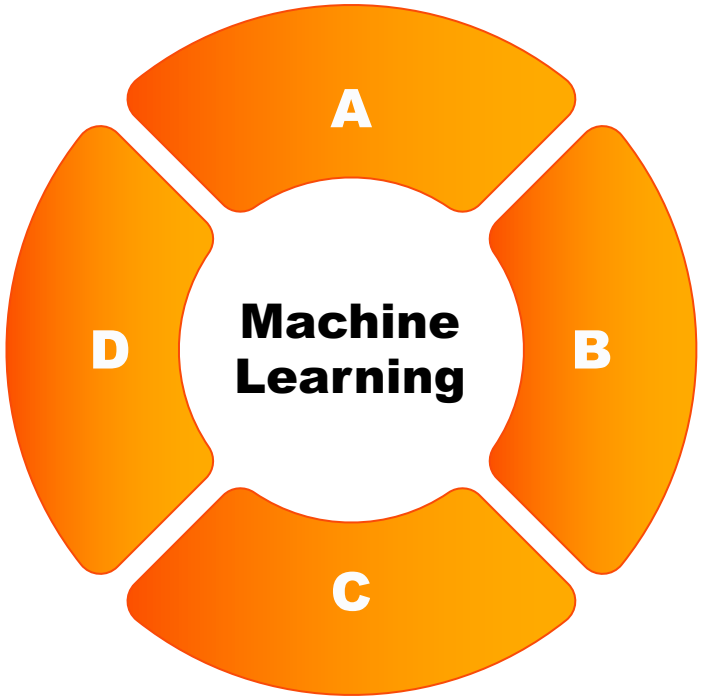
# Understanding Classifiers

# Machine Learning has a lot of techniques that can be used to analyze a dataset

## Classification

*(For Supervised Learning where the output variable is known)*

Logistic Regression
K-Nearest Neighbors
Support Vector
Naïve Bayes
Decision Tree
Random Forest




## Regression

Linear and polynomial
Support Vector (with/without Kernel)
Decision Tree
Random Forest

*(For Continuous values, where there are no categorical output values)*

## Deep Learning

*(Which learns from a reward mechanism and improves itself through hidden layers)*

Artificial Neural Networks
Convolutional/Recurrent Neural
Reinforcement Neural

+ Dimension Reduction Analysis
Big Data techniques
NLP and other

## Clustering

K-Means
Hierarchical

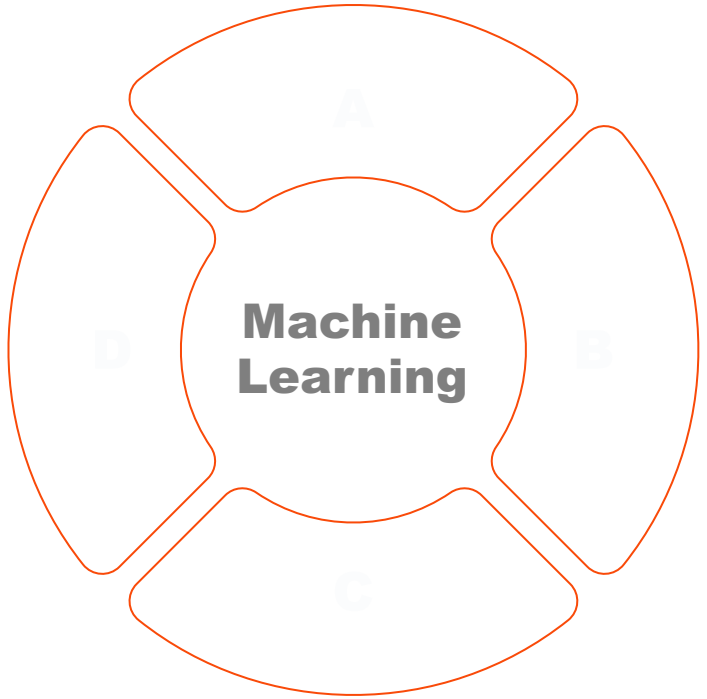
*(For Unsupervised Learning where w don not know the output variable, it is not labeled as yet)*

# I will be using the following for the purposes of analyzing the Shark Tank data:

## Classification

- Logistic Regression
- K-Nearest Neighbors
- Support Vector
- Naïve Bayes
- Decision Tree
- Random Forest

To predict Deal or No Deal outcomes i.e. Categorical Output




## Regression

- Linear
- Polynomial
- Support Vector (with/without Kernel)
- Decision Tree
- Random Forest

To understand how much a variable is influencing the outcome

## Deep Learning

Artificial Neural Networks
Convolutional/Recurrent Neural
Reinforcement Neural

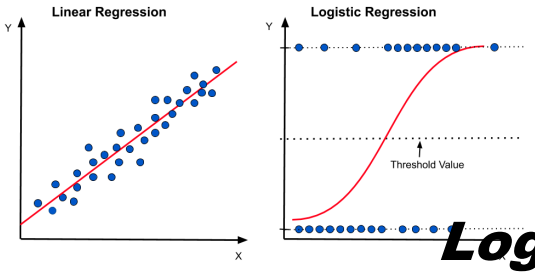
## Clustering

- K-Means
- Hierarchical

To understand if there can be better categorical outcomes for the same data

+ Dimension Reduction Analysis
Big Data techniques
NLP and other

# What each of these Supervised Classification models show:

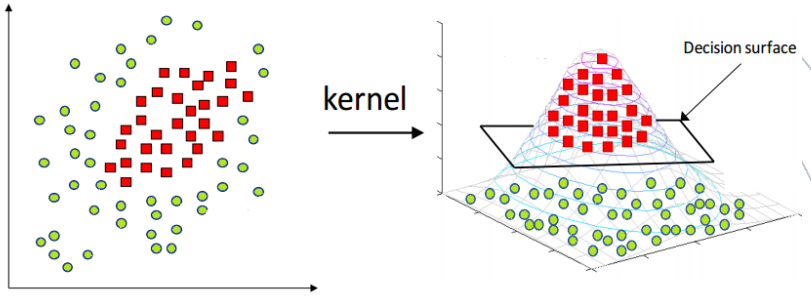


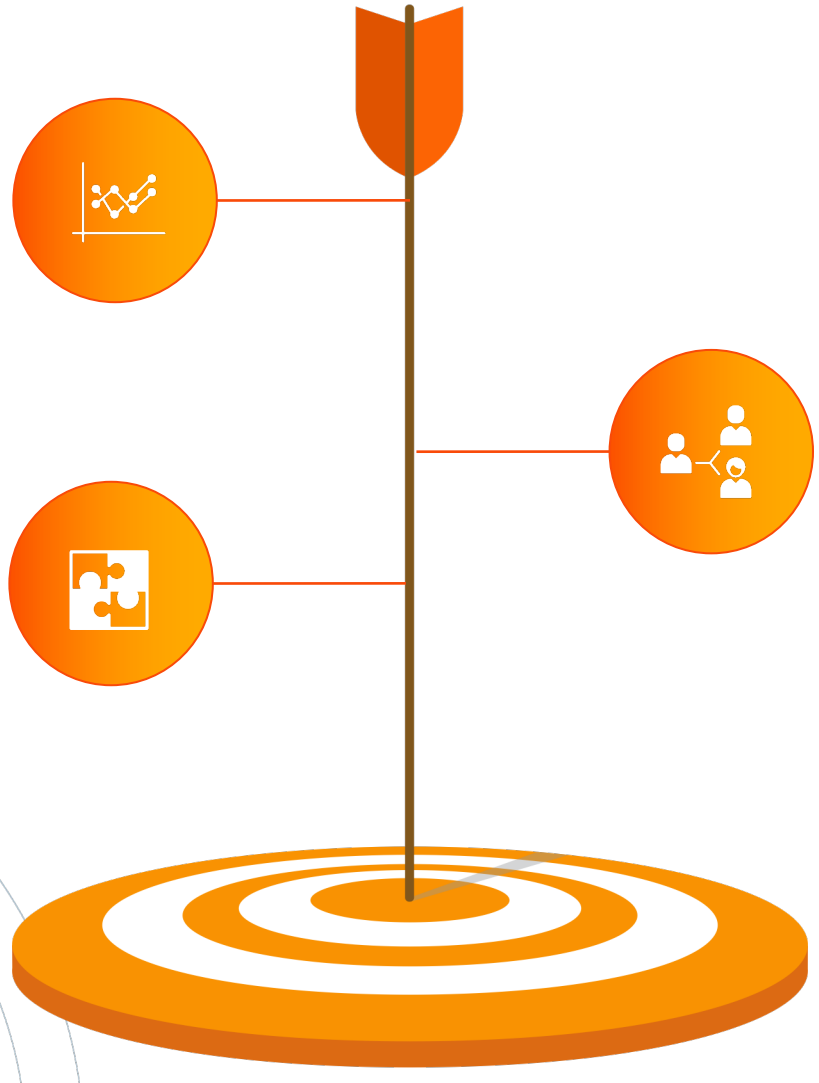

## Logistic Reg.

*Based on the concept of probability - from 0 to 1, of a datapoint falling into each class instead of Linear where the points can fall anywhere along a linear line*

## Kernel SVM

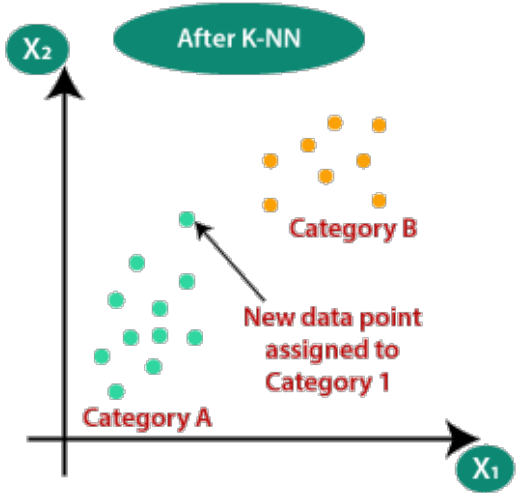
*It either creates a hyperplane where the classes exist separately – in a Linear SVM or for Kernal method (RBF) / non linear data it separates via area on a higher plane*



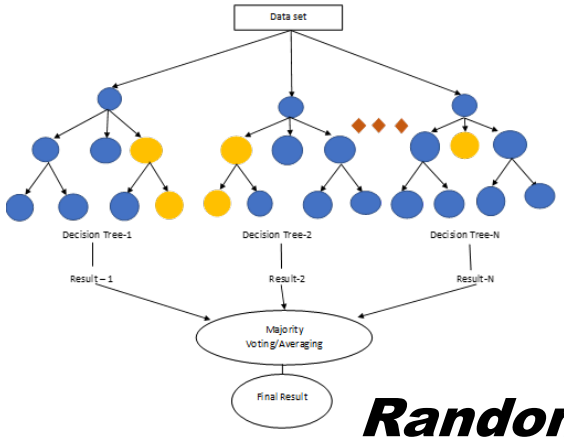



## K-NN

*Defines a dataset's class by finding their distances from the chosen variables. A point should be nearer to it's neighbors and farther from other classes*

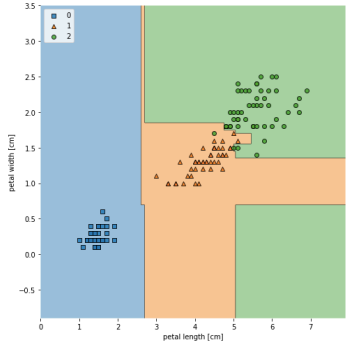


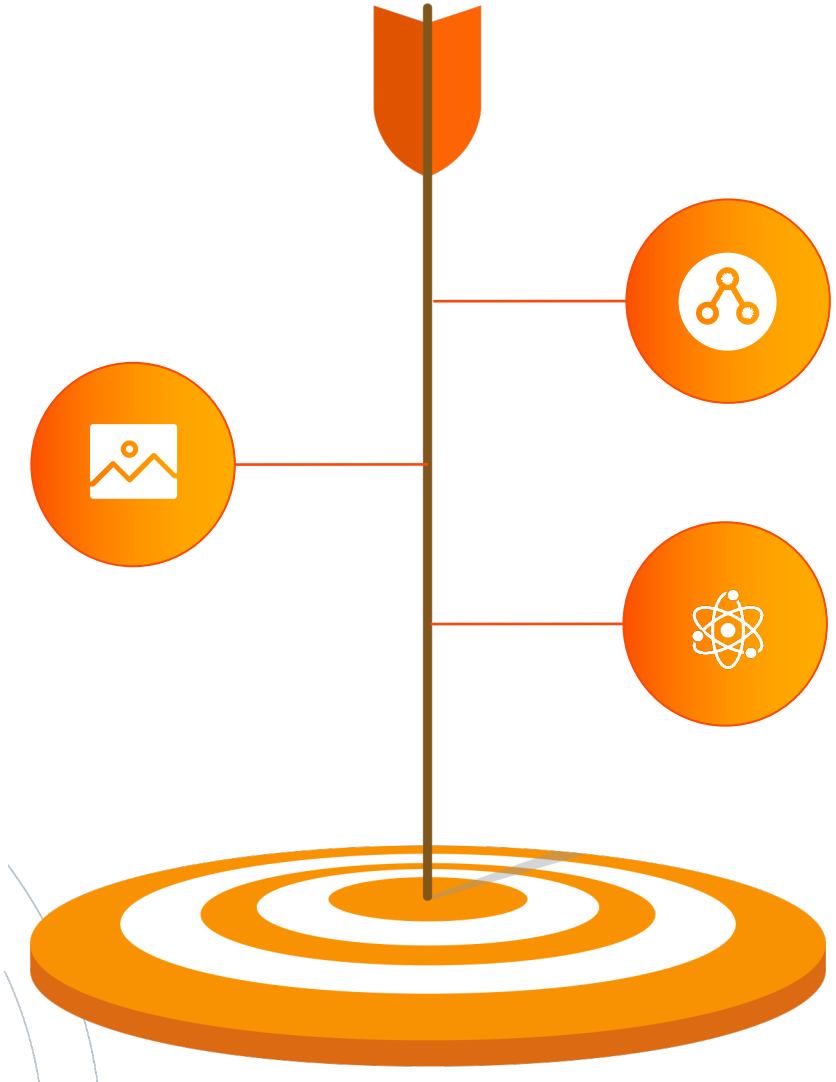
# What each of these Supervised Classification models show:

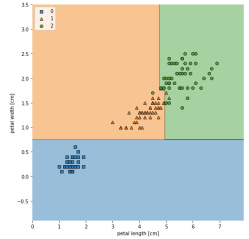


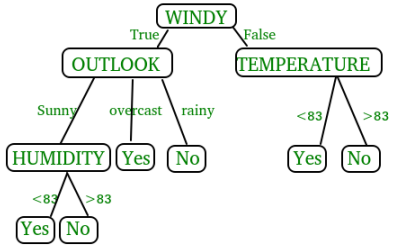

## Random Forest

*As the name suggests it's a cluster of different decision trees and building a strong classifier using weaker classifiers*








## Decision Tree

*Based on a series of events occurring, we go from the node that has the highest certainty and keep branching low certainty nodes, variables*

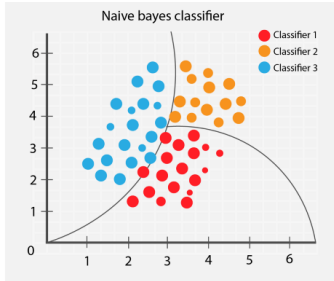
## Naïve Bayes

*It is also based on probability of a datapoint being in one class given the probability that other have been distributed to it's neighboring classes. I.e. event A occurring while B has already occurred*

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

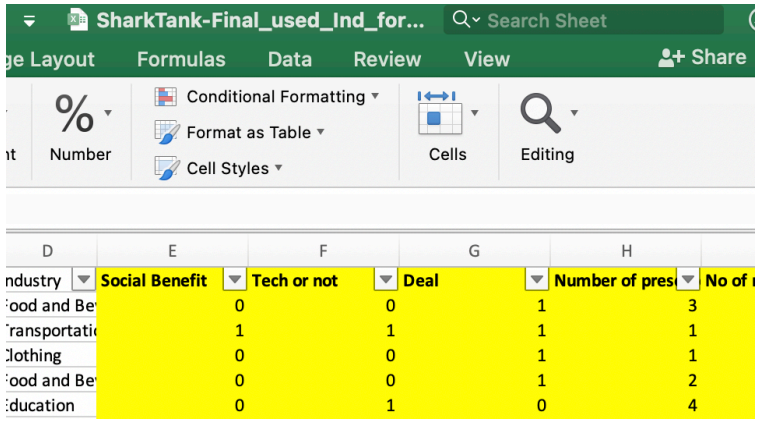
using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

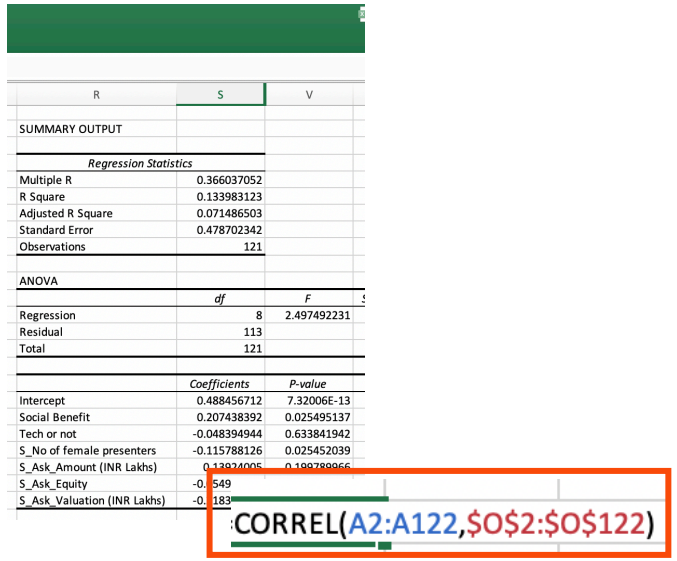


***Before we get into the Classifiers, let us see the pertinent variables to run ML models by using Regression & Correlation in Excel***

# I did a bit of data preprocessing in the excel file itself:




| D | E | F | G | H |
|---|---|---|---|---|
| ndustry | Social Benefit | Tech or not | Deal | Number of pres | No of |
| ood and Be | 0 | 0 | 1 | 3 | |
| ransportati | 1 | 1 | 1 | 1 | |
| lothing | 0 | 0 | 1 | 1 | |
| ood and Be | 0 | 0 | 1 | 2 | |
| ducation | 0 | 1 | 0 | 4 | |



SUMMARY OUTPUT

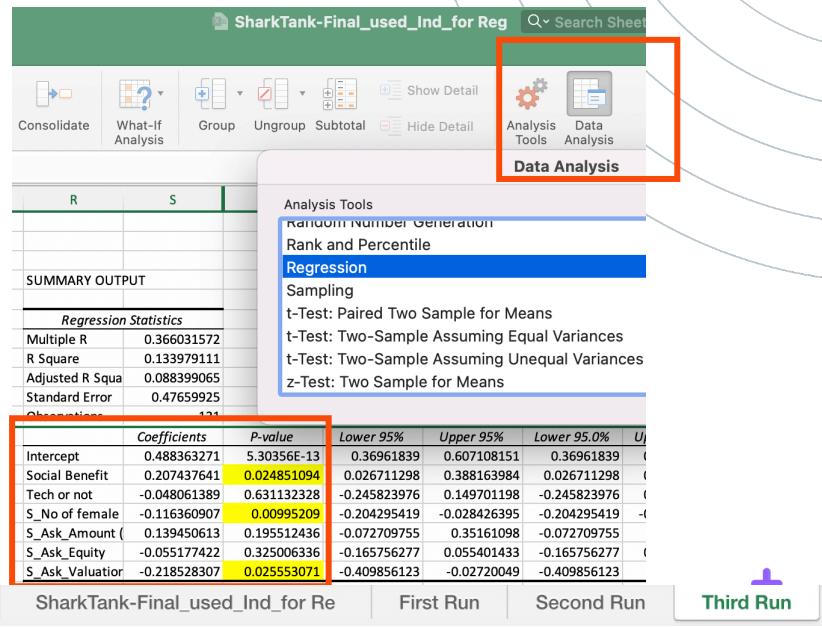
| *Regression Statistics* | |
|---|---|
| Multiple R | 0.366037052 |
| R Square | 0.133983123 |
| Adjusted R Square | 0.071486503 |
| Standard Error | 0.478702342 |
| Observations | 121 |

ANOVA

| | *df* | *F* |
|---|---|---|
| Regression | 8 | 2.497492231 |
| Residual | 113 | |
| Total | 121 | |

| | *Coefficients* | *P-value* |
|---|---|---|
| Intercept | 0.488456712 | 7.32006E-13 |
| Social Benefit | 0.207438392 | 0.025495137 |
| Tech or not | -0.048394944 | 0.633841942 |
| S_No of female presenters | -0.115788126 | 0.025452039 |
| S_Ask_Amount (INR Lakhs) | 0.13924005 | 0.199789866 |
| S_Ask_Equity | -0.549 | |
| S_Ask_Valuation (INR Lakhs) | -0.183 | |

=CORREL(A2:A122,$O$2:$O$122)




From the pitches and industries, I **categorized** the pitches as offering some **Social Benefit** or not

Chose pitchers' requested **amount, equity, valuation, number and gender of pitchers & tech vs not tech** pitches as predicting Deal or no Deal in the **first Regression Run**

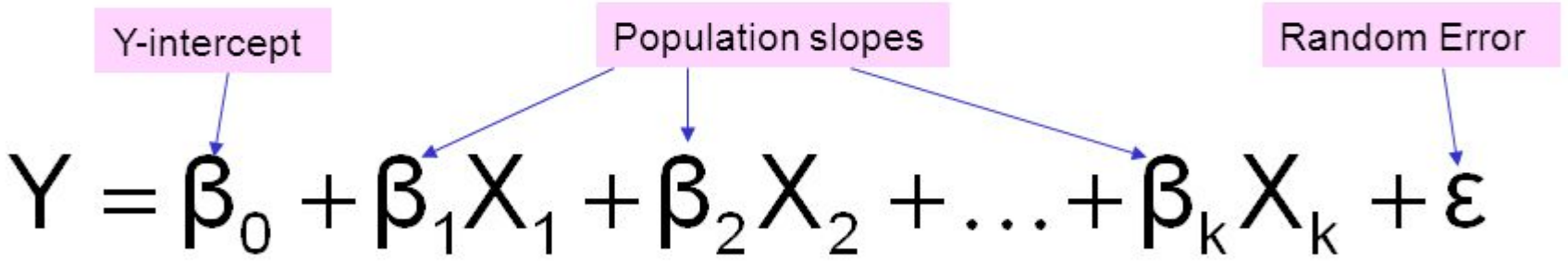
Did **3 rounds of regressions** and correlations, eliminating the variables with a high P value. Tool the ones with a high correlation **and low P value (<0.5) to as the most influential variables** for the ML predictor models

**+ I used the STANDARDISE Formula with Mean and Standard Deviation to standardize all values before the regression runs**

# What is a Regression Equation and hence how are Coefficients of variables useful?

It helps examine each of the **independent** variables' **linear relationship** with the dependent variable

**Multiple Regression Model with k Independent Variables:**




Whereas a Correlation model **directly gives us the relation** of one array of values in a variable to those of another variable. That is also a Linear relationship measure, **though not causal**

# Variables Chosen for the ML Predictions of Deal or No Deal:

I've chosen **Social Benefit** which has the strongest positive correlation and **Asked for valuation** which has the strongest negative correlation

Interestingly most marketing data I've worked with usually has a low R Square and a high Error We see the same thing here with **low regression coefficients and correlation scores**

**Tech or not Tech Pitch**

Correlation = 0.01
Reg. Coeff. = -0.04

**Pitchers willing to give % of Equity**

Correlation = -0.06
Reg. Coeff. = -0.05

**No of Female Pitchers in team**

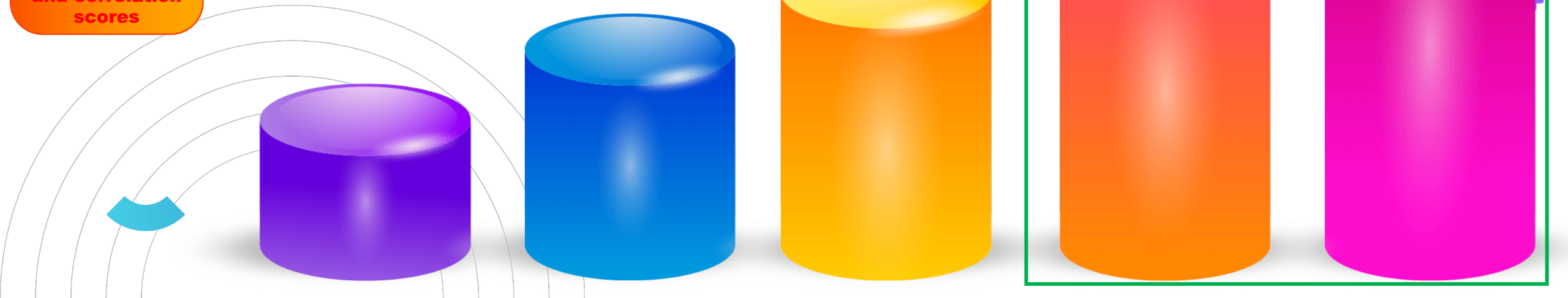
Correlation = -0.2
Reg. Coeff. = -0.11

**Social Benefit of the Pitch product/ service**
(Eg: environment, health, city)

Correlation = 0.16
Reg. Coeff. = 0.2

**Asked valuation by the pitchers for their product/ service**

Correlation = -0.18
Reg. Coeff. = -0.21



# Predicting Deal or No Deal for 25% of the pitches X_test and y_test + Data Preprocessing Using Python

# Showing the Complete ML Run for Logistic Regression

Link – to All classifiers
python notebook


All Classifers_Shark Tank India.ipynb

File Edit View Insert Runtime Tools Help All ch

+ Code + Text

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# method for uploading data from local
from google.colab import files
uploaded = files.upload()
```

```
# Importing the dataset
import io
dataset = pd.read_csv(io.BytesIO(uploaded['SharkTank-Final_used_Ind.csv']))
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 36 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Episode                   121 non-null    int64
 1   Company                   1 non-null      object
 2   Ide
 3   Inc
 4   Soc
 5   Tec
```

Used a Google Colab environment for this, as my organization does not allow other notebooks.
All Python code can be found on my **GitHub page – IITD-BAO-21-Notebooks**

```
# discontinuous columns https://stackoverflow.com/questions/50
X = dataset.iloc[:, [4,12]].values
y = dataset.iloc[:, 6].values
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
```

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
```

```
[[ 2 15]
 [ 1 13]]
```

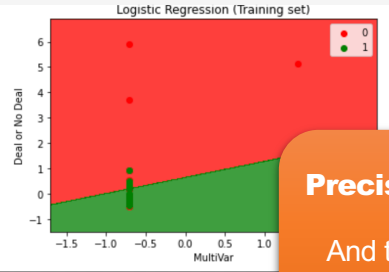
Setting Social Benefit, Ask Valuation as X Array and Deal as Y Array through **iloc function**
Using the **StandardScaler** to transform all comuns as Social benefit, deal and Asked Valuations are all in different ranges
And fitting the **Logistic regression** model for y-pred

# Splitting the data into Train and Test and fitting different ML models

```
#Performance metrics
# Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy is %f' % accuracy)

# Precision
from sklearn.metrics import precision_score
precision = precision_score(y_test, y_pred)
print('Precision is %f' % precision)
```

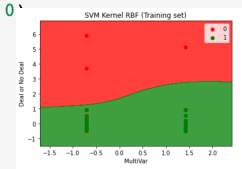
```
Accuracy is 0.483871
Precision is 0.464286
```

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('MultiVar')
plt.ylabel('Deal or No Deal')
plt.legend()
plt.show()
```

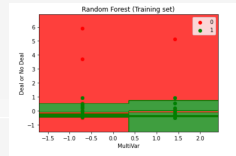



**Precision** a measure of how the model is performing is .48
And the **Matplotlib** chart shows the errors the model has made in predicting False positives, i.e. in this case Deal = 1 instead of No Deal = 0

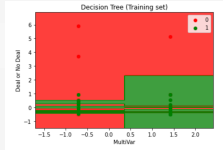

```
# Kernel SVM

# Fitting Kernel SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```



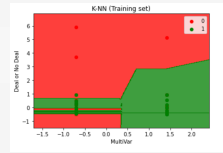

```
# Random Forest Classification
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```



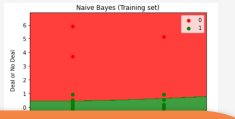

```
# Decision Tree Classification

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```




```
# K-Nearest Neighbors (K-NN)

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

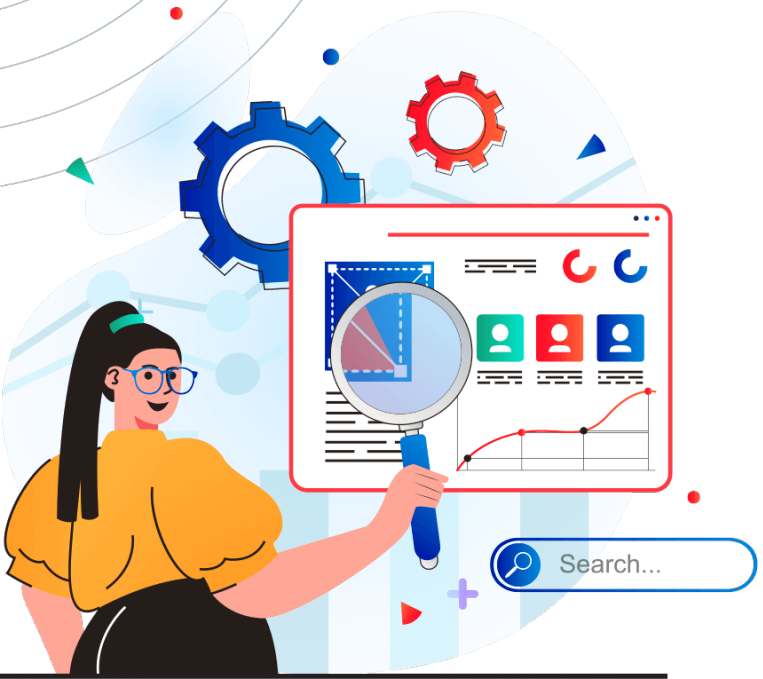



```
# Naive Bayes

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_trai
```

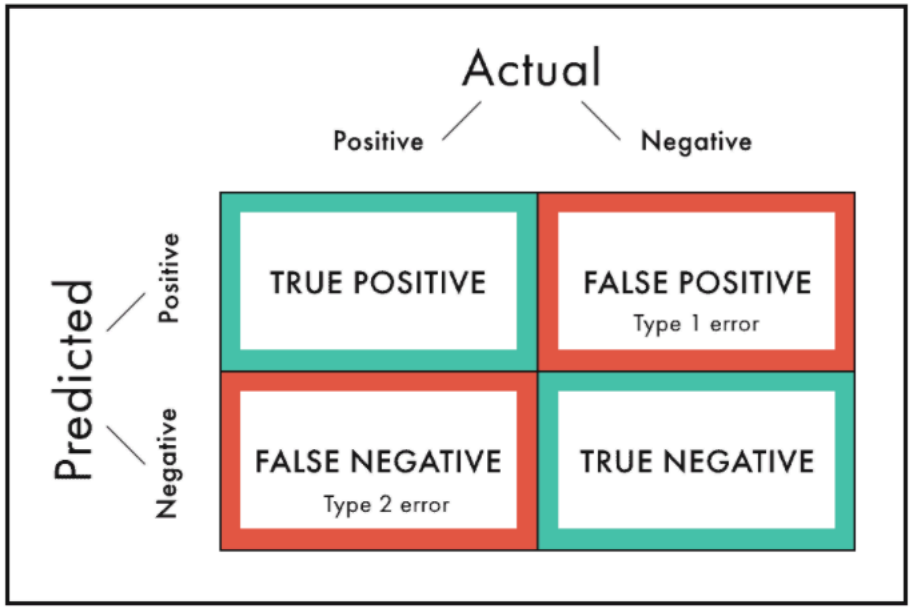



Repeating the fitting, predicting, accuracy and plotting commands **for SVM Kernel, Random Forest, Decision Tree, K-Nearest Neighbors and Naïve Bayes**

# What does the Confusion Matrix show:




The Confusion Matrix




**This is a way of checking how are our y_predicts compared to the y_test data outputs**

These are then used to check the :
**Accuracy=** (True predictions)/(All Predictions)
**Precision**= True Positives/(False Positives + True Positives)

# Accuracies that I got through Different Models Shark Tank India data

Confusion matrices

```
print(cm)
```

```
[[ 2 15]
 [ 1 13]]
```

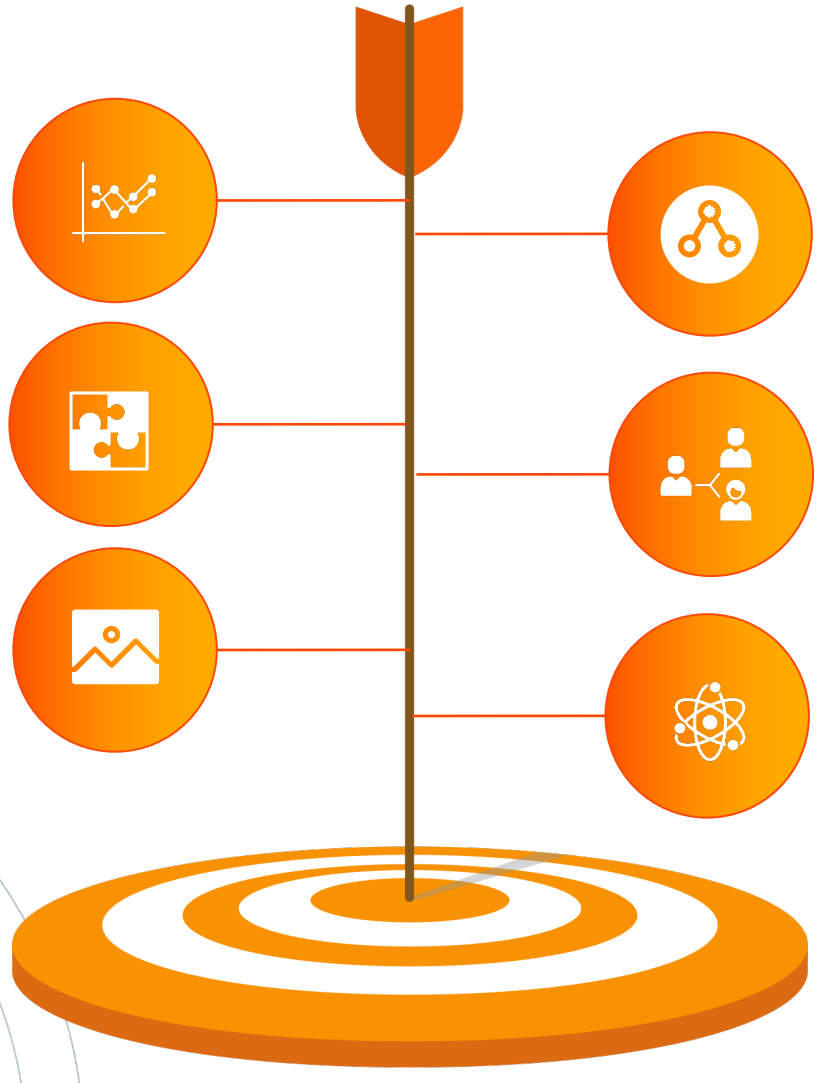
## Logistic
***Accuracy: 0.48***
***Precision 0.46***

```
[[ 1 16]
 [ 0 14]]
```

## Kernel SVM
***Accuracy: 0.48***
***Precision 0.47***

```
[[ 6 11]
 [ 4 10]]
```

## Random Forest
***Accuracy: 0.52***
***Precision 0.48***



Confusion matrices

## Decision Tree
***Accuracy: 0.45***
***Precision 0.43***

```
[[ 5 12]
 [ 5  9]]
```

## K-NN
***Accuracy: 0.54***
***Precision 0.50***

```
[[ 7 10]
 [ 4 10]]
```

## Naïve Bayes
***Accuracy: 0.54***
***Precision 0.50***

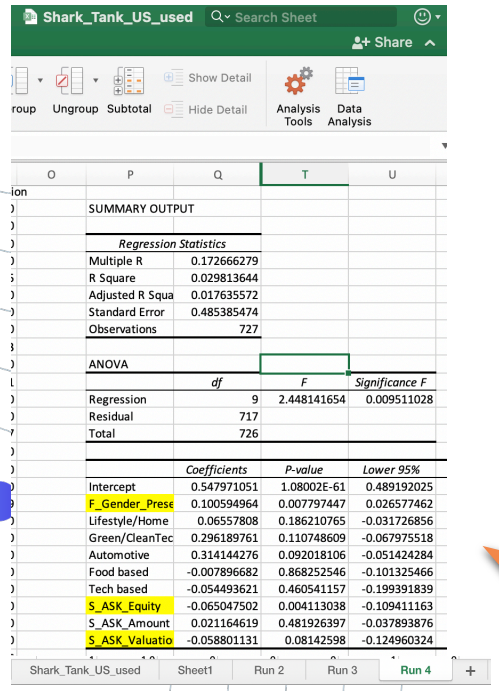
```
[[ 3 14]
 [ 0 14]]
```

***Given that the accuracies are at 50%, should we assume that the more data we compile from further seasons, the better prediction we can make?***

# Testing our assumption by using Season 5-11, 2014-2021 data of the US Shark Tank

kaggle [Link](#) to dataset

[Link](#) to Python Run on GitHub



Shark_Tank_US_used

SUMMARY OUTPUT

| Regression Statistics | |
|---|---|
| Multiple R | 0.172666279 |
| R Square | 0.029813644 |
| Adjusted R Squa | 0.017635572 |
| Standard Error | 0.485385474 |
| Observations | 727 |

ANOVA

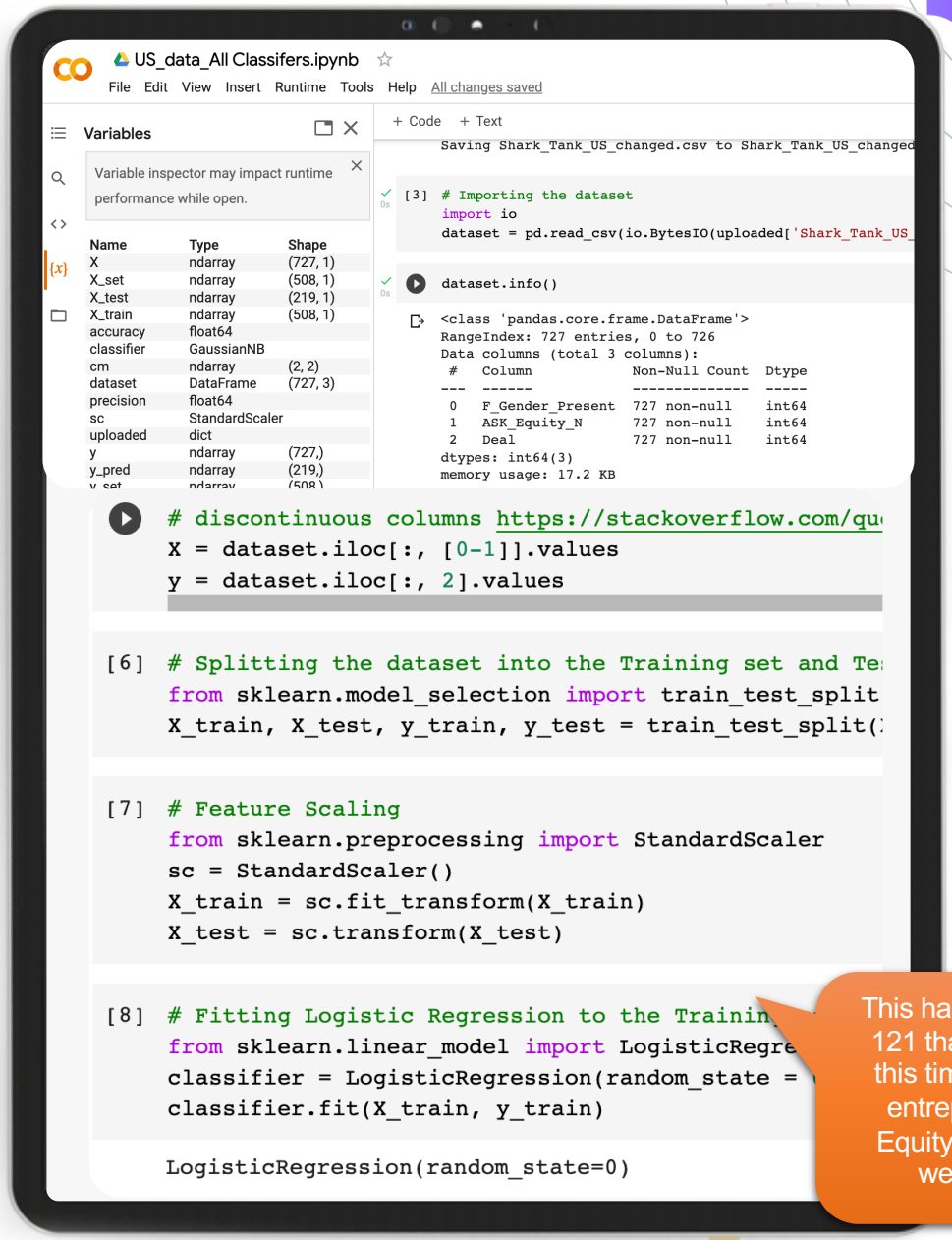
| | df | F | Significance F |
|---|---|---|---|
| Regression | 9 | 2.448141654 | 0.009511028 |
| Residual | 717 | | |
| Total | 726 | | |

| | Coefficients | P-value | Lower 95% |
|---|---|---|---|
| Intercept | 0.547971051 | 1.08002E-61 | 0.489192025 |
| F_Gender_Prese | 0.100594964 | 0.007797447 | 0.026577462 |
| Lifestyle/Home | 0.06557808 | 0.186210765 | -0.031726856 |
| Green/CleanTec | 0.296189761 | 0.110748609 | -0.067975518 |
| Automotive | 0.314144276 | 0.092018106 | -0.051424284 |
| Food based | -0.007896682 | 0.868252546 | -0.101325466 |
| Tech based | -0.054493621 | 0.460541157 | -0.199391839 |
| S_ASK_Equity | -0.065047502 | 0.004113038 | -0.109411163 |
| S_ASK_Amount | 0.021164619 | 0.481926397 | -0.037893876 |
| S_ASK_Valuatio | -0.058801131 | 0.08142598 | -0.124960324 |

Shark_Tank_US_used | Sheet1 | Run 2 | Run 3 | Run 4

**Regression & Correlation runs have been added to the project submission folder**

Variables for the ML model chosen bases the regression and correlation coefficients again – by multiple rounds of P value based eliminations



US_data_All Classifers.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Variables

Variable inspector may impact runtime performance while open.

| Name | Type | Shape |
|---|---|---|
| X | ndarray | (727, 1) |
| X_set | ndarray | (508, 1) |
| X_test | ndarray | (219, 1) |
| X_train | ndarray | (508, 1) |
| accuracy | float64 | |
| classifier | GaussianNB | |
| cm | ndarray | (2, 2) |
| dataset | DataFrame | (727, 3) |
| precision | float64 | |
| sc | StandardScaler | |
| uploaded | dict | |
| y | ndarray | (727,) |
| y_pred | ndarray | (219,) |

+ Code + Text

```
Saving Shark_Tank_US_changed.csv to Shark_Tank_US_changed
```

```
[3] # Importing the dataset
import io
dataset = pd.read_csv(io.BytesIO(uploaded['Shark_Tank_US_
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 727 entries, 0 to 726
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   F_Gender_Present  727 non-null    int64
 1   ASK_Equity_N      727 non-null    int64
 2   Deal              727 non-null    int64
dtypes: int64(3)
memory usage: 17.2 KB
```

```
# discontinuous columns https://stackoverflow.com/qu
X = dataset.iloc[:, [0-1]].values
y = dataset.iloc[:, 2].values
```

```
[6] # Splitting the dataset into the Training set and Te
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
```

```
[7] # Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
[8] # Fitting Logistic Regression to the Trainin
from sklearn.linear_model import LogisticRegre
classifier = LogisticRegression(random_state =
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

This has 727 cases up from the 121 that we had for India, and this time we used the Female entrepreneurs data and the Equity stake that the pitchers were willing to give up

# Accuracies that I got through Different Models for the US Shark Tank data

Confusion matrices

```
[11] print(cm)

[[ 81   0]
 [  0 138]]
```

## Logistic
***Accuracy: 1.00***
***Precision 1.00***

```
[[ 81   0]
 [  0 138]]
```

## SVM
***Accuracy: 1.00***
***Precision 1.00***

```
[[ 81   0]
 [  0 138]]
```

## Random Forest
***Accuracy: 1.00***
***Precision 1.00***

Confusion matrices

## K-NN
***Accuracy: 1.00***
***Precision 1.00***

```
[[ 81   0]
 [  0 138]]
```

## Naïve Bayes
***Accuracy: 1.00***
***Precision 1.00***

```
[[ 81   0]
 [  0 138]]
```

By this time and with the dataset that we already have for US Shark Tank – the prediction of all predictions is 100% accurate – specially when taken on the simplistic terms of Equity offered and number of Female Pitchers

***However, I discovered that for some datasets if one of the prediction classes is in the ratio of 1:9 with the other class – the prediction model predicts all Negative predicts – resulting in high accuracies.***
***To solve for this we need to weight down the dominant class:***

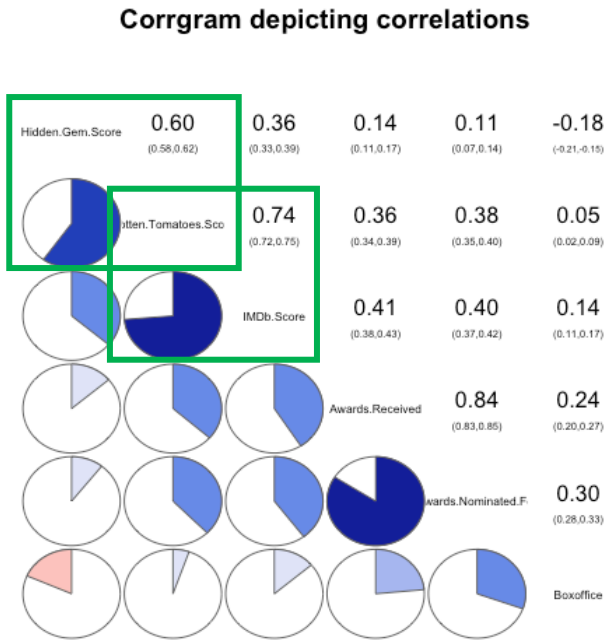
**Also Using R Studio instead of Excel to check Correlations among all variables**

# *Testing this assumption by using a comprehensive Netflix database to predict:*

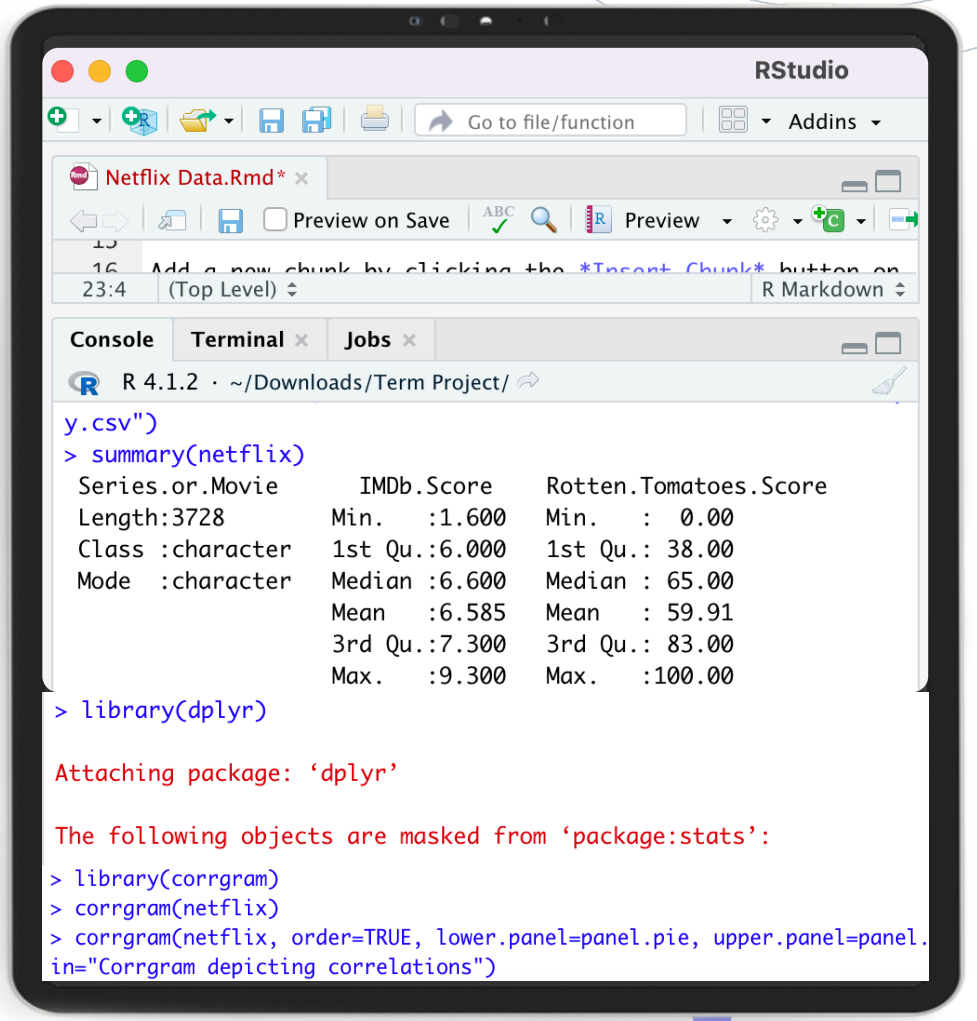
*Hidden Gem scores categorized as "good" (>=7) and "not good"(<=6.9)*

RStudio *Link to R Studio Code on GitHub*

kaggle *Link to Dataset*




The **Corrgram** analysis from **R** gave us a visual view of the Hidden gem scores being highly correlated to **Rotten Tomato** Scores and them in turn related to **IMDB Ratings**, so we discarded the Awards and Box Office data from this analysis




**All Python and this R File and output has also been included as attachment in folder**

***There were 3000+ datapoints but only 10% of these were categorized as “Good” Hidden Gem Score Movies***

[Link](#) to Python Run on GitHub


First I ran the Linear Regression since the scores to be predicted were values ranging from 0.0 to 10.0
Then classifying 7>= as Good and others as not Good, I ran the classifiers

# Accuracies that I got through Different Models for the Original Netflix data

Confusion matrices

## Linear Regression

***Mean Absolute Error: 1.01***

***MSE 2.22***

## Logistic Regression

```
[[1043    1]
 [  74    1]]
```

***Accuracy: 0.93***

***Precision 0.50***

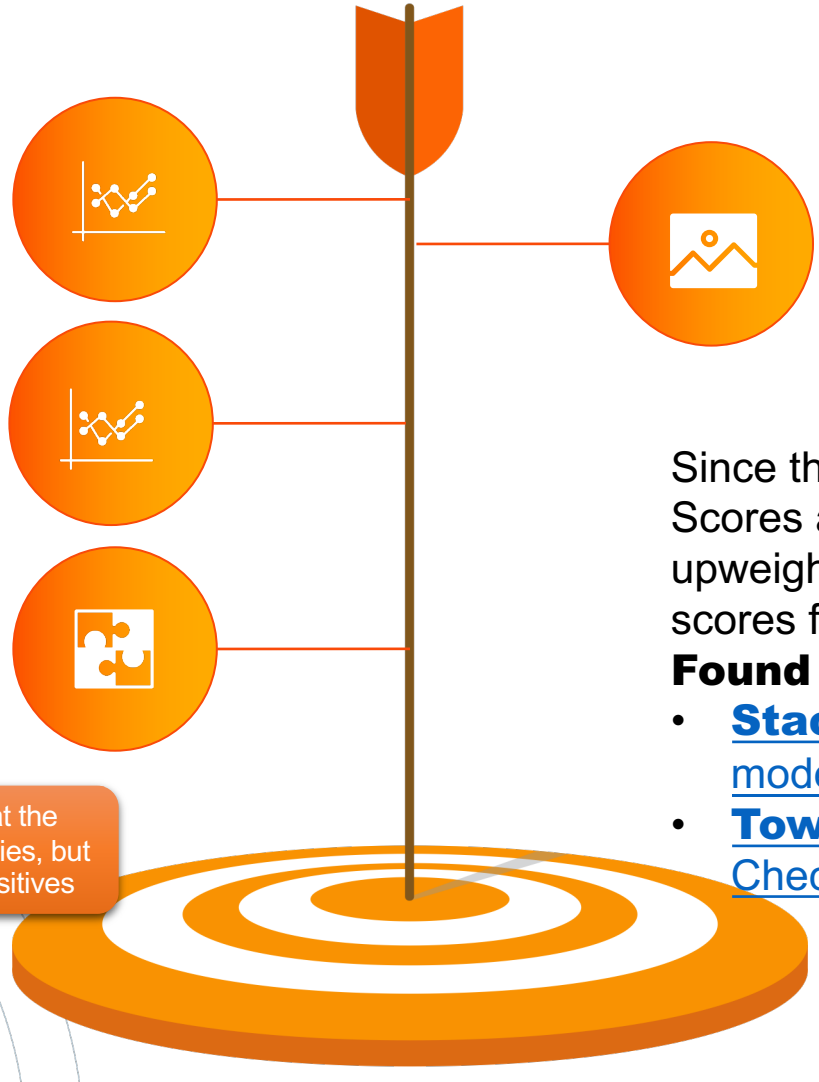
## SVM

```
[[1044    0]
 [  75    0]]
```

***Accuracy: 1.00***

***Precision 1.00***

The issue here is not that the models have good accuracies, but that all predictions are Positives



Confusion matrices

## Random Forest

***Accuracy: 1.00***

***Precision 1.00***

```
[[ 81   0]
 [  0 138]]
```

Since the number of “Good” Hidden Gem Scores are 10% of the data, we need to upweight this and down weight the “Not Good” scores for a reliable predictive model

**Found 2 relevant articles on this:**

- **StackExchange**: 100% accuracies in model
- **Towards Data Science** on Medium: Check if Classification model is Overfit

***Data corrected in the excel file + Checking if better clusters can be made off the Hidden Gem scores than just Good or Bad classes using K-Means Clustering Analysis***

***I corrected the data in excel making it so that the "Good" scores were at least 25-30% of the total dataset and re-ran the analysis with 1000+ datapoints***

[Link](#) *to corrected Python Run on GitHub*


The revised runs from all model are now not skewed to a Positive predict and the Accuracies are:

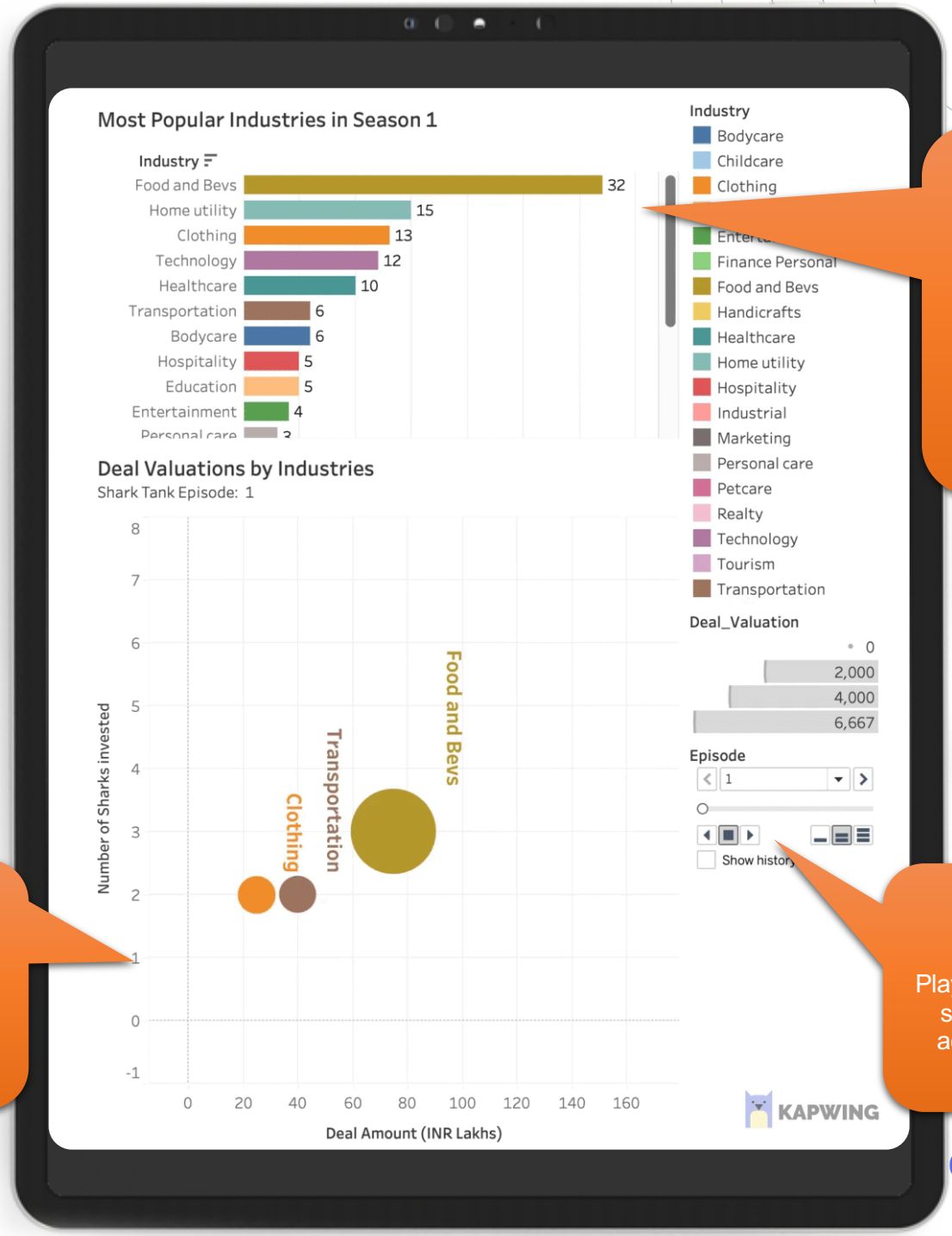
**Logistic Reg: 84%**
SVM 82%
Random Forest 78%
Decision Tree: 78%
K-NN = 81%
Naïve Bayes: 82%

I also ran a **K Means** clustering to see if the data can be better clustered rather than just Good or Bad ratings, and sure enough the Elbow method threw up **3 possible clusters with unique centroids**:

# Understanding Shark Tank India data using Tableau Visualizations

***A lot can be learned from the data through pure crosstabs and clear data visuals:***

*Link to Tableau Public Dashboard with Story Play across all Episodes of the show*




**India Loves to Eat!** Foods and Beverages had the highest number of pitches and conversions

The biggest Deal Valuations that got converted came through **Education and Healthcare** – **Social Benefit matters** when it comes to Start-up investors in India

Play for the episode wise story here, once you access the link to the dashboard

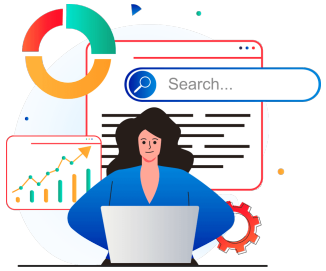
# *Some useful Tableau tutorials:*

- String based calculations: Link on Youtube
- Creating a Storyboard: Link
- Motion charts in Tableau: Link

# Key Nuggets from this Data Analysis

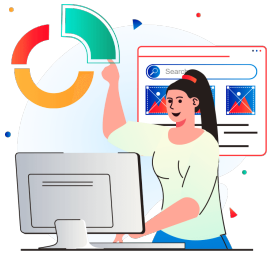
### *Data Preprocessing needs time*

A lot of effort went into **sorting excel files**, **standardizing** variables in Python and understanding **overweighted** data, running **correlation** which help run a better ML model




### *The Bigger the Sample the Better the run*

When we got a bigger dataset either through the US version or Netflix, the **accuracies** of the model **improved**, but do check for overweighted classes



### *Run all possible Classifiers to know the best*

**Naïve Bayes is not always** the go to model as was my previous assumption. The Netflix data had better accuracy using the Logistic Regression


### *There is always a better way to cluster or view the data*

Do not assume that the existing classes are the only classes, sometimes an **unsupervised cluster analysis can give better segments** of the data. Also **visualizations help learn more** about the data than even ML sometimes

# *Thanks for taking the time to read this!*

*Please reach out for further discussion – also I've tried to keep all relevant material used in this GitHub Notebook:* IITD-BAO-21-Notebooks