# COMPARISON OF CLASSIFICATION ALGORITHMS TO PREDICT THE SUCCESS OR FAILURE OF A STARTUP

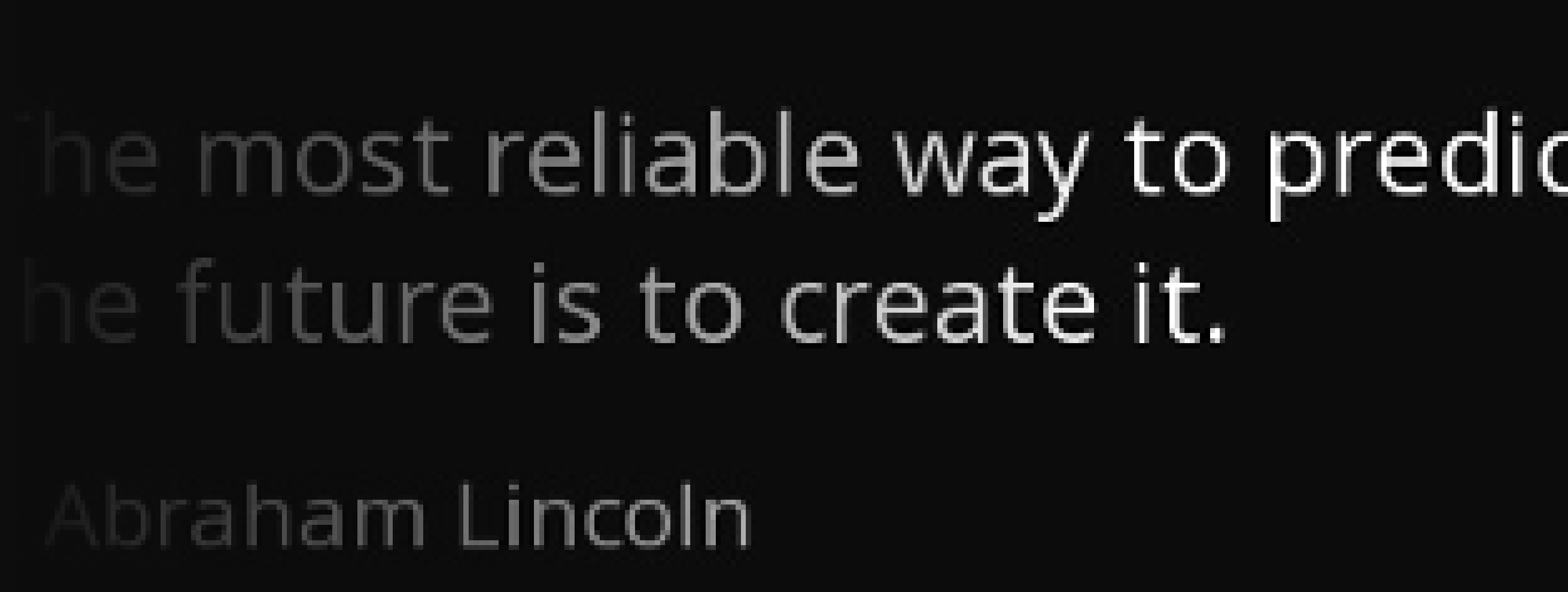


PRESENTED BY

HARSHITHA MOHANRAJ RADHIKA

FNU MARIA POULOSE

SAMIKSHA TALWEKAR

## MOTIVATION AND PROJECT BACKGROUND

**PROBLEM STATEMENT:** To predict the success or failure of a startup which allows investors to identify companies with the potential for rapid growth, allowing them to stay one step ahead of the competition.

**SOLUTION:** Creating a ML model using supervised algorithms to classify the startups as acquired or closed using the important features which impact the growth of the startups

# Significance to the real world

- Predicting a startup's success allows investors to identify companies with the potential for rapid growth, allowing them to stay one step ahead.
- Startups play a significant role in economic growth. They bring new ideas, stimulate innovation, and create jobs, thereby moving the economy forward. Startups have grown at an exponential rate in recent years.
- Success prediction also helps audiences who want to implement their business idea and need guidance for estimating their idea's success, thus encouraging them to pursue the idea.
- There will be an increase in good ideas coming into the market, thus giving founders and investors the tools, methods, and advice that will give them a competitive advantage

# LITERATURE SURVEY

| TITLE OF THE PAPER | GOAL OF THE PAPER | ALGORITHMS USED | CONCLUSIONS/RESULTS |
|---|---|---|---|
| **Assessment of Machine Learning Performance for Decision Support in Venture Capital Investments** | To predict the startup failure using the feature like IPO achieved in which stage of progress ,based on funding and the startup founder's data | Decision Trees, Random Forests, Extremely Randomized Trees, Gradient Tree Boosting, SVM | Gradient Tree Boosting worked well when compared with the other algorithms and achieved a 82% accuracy |
| **Web-based Startup Success Prediction** | To predict whether a company that has already received initial (seed or angel) funding will attract a second round of investment. | Gradient Boosting model | company mentions on the Web yields a significant performance boost , gain insights into both the types of useful signals that can be found on the Internet and the market mechanisms that underpin the funding process |
| **Prediction of the Success of Startup Companies Based on Support Vector Machine and Random Forest** | Prediction of success of a startup using the ML models and then applying various evaluation metrics to find the best model | Random Forest classifier and SVM | Random Forest got 89% accuracy ,f1-score of 0.93 |

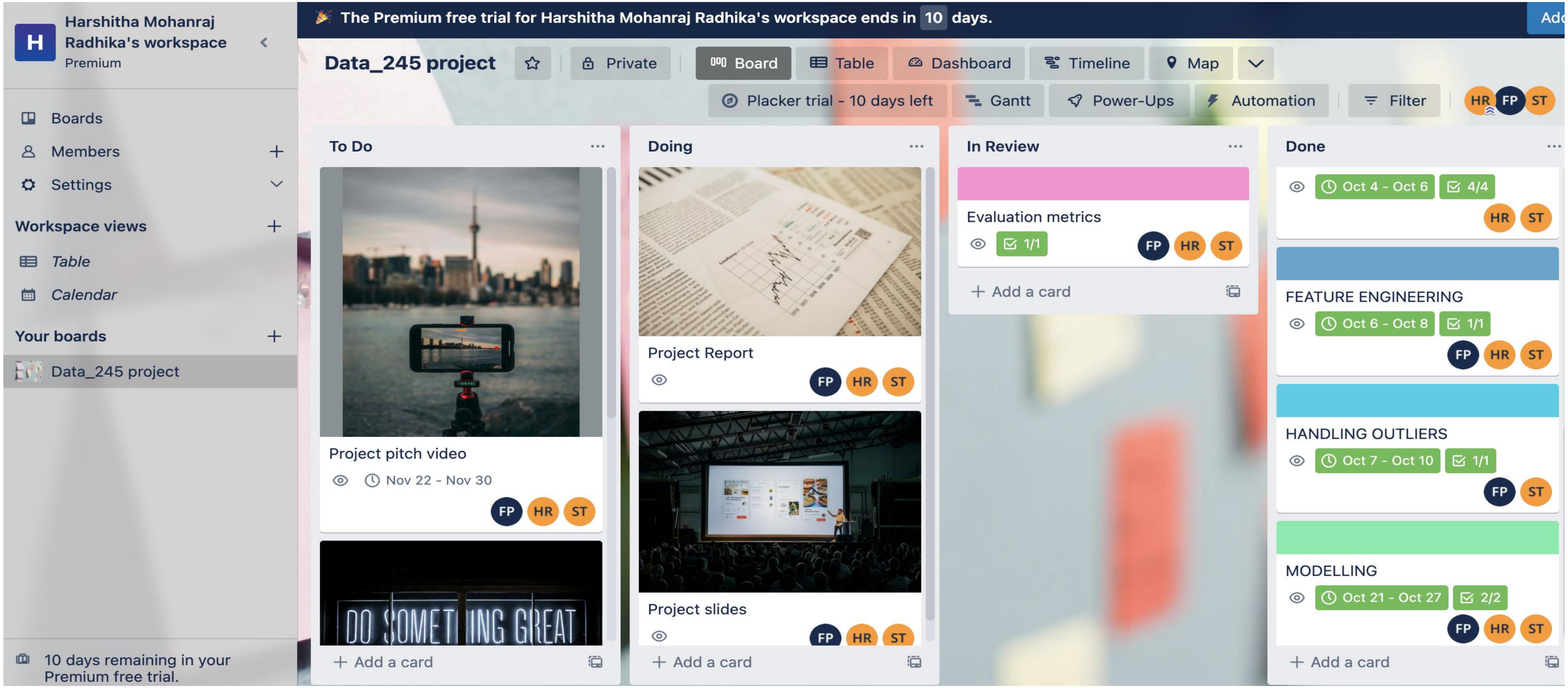
# INNOVATION

For a specific data set, a single method might not produce the ideal prediction. Machine learning algorithms have their limitations, and it might be difficult to create a model with high accuracy. We can increase the accuracy overall if we create and merge numerous models. This is done using ensemble modeling and we implemented in our model.

# PROJECT MANAGEMENT

**TRELLO BOARD TO PLAN THE PROJECT,SCHEDULE THE TASKS,ASSIGNEE MEMBERS FOR EACH TASKS AND KEEP TRACK OF THE PROGRESS**

https://trello.com/invite/b/X96SsSxf/ATTId27cfa2bcda58f1e0e09fbc175ea27210BC03265/data245-project



# DATA COLLECTION

LOADING THE DATASET

```
df=pd.read_csv("startup.csv")
```

## RAW DATASET

```
df
```

| | Unnamed: 0 | state_code | latitude | longitude | zip_code | id | city | Unnamed: 6 | name | labels | ... | object_id | has_VC | has_angel | has_round/ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1005 | CA | 42.358880 | -71.056820 | 92101 | c:6669 | San Diego | NaN | Bandsintown | 1 | ... | c:6669 | 0 | 1 | |
| **1** | 204 | CA | 37.238916 | -121.973718 | 95032 | c:16283 | Los Gatos | NaN | TriCipher | 1 | ... | c:16283 | 1 | 0 | |
| **2** | 1001 | CA | 32.901049 | -117.192656 | 92121 | c:65620 | San Diego | San Diego CA 92121 | Plixi | 1 | ... | c:65620 | 0 | 0 | |
| **3** | 738 | CA | 37.320309 | -122.050040 | 95014 | c:42668 | Cupertino | Cupertino CA 95014 | Solidcore Systems | 1 | ... | c:42668 | 0 | 0 | |
| **4** | 1002 | CA | 37.779281 | -122.419236 | 94105 | c:65806 | San Francisco | San Francisco CA 94105 | Inhale Digital | 0 | ... | c:65806 | 1 | 1 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **918** | 352 | CA | 37.740594 | -122.376471 | 94107 | c:21343 | San Francisco | NaN | CoTweet | 1 | ... | c:21343 | 0 | 0 | |
| **919** | 721 | MA | 42.504817 | -71.195611 | 1803 | c:41747 | Burlington | Burlington MA 1803 | Reef Point Systems | 0 | ... | c:41747 | 1 | 0 | |
| **920** | 557 | CA | 37.408261 | -122.015920 | 94089 | c:31549 | Sunnyvale | NaN | Paracor Medical | 0 | ... | c:31549 | 0 | 0 | |
| **921** | 589 | CA | 37.556732 | -122.288378 | 94404 | c:33198 | San Francisco | NaN | Causata | 1 | ... | c:33198 | 0 | 0 | |
| **922** | 462 | CA | 37.386778 | -121.966277 | 95054 | c:26702 | Santa Clara | Santa Clara CA 95054 | Asempra Technologies | 1 | ... | c:26702 | 0 | 0 | |

923 rows × 49 columns

**DATA SOURCE:**

CSV dataset from Kaggle

**FEATURE NAMES**

Location details, type of industry, name of the companies, about the venture capitalists, is it in the top 500 companies or not and many more.

**NUMBER OF ROWS AND COLUMNS IN TRAINING DATASET:**

923 rows and 49 columns

**NUMBER OF CLASSES OR OUTCOMES:**

2 Classes -Acquired and Closed

# DATA PREPROCESSING

- The status column is the target column where there are two classes acquired and closed.
- Converted the categorical value to numerical value where the acquired is replaced as 1 and closed is replaced as 0 for the purpose of modeling.

**Data Cleaning**

- Checking duplicate values: Removing all the duplicate values in the dataset as they have so much impact on the generation of the model.
- Replacing negative values: Some columns like last_year_fundings have negative values and cause error in the model generation.

```
startup_df['status'] = startup_df.status.replace({'acquired':1, 'closed':0})
startup_df.head()
```

| ude | longitude | zip_code | id | city | Unnamed: 6 | name | labels | ... | object_id | has_VC | has_angel | has_roundA | has_roundB | has_roundC | has_roundD | avg_participants | is_top500 | status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 880 | -71.056820 | 92101 | c:6669 | San Diego | NaN | Bandsintown | 1 | ... | c:6669 | 0 | 1 | 0 | 0 | 0 | 0 | 1.0000 | 0 | 1 |
| 916 | -121.973718 | 95032 | c:16283 | Los Gatos | NaN | TriCipher | 1 | ... | c:16283 | 1 | 0 | 0 | 1 | 1 | 1 | 4.7500 | 1 | 1 |
| 049 | -117.192656 | 92121 | c:65620 | San Diego | San Diego CA 92121 | Plixi | 1 | ... | c:65620 | 0 | 0 | 1 | 0 | 0 | 0 | 4.0000 | 1 | 1 |
| 309 | -122.050040 | 95014 | c:42668 | Cupertino | Cupertino CA 95014 | Solidcore Systems | 1 | ... | c:42668 | 0 | 0 | 0 | 1 | 1 | 1 | 3.3333 | 1 | 1 |
| 281 | -122.419236 | 94105 | c:65806 | San Francisco | San Francisco CA 94105 | Inhale Digital | 0 | ... | c:65806 | 1 | 1 | 0 | 0 | 0 | 0 | 1.0000 | 1 | 0 |

Checking duplicate rows

```
duplicate = startup_df[startup_df.duplicated()]
print("Duplicate Rows :")
```

```
Duplicate Rows :
```

Removing negative values

```
[69] startup_df=startup_df.drop(startup_df[startup_df.age_first_funding_year<0].index)
     startup_df=startup_df.drop(startup_df[startup_df.age_last_funding_year<0].index)
     startup_df=startup_df.drop(startup_df[startup_df.age_first_milestone_year<0].index)
     startup_df=startup_df.drop(startup_df[startup_df.age_last_milestone_year<0].index)
```

# DATA PREPROCESSING(CONT.)

**REMOVING THE IRRELEVANT COLUMNS**: id, object_id, unnamed columns were identified during analysis.

**REPLACING NAN VALUES WITH ZERO**: column/rows which are not required while computation Hence, to carry out any operations we convert it into a numeric value such as 0 or any other values relevant.

**REPLACING NAN VALUES WITH MEDIAN VALUE**: Some columns factors majorly in computation and having insignificant values can hamper the modelling. Therefore, the Nan values are replaced by the median values.

Removing irrelevant columns

```
[64] startup_df = startup_df.drop(["Unnamed: 0", "Unnamed: 6", "labels","closed_at", "id"], axis=1)
```

Filling missing values

```
[67] startup_df['age_first_milestone_year'] = startup_df['age_first_milestone_year'].fillna(startup_df['age_first_milestone_year'].median())
```

```
[68] startup_df['age_last_milestone_year'] = startup_df['age_last_milestone_year'].fillna(startup_df['age_last_milestone_year'].median())
```

Replacing NaN values with zero.

```
startup_df.fillna(0)
```

| | state_code | latitude | longitude | zip_code | city | name | founded_at | first_funding_at | last_funding_at | age_first_funding_year | ... | object_id | has_VC | has_a |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CA | 42.358880 | -71.056820 | 92101 | San Diego | Bandsintown | 1/1/2007 | 4/1/2009 | 1/1/2010 | 2.2493 | ... | c:6669 | 0 | |
| 1 | CA | 37.238916 | -121.973718 | 95032 | Los Gatos | TriCipher | 1/1/2000 | 2/14/2005 | 12/28/2009 | 5.1260 | ... | c:16283 | 1 | |
| 2 | CA | 32.901049 | -117.192656 | 92121 | San Diego | Plixi | 3/18/2009 | 3/30/2010 | 3/30/2010 | 1.0329 | ... | c:65620 | 0 | |
| 3 | CA | 37.320309 | -122.050040 | 95014 | Cupertino | Solidcore Systems | 1/1/2002 | 2/17/2005 | 4/25/2007 | 3.1315 | ... | c:42668 | 0 | |
| 4 | CA | 37.779281 | -122.419236 | 94105 | San Francisco | Inhale Digital | 8/1/2010 | 8/1/2010 | 4/1/2012 | 0.0000 | ... | c:65806 | 1 | |

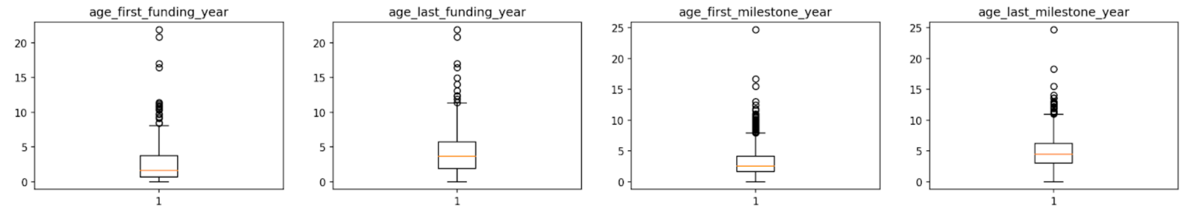
# DATA PREPROCESSING

**Handling Outliers:** During data analysis, we identified outliers using box plots, one of the effective methods to spot outliers is to visualize on the graph. Use those data points and by using data scaling, we spread the data points accordingly.

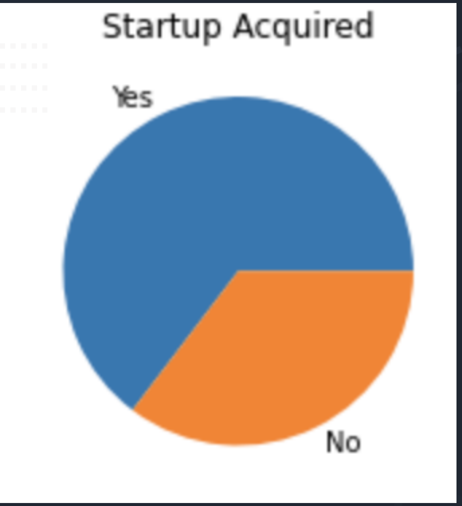
To remove bias towards a certain feature having higher magnitude and smoothing the flow of gradient descent.

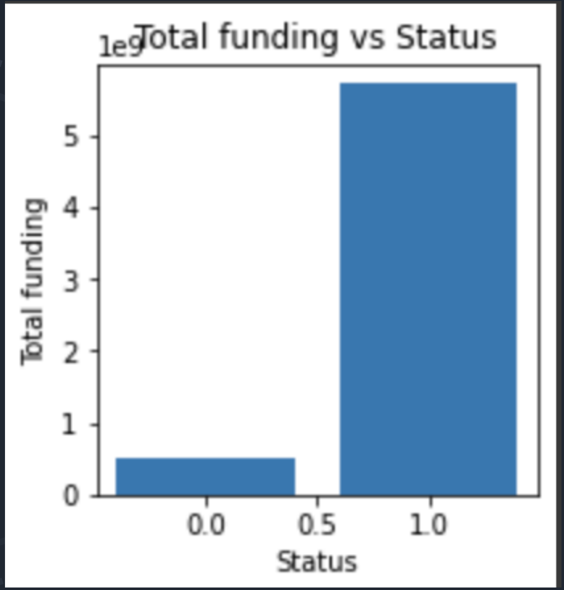
Handling outliers

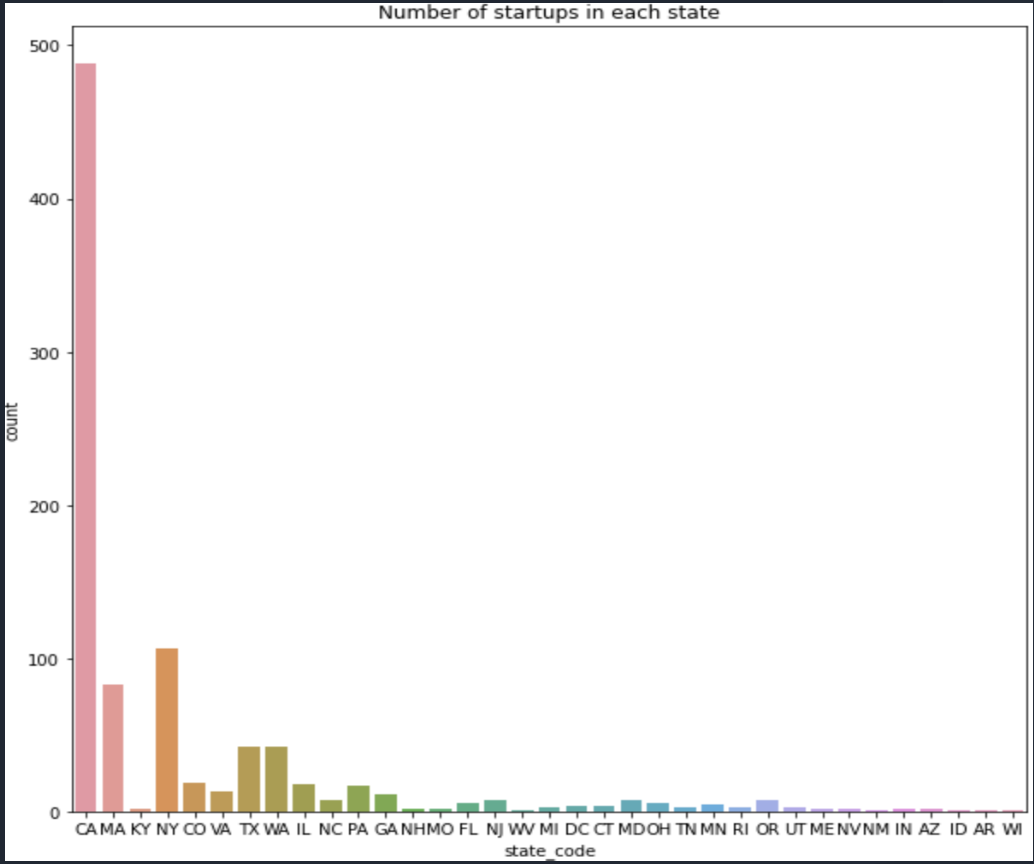
```
variable=["age_first_funding_year","age_last_funding_year","age_first_milestone_year","age_last_milestone_year"]

plt.figure(figsize=(20,3),dpi=150)
for i in range(len(variable)):
    plt.subplot(1,4,i+1)
    plt.title("{}". format(variable[i]))
    plt.boxplot(startup_df[variable[i]]);
```



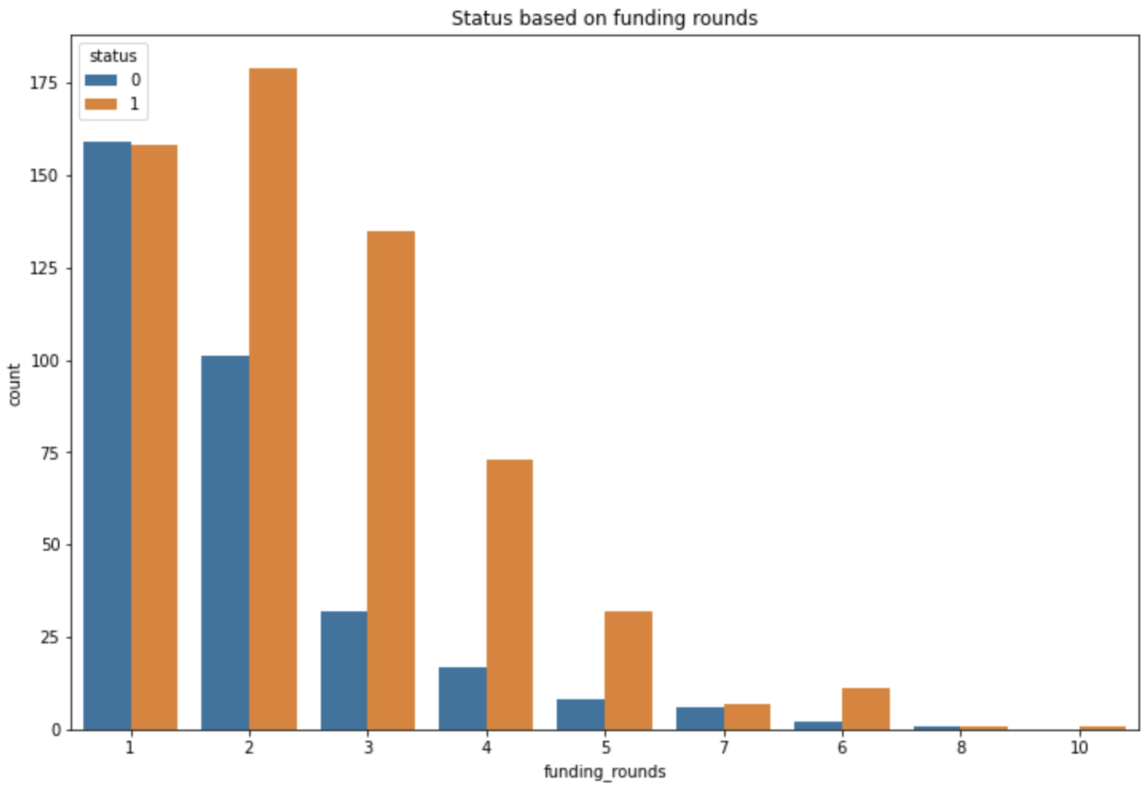
# EXPLORATORY DATA ANALYSIS

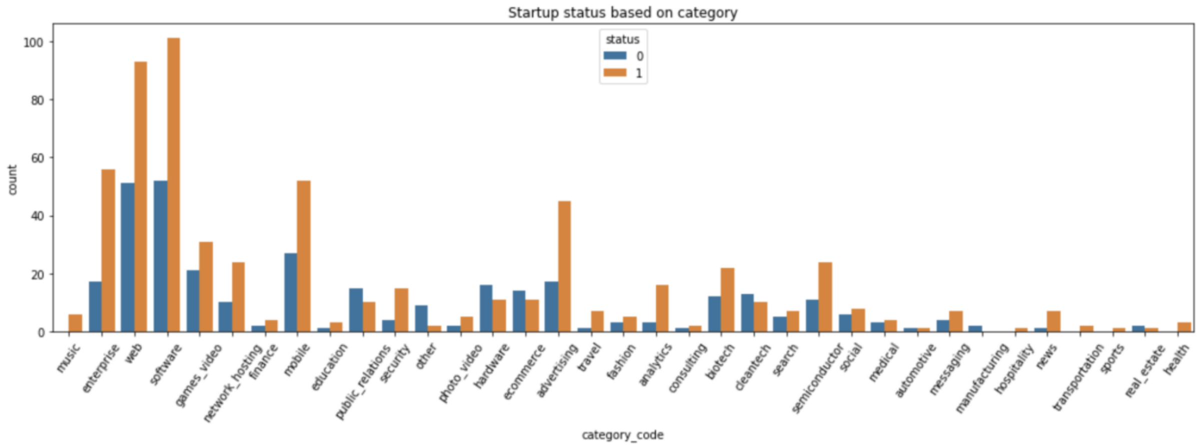


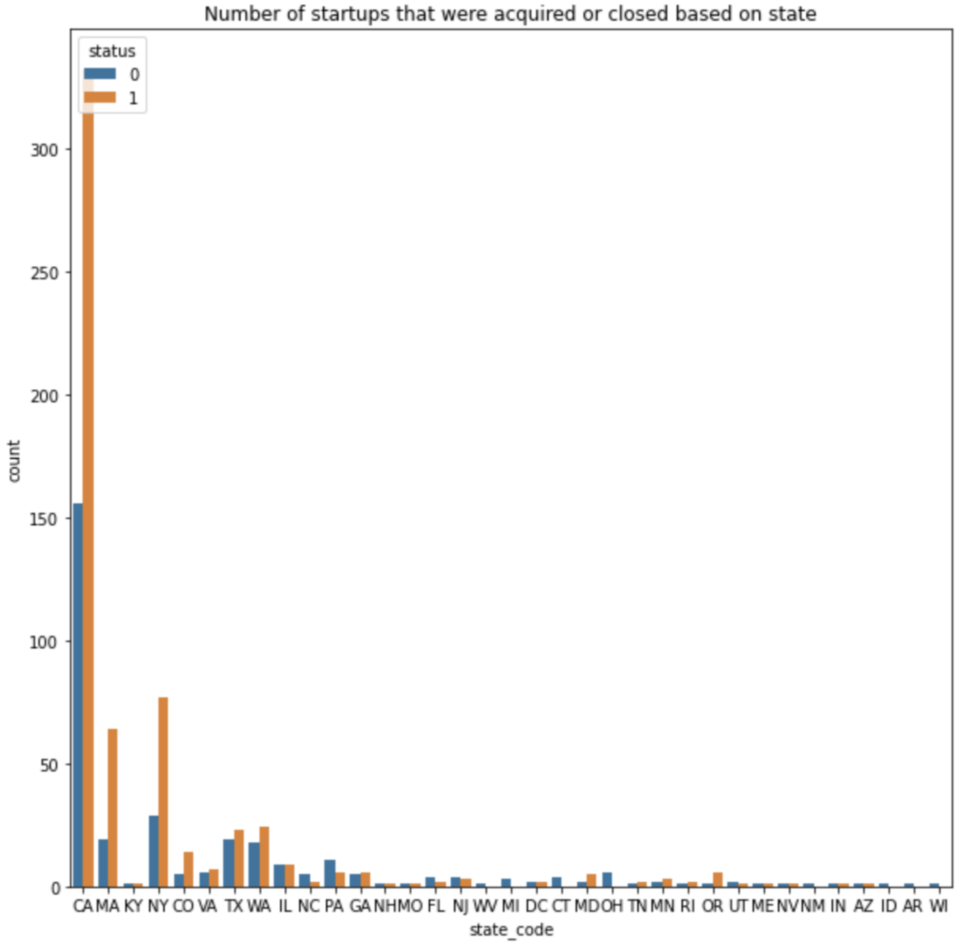



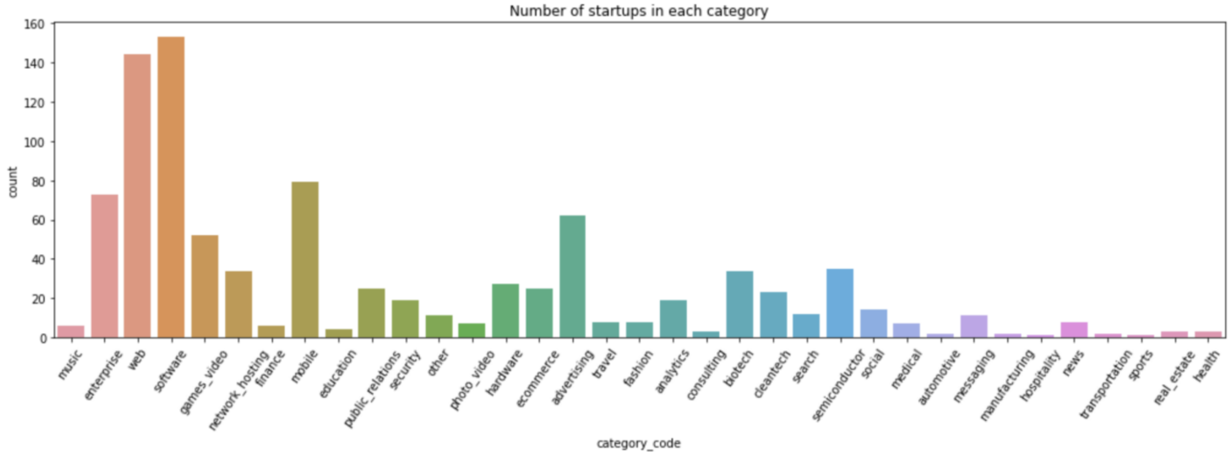
# EXPLORATORY DATA ANALYSIS



# EXPLORATORY DATA ANALYSIS



Creating new column has_investor: It would help us understand the credibility of the startup

```
startup_df['has_Investor'] = np.where((startup_df['has_VC'] == 1) | (startup_df['has_angel'] == 1), 1, 0)
startup_df.head()
```

| last_funding_at | age_first_funding_year | ... | has_angel | has_roundA | has_roundB | has_roundC | has_roundD | avg_participants | is_top500 | status | has_RoundABCD | has_Investor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/1/2010 | 2.2493 | ... | 1 | 0 | 0 | 0 | 0 | 1.0000 | 0 | 1 | 0 | 1 |
| 12/28/2009 | 5.1260 | ... | 0 | 0 | 1 | 1 | 1 | 4.7500 | 1 | 1 | 1 | 1 |
| 3/30/2010 | 1.0329 | ... | 0 | 1 | 0 | 0 | 0 | 4.0000 | 1 | 1 | 1 | 0 |

```
startup_df['has_RoundABCD'] = np.where((startup_df['has_roundA'] == 1) | (startup_df['has_roundB'] == 1) | (startup_df['has_roundC'] == 1) | (startup_df['has_roundD'
startup_df.head()
```

| ng_at | last_funding_at | age_first_funding_year | ... | has_VC | has_angel | has_roundA | has_roundB | has_roundC | has_roundD | avg_participants | is_top500 | status | has_RoundABCD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /2009 | 1/1/2010 | 2.2493 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 1.0000 | 0 | 1 | 0 |
| /2005 | 12/28/2009 | 5.1260 | ... | 1 | 0 | 0 | 1 | 1 | 1 | 4.7500 | 1 | 1 | 1 |
| /2010 | 3/30/2010 | 1.0329 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 4.0000 | 1 | 1 | 1 |
| /2005 | 4/25/2007 | 3.1315 | ... | 0 | 0 | 0 | 1 | 1 | 1 | 3.3333 | 1 | 1 | 1 |
| /2010 | 4/1/2012 | 0.0000 | ... | 1 | 1 | 0 | 0 | 0 | 0 | 1.0000 | 1 | 0 | 0 |

Using a column invalid start up to discard it as an input to the model

```
startup_df['invalid_startup'] = np.where((startup_df['has_RoundABCD'] == 0) & (startup_df['has_VC'] == 0) & (startup_df['has_angel'] == 0), 1, 0)
startup_df.head()
```

| _funding_at | age_first_funding_year | ... | has_roundB | has_roundC | has_roundD | avg_participants | is_top500 | status | has_RoundABCD | has_Investor | has_Seed | invalid_startup |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/1/2010 | 2.2493 | ... | 0 | 0 | 0 | 1.0000 | 0 | 1 | 0 | 1 | 1 | 0 |
| 12/28/2009 | 5.1260 | ... | 1 | 1 | 1 | 4.7500 | 1 | 1 | 1 | 1 | 0 | 0 |
| 3/30/2010 | 1.0329 | ... | 0 | 0 | 0 | 4.0000 | 1 | 1 | 1 | 0 | 0 | 0 |

# FEATURE ENGINEERING

Created a few new columns by combining multiple columns into one.

# FEATURE ENGINEERING

- **Scaling the data:** To remove bias towards a certain feature having higher magnitude and smoothing the flow of gradient descent.

```python
from sklearn.preprocessing import StandardScaler

scale= StandardScaler()
scaled_data = scale.fit_transform(X)
```

# SPLITTING THE DATA

```
X = startup_df[['relationships', 'milestones', 'is_top500', 'has_roundB', 'funding_rounds', 'age_last_milestone_year',
               'avg_participants', 'has_roundA', 'has_roundC', 'has_roundD', 'age_first_milestone_year', 'is_MA',
               'is_CA', 'is_enterprise', 'age_last_funding_year', 'is_NY', 'latitude', 'is_advertising', 'is_advertising',
               'funding_total_usd', 'is_software', 'is_mobile', 'is_consulting', 'is_biotech', 'is_biotech', 'is_web',
               'is_gamesvideo', 'longitude', 'is_othercategory', 'is_TX', 'has_VC', 'is_ecommerce', 'has_angel',
               'age_first_funding_year', 'is_otherstate']]
```

```
y = startup_df['status']
```

```
print("The Shape of the X Train :", X_train.shape)
print("The Shape of the X test :", X_test.shape)
print("The Shape of the y Train :", y_train.shape)
print("The Shape of the y test :", y_test.shape)
```

```
The Shape of the X Train : (672, 35)
The Shape of the X test : (168, 35)
The Shape of the y Train : (672,)
The Shape of the y test : (168,)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

The dataset is split as 80% training and 20% testing sets.

# MODELLING

We have used Classification algorithms of machine learning to predict the success of a startup.

Classifications Models used:

-SVM

-Random Forest classifier

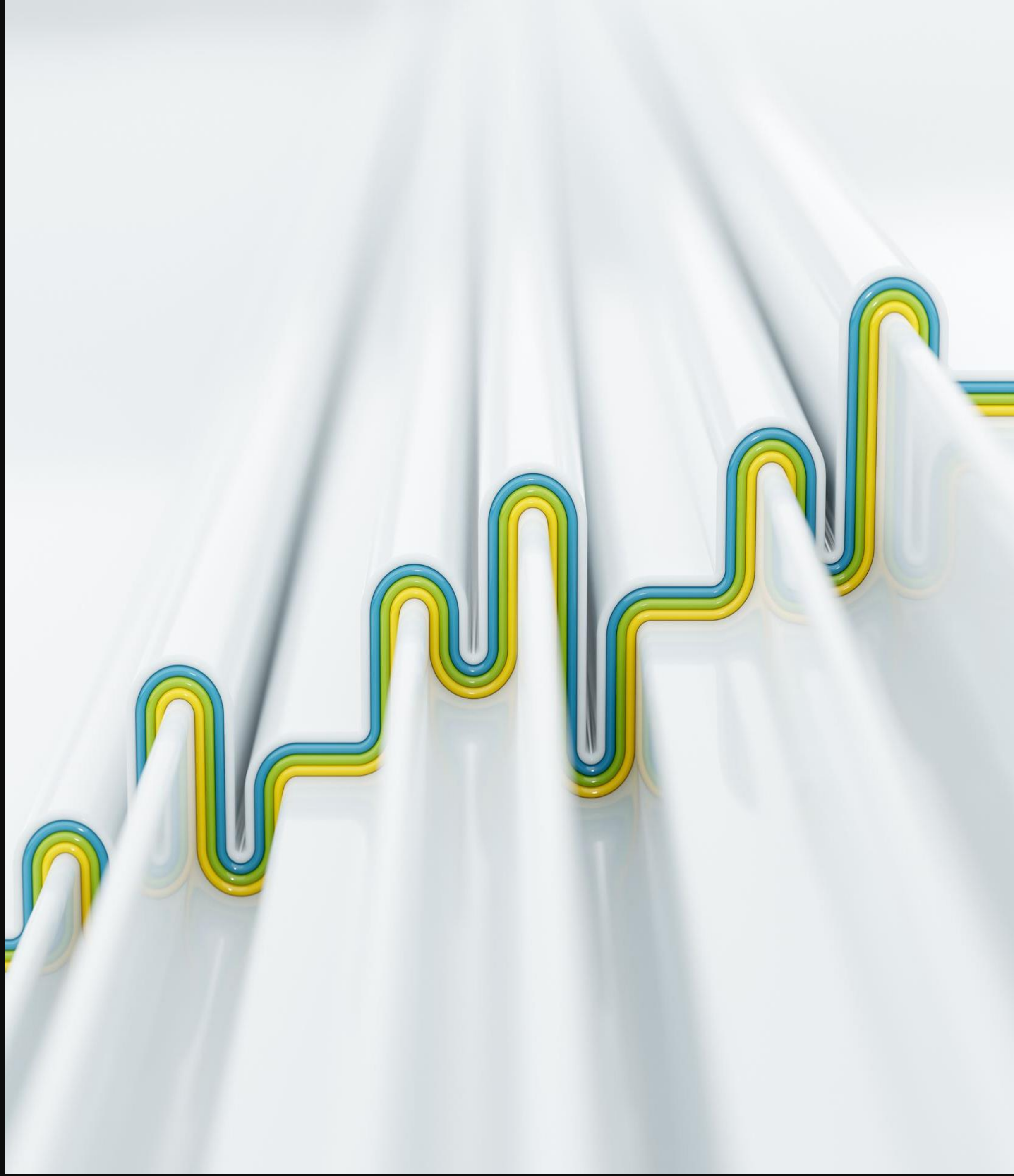
-Logistic Regression

-Decision Tree Classifier

-Gradient Boosting Classifier

-KNN(K-nearest neighbors )

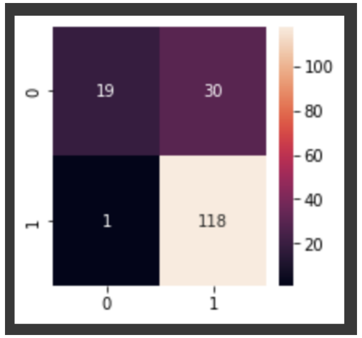
-Ensemble modelling



# SVM

- Support Vector Machine classification calculations for two-group classification issues
- In SVC method (where n is the number of feature you have), we plot each data item as an individual point in space with the value of each feature being a coordinate

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 49 |
| 1 | 0.71 | 1.00 | 0.83 | 119 |
| accuracy | | | 0.71 | 168 |
| macro avg | 0.35 | 0.50 | 0.41 | 168 |
| weighted avg | 0.50 | 0.71 | 0.59 | 168 |



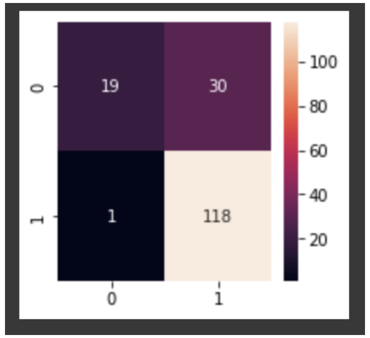

```
print("Accuracy:",accuracy_score(y_test, y_pred_sv))
Accuracy: 0.7083333333333334
```

| Accuracy | 70.83 |
|---|---|
| roc_auc | 0.689 |
| Precision Score | 0.79 |
| Recall Score | 0.99 |
| F1 score | 0.829 |

# Random Forest classifier

- Random Forest is used for classification, and it is based on the concept of gathering learning, which could be a handle of combining numerous classifiers to unravel a complex issue.
- This also help to avoid overfitting.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.59 | 0.73 | 49 |
| 1 | 0.86 | 0.99 | 0.92 | 119 |
| accuracy | | | 0.88 | 168 |
| macro avg | 0.91 | 0.79 | 0.83 | 168 |
| weighted avg | 0.89 | 0.88 | 0.86 | 168 |



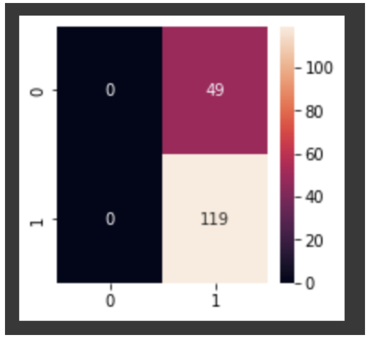

```
print("Accuracy:",accuracy_score(y_test, y_pred_rf))
Accuracy: 0.875
```

| **Accuracy** | **87.5** |
|---|---|
| roc_auc | 0.791 |
| Precision Score | 0.855 |
| Recall Score | 0.991 |
| F1 score | 0.929 |
| Cohen kappa score | 0.644 |

# Logistic Regression

- One of the prominent Machine Learning algorithms for predicting a categorical dependent variable from a set of independent variables.

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        49
           1       0.71      1.00      0.83       119

    accuracy                           0.71       168
   macro avg       0.35      0.50      0.41       168
weighted avg       0.50      0.71      0.59       168
```



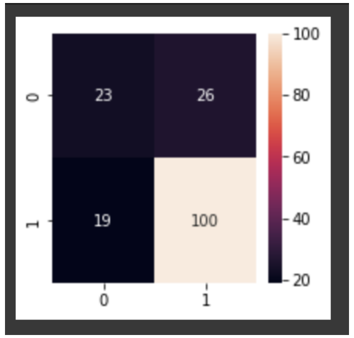

```
print("Accuracy:",accuracy_score(y_test, y_pred_lr))
Accuracy: 0.7083333333333334
```

| Accuracy | 70.8 |
|---|---|
| roc_auc | 0.5 |
| Precision Score | 0.708 |
| Recall Score | 1.0 |
| F1 score | 0.829 |

# Decision Tree Classifier

- Decision tree classifier for a record we tend to begin from the basis of the tree. we tend to compare the values of the basis attribute with the record's attribute.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.47 | 0.51 | 49 |
| 1 | 0.79 | 0.84 | 0.82 | 119 |
| accuracy | | | 0.73 | 168 |
| macro avg | 0.67 | 0.65 | 0.66 | 168 |
| weighted avg | 0.72 | 0.73 | 0.73 | 168 |



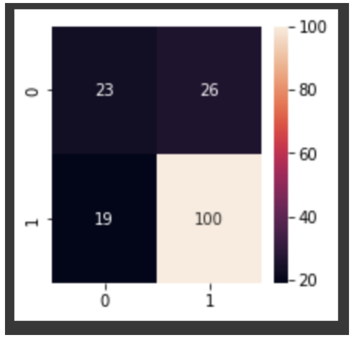

```
print("Accuracy:",accuracy_score(y_test, y_pred_clf))
Accuracy: 0.7321428571428571
```

| Accuracy | 73.2 |
|---|---|
| roc_auc | 0.654 |
| Precision Score | 0.793 |
| Recall Score | 0.84 |
| F1 score | 0.806 |
| Cohen kappa score | 0.23 |

# Gradient Boosting Classifier

- This classifier is built on forward stage-wise fashion and is used when the target column is binary.
- This helps us minimize bias error of the model

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.47 | 0.51 | 49 |
| 1 | 0.79 | 0.84 | 0.82 | 119 |
| accuracy | | | 0.73 | 168 |
| macro avg | 0.67 | 0.65 | 0.66 | 168 |
| weighted avg | 0.72 | 0.73 | 0.73 | 168 |




```
print("Accuracy:",accuracy_score(y_test, y_pred_clf))
Accuracy: 0.7321428571428571
```

| **Accuracy** | **73.21** |
|---|---|
| roc_auc | 0.654 |
| Precision Score | 0.793 |
| Recall Score | 0.84 |
| F1 score | 0.852 |
| Cohen kappa score | 0.42 |

# KNN(K-nearest neighbors )

```
0.6369047619047619
[[34 15]
 [46 73]]
              precision    recall  f1-score   support

           0       0.42      0.69      0.53        49
           1       0.83      0.61      0.71       119

    accuracy                           0.64       168
   macro avg       0.63      0.65      0.62       168
weighted avg       0.71      0.64      0.65       168
```

KNN is one of the classification predictive algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

| Accuracy | 63.6 |
|---|---|
| Roc_auc | 0.65 |
| Precision Score | 0.82 |
| Recall Score | 0.61 |
| F1 score | 0.706 |
| Cohen kappa score | 0.25 |

# Model Comparison

| | SVM | Random Forest Classifier | Logistic Regression | Decision Tree Classifier | Gradient Boosting Classifier | KNN(K-nearest neighbors ) |
|---|---|---|---|---|---|---|
| **Accuracy** | **70.83** | **87.5** | **70.8** | **73.2** | **77.21** | **63.6** |
| roc_auc | 0.5 | 0.791 | 0.5 | 0.654 | 0.694 | 0.65 |
| Precision Score | 0.70 | 0.855 | 0.708 | 0.793 | 0.81 | 0.82 |
| Recall Score | 1.0 | 0.991 | 1.0 | 0.82 | 0.89 | 0.61 |
| Cohen kappa score | 0 | 0.644 | 0 | 0.23 | 0.42 | 0.25 |
| F1 score | 0.829 | 0.929 | 0.829 | 0.806 | 0.852 | 0.705 |

# Ensemble Modelling

```
6.578933539124385
The accuracy of the ensemble method is: 80.95238095238095
              precision    recall  f1-score   support

           0       0.77      0.49      0.60        49
           1       0.82      0.94      0.87       119

    accuracy                           0.81       168
   macro avg       0.80      0.72      0.74       168
weighted avg       0.80      0.81      0.79       168
```

Models used for Ensembing:

Logistic regression

Random forest classifier

Gradient boosting classifier

SVC

# Conclusion

- We were able to successfully build a machine learning model that predicts the success/failure of a startup
- Random forest classifier outperformed other models. With an accuracy of 87.5 %, AUC of 79%, Precision score of 85%, Recall score 99%, and Cohen kappa score 64%.
- We have come to an assumption that if anyone want best result from it, they should take the Random forest classifier as it has the highest accuracy rate.
- SVM and Logistic regression gave us a recall of 1 which means these model can be used for the dataset when we want most accurate prediction

# Future Work

As a future work, we will implement more machine learning models for model training and hyper tune the models to show better accuracy

Create an Interactive user interface that allows the user to gain additional information about companies in each state.

# LESSONS LEARNED

Analyzing the outliers helped us to show better accuracy and recall score.

Model’s performance is not just calculated with the accuracy ,but other metrics like recall, precision, Cohen kappa score plays a major role

Random Forest Regressor is a better classification algorithm for such kinds of dataset

When we need the data to be more accurate, we need to use SVM or Logistic Regression algorithms

Ensemble modelling gave us better accuracy than individual models

# TECHNICAL DIFFICULTY

- Interactive user interface visualization should have been implemented for users to easily make use of this prediction problem. But since our project is not focused on front end part. We were not able to do it.
- We were focused on models relevant to our coursework. May be other models might give a better prediction compared to the models we used.
- Volume of the dataset is very less, we felt we could have achieved better accuracy when there is more data.
- Due to time constraint, we were unable to implement hyper parameter tuning for all the models, but we tried to alter the parameters while doing modelling .Hyperparameter tuning might give a better accuracy for models.

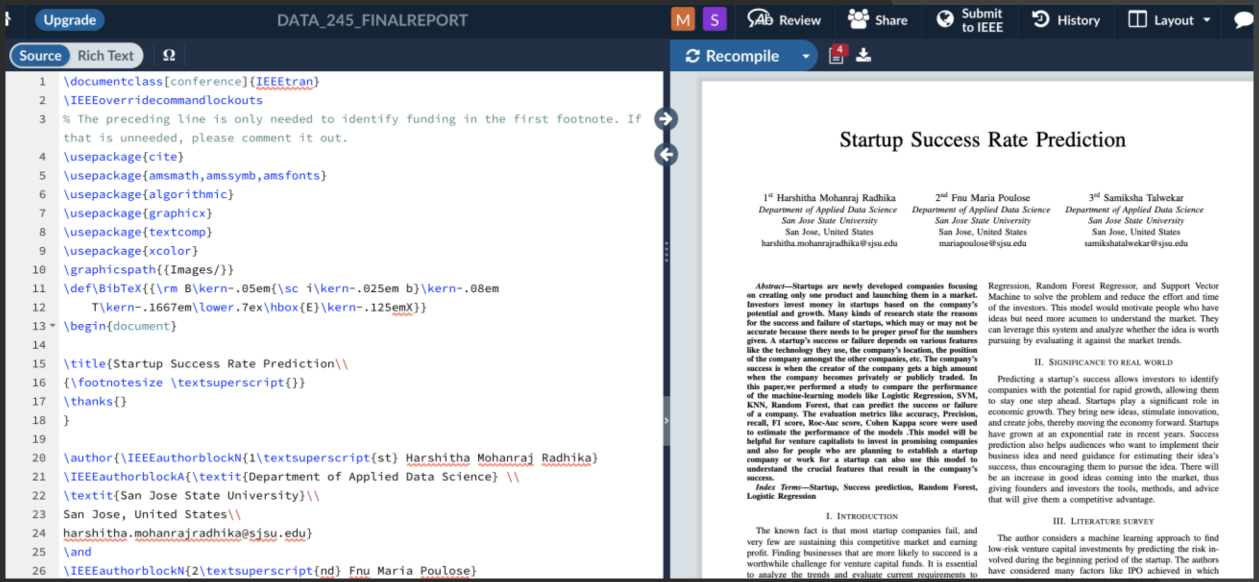
# LINKS

GITHUB REPOSITORY LINK – PAIR PROGRAMMING

https://github.com/samiksha9797/ML_Project

OVERLEAF LATEX LINK-

https://www.overleaf.com/read/mkvmxfwyhfdd

# REFERENCE


- Corea, F., Jimenez-Diaz, G., & Recio-Garcia, J. A. (2019b). Assessment of Machine Learning Performance for Decision Support in Venture Capital Investments. IEEE Access, 7, 124233-124243. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8821312
- Roizner, M., Rumyantsev, A., Ozornin, D., Serdyukov, P., & de Rijke, M. (2018, October 17). Web-based Startup Success Prediction. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. https://doi.org/10.1145/3269206.3272011
- Jinze Li. 2021. Prediction of the Success of Startup Companies Based on Support Vector Machine and Random Forset. In 2020 2nd International Workshop on Artificial Intelligence and Education (WAIE 2020). Association for Computing Machinery, New York, NY, USA, 5–11. https://doi.org/10.1145/3447490.3447492
- Das, S.,Sciro, D.,& Raza, H. (2021, November).CapitalVX: A machine learning model for startup selection and exit prediction.The Journal of Finance and Data Science, 7, 94–114. https://doi.org/10.1016/j.jfds.2021.04.001
- A. Krishna, A. Agrawal and A. Choudhary, "Predicting the Outcome of Startups: Less Failure, More Success," 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016, pp. 798-805, https://doi:10.1109/ICDMW.2016.0118
- Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, ``Electron spectroscopy studies on magneto-optical media and plastic substrate interface,'' IEEE Transl. J. Magn. Japan, vol. 2, pp. 740--741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982]
- M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

# THANK YOU

## Q & A TIME