Presented By: Maha Abdalla Aly

Course name:

ANALYTICS LAB

Project:

Predicting unicorn Success using Machine learning model and business analytics

Acknowledgement

I am deeply grateful to God for His unwavering support and guidance throughout my studies. His love and presence have been a constant source of strength and inspiration, and I am deeply thankful for the many ways in which He has helped me to accomplish my goals. I am also thankful for the numerous blessings He has bestowed upon me, and for the opportunities He has given me to grow and learn. I am truly blessed to have had His support throughout this journey, and I am forever grateful to Him for His love and guidance.

It has been an absolute pleasure and privilege to collaborate with Dr. Cheang Peck Yeng Sharon, whose invaluable guidance and support have been instrumental in the success of my research. Her expertise and mentorship have been invaluable, and I am deeply thankful for the time and effort she has dedicated to my growth and development.

I am also thankful to Dr. Khaw Khai Wah, for his continuous guidance, meticulous suggestions, astute criticism, and inexhaustible patience during the practical and correction phases of my dissertation. His support has been invaluable and has greatly contributed to the successful completion of my studies. I am deeply grateful to both professors for their invaluable contributions and support.

I am deeply grateful to Dr. Yasmine Fahmy for her crucial role in encouraging and coordinating this project, as well as for keeping me updated on scientific developments. her support and guidance have been invaluable during this study, and I am thankful for her unwavering commitment to my success. Thank you, Dr. Yasmine Fahmy, for your invaluable contributions and support.

I am extremely grateful to Dr. Mayada Hadhoud for her invaluable guidance and support as the Guardian Lecturer for the fintech group. Her knowledge, expertise, and consistent encouragement have been instrumental in my academic journey, and I am deeply thankful for the time and effort she has dedicated to my growth and development. I am forever grateful for her support and guidance, and I am thankful to have had the opportunity to learn from such a talented and knowledgeable professor.

I am also thankful to my classmates for their support, feedback sessions, and moral encouragement, which have inspired and impacted me in countless ways. Their contribution to my academic journey has been invaluable, and I am grateful to have had their support throughout my studies.

I

Table of Contents

Abstract

A unicorn company is a business that is founded with the goal of pursuing an innovative idea. Many unicorns are founded by young entrepreneurs and are characterized by their unique and creative approaches to problem-solving. However, not all unicorns are successful in the long run.

The economic significance of unicorns in terms of dynamism, innovation, and competition has been recognized in the literature. However, the unpredictable and volatile nature of the unicorn company ecosystem makes it challenging to analyze and interpret information to determine the likelihood of a unicorn company's success.

Given the time and computational demands of this prediction problem, there is a need for a quantified model that allows for an objective approach to predicting unicorn company success.

In this paper, I aim to develop reproducible models for predicting unicorn success using ML methods. Investors often turn to data-driven approaches to predict the success of unicorn more accurately, rather than relying solely on human expertise and intuition. In recent years, the industry has seen a shift from traditional statistical methods to machine learning (ML) based approaches. The exponential growth of data volume and variety has also led to the emergence of deep learning (DL).

I make use of data scraped from CB Insights. To address sampling bias and imbalance, I apply the oversampling approach SMOTE to the data.

I implement six different models to predict unicorn success. RidgeClassifier, KNN, LDA, LR, NC, and SVC Using various goodness-of-fit measures, I find that the best models are the ensemble SVC, with a test set prediction accuracy of 97% and an ROC of 96.9%.

The models developed in this study can be used to predict the success rate of future firms or ventures in a repeatable way using CB Insights' large-scale data. By using this tool, I hope to provide valuable insights and guidance to aspiring entrepreneurs as they embark on their unicorn journey.

Keywords: *ML, classification algorithm, unicorn company, investment, true positive rate (TPR), false positive rate (FPR), business, data analytics, algorithm design, support system.*

List of Figures

List of Tables

List of Acronym

ML	Machine Learning
AI	Artificial Intelligence
LR	Logistic Regression

SMOTE	Synthetic Minority Oversampling Technique
KNN	KNeighborsClassifier
LDA	LinearDiscriminantAnalysis
NC	NearestCentroid
SVC	Support Vector Machines
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic

1. Introduction

In this chapter, I provide the background and context for the study, including the motivation for exploring the use of classification algorithms to predict the success of unicorn companies. I also state the specific aim and objectives of the study, as well as the research questions that will guide the investigation. The scope of the study is defined, including the specific focus and limitations of the research. Finally, I provide a brief overview of the remaining chapters of the report, outlining the structure and content of the document.

Unicorn companies, which are typically associated with the technology industry, include well-known companies such as Uber, Airbnb, and Palantir. However, unicorn companies can also be found in other industries, such as healthcare, finance, and e-commerce.

These companies are often successful due to their innovative business models and their ability to disrupt traditional industries. They may achieve this by introducing new technologies or offering more convenient or cost-effective alternatives to existing products or services.

Despite their potential for success, unicorn companies also carry a high level of risk due to the fastpaced and competitive nature of the markets in which they operate (Hacklin, Björkdahl, & Wallin, 2018).

As a result, predicting the success of these companies has become a major area of interest for investors and entrepreneurs. In recent years, there has been a growing amount of research on the factors that contribute to the success of unicorn companies, including the founding team, business model, market conditions, and regulatory environment (Emir Hidayat, Bamahriz, Hidayati, Sari, & Dewandaru, 2022).

Overall, the future of unicorn companies is uncertain. Some experts predict that a significant number of these companies will fail in the coming years, while others believe that they will continue to thrive and drive innovation in their respective industries. While unicorn companies have the potential to generate significant wealth for their founders and investors, they also carry a high level of risk due to the fastpaced and competitive nature of the markets in which they operate.

1.1. Background

The term "unicorn company" was first coined in 2013 by venture capitalist Aileen Lee to describe privately held unicorn with a valuation of over $1 billion. These companies, which are often focused on technology and innovation, have achieved significant market success, and have disrupted traditional industries. In recent years, unicorn companies have gained significant attention due to their rapid growth and potential for generating significant wealth for their founders and investors. Since then, the number of unicorn companies has significantly increased, with more than 500 companies globally achieving this status as of 2021 (Malyy, Tekic, & Podladchikova, 2021).

Predicting the success of unicorn companies has become a major area of interest for investors and entrepreneurs in recent years. This private unicorn, defined as companies valued at over $1 billion, have the potential to generate significant wealth for their founders and investors, but they also carry a high level of risk due to the fast-paced and competitive nature of the markets in which they operate (Singhal, Rane, Wadalkar, Deshpande, & Joshi, 2022).

As a result, investors are increasingly seeking methods and techniques to accurately predict the success or failure of unicorn companies. This has led to the development of several predictive models and approaches that use a variety of data sources and machine learning algorithms to identify patterns and trends that can indicate the likelihood of success or failure (Li, Liu, Dong, Lu, & Lu, Exploring Impact Factors of Risk Contagion in Venture Capital Markets: A Complex Network Approach, 2021).

According to data from CB Insights, as of December 2021, there were over 700 unicorn companies globally with a combined valuation of over $2 trillion. The United States is home to the largest number of unicorn companies, followed by China and India. In a short period of time, unicorn companies have achieved significant success and have disrupted traditional industries and business models (cbinsights, 2022).

However, unicorn companies also face challenges, such as the need to maintain high levels of growth, navigate regulatory and legal issues. It will be interesting to see how the unicorn landscape evolves in the coming years and how these companies continue to shape the global economy.

One key factor in the success of unicorn companies is their ability to adapt to changing market conditions and technological advances, as well as their ability to identify and pursue new opportunities.

Data analysis and machine learning techniques can be used to help companies identify trends and patterns in the market and develop strategies to respond to these changes (Li, Liu, Dong, Lu, & Lu, Exploring Impact Factors of Risk Contagion in Venture Capital Markets: A Complex Network Approach, 2021).

Finally, unicorn companies must also be able to effectively manage their finances and resources to achieve long-term sustainability. This requires a strong financial foundation, as well as the ability to access capital and other resources as needed. Data analysis can be used to help companies understand their financial health and identify opportunities for growth and optimization (Suroso J. S., Kaburuan, Setyo, & Jery, 2020).

Overall, the ability to accurately predict the success of unicorn companies is critical for investors and entrepreneurs, greatly aiding them in the decision-making process. By using data analysis and machine learning techniques, it is possible to identify key trends and patterns that can help companies make informed decisions and achieve long-term success. In this research, I aim to investigate the characteristics of the unicorn ecosystem by analyzing the sectors, locations, and driving factors of these companies. I will examine the trends and patterns that have surfaced in the landscape of unicorns, including the successes and challenges faced by these organizations.

By analyzing a wide range of data points and utilizing advanced algorithms, these models can provide valuable insights into the likelihood of a company achieving long-term success. While no prediction model can offer perfect accuracy, the use of machine learning in this context can significantly increase the chances of making informed and profitable investment decisions. By analyzing this information, I hope to provide a deeper understanding of the current state of unicorn companies and their potential future trajectory (Malyy, Tekic, & Podladchikova, 2021).

1.2. Purpose of the study

The purpose of a study to predict the success of a unicorn company using a classification algorithm might be to identify potential risks and opportunities for the company, as well as to help make informed decisions about investments, partnerships, and strategies. The study might involve analyzing various factors that could impact the company's success.

The results of the study could be used to inform the development of a business plan or to assess the feasibility of different courses of action. They might also provide the investor and co-founder with insight and information that they can use to make informed decisions about their investment in and involvement

with the company. This efficiency can be beneficial for stakeholders seeking to make informed and strategic decisions about their involvement with a unicorn company.

The study could potentially provide more accurate and reliable predictions of the company's success compared to other methods. This could be particularly valuable for the investor and co-founder, as they would be able to use the results of the study to assess the potential risks and opportunities associated with their investment and to develop strategies for maximizing the chances of success for the company.

Additionally, the results of the study could be used to inform the development of a business plan or assess the feasibility of different courses of action. This could help the investor and co-founder identify key areas for focus and allocate resources in a way that is most likely to contribute to the company's success.

1.3. Problem statement

Unicorn companies, which are businesses that are typically supported by digital services and have become integral to innovation systems and economies globally, often require a lot of funding to operate with small teams. As a result, venture capitalists (VCs) closely monitor the performance and progress of unicorn companies to inform their decision-making about whether to invest in a particular unicorn companies' growth. To effectively evaluate a unicorn potential for success, it is crucial to analyze the factors that contribute to a unicorn success and how to measure it.

Predicting the success of a company is a major challenge for investors and founders, as these companies face significant risks and uncertainties. Founders of unicorn companies may be particularly concerned about the potential valuation and return on investment of their company based on industry and country factors. They may want to understand which countries and industries are most favorable for their business and what makes those locations attractive to their industry to attract the right investors.

Investing in unicorn companies can be a risky endeavor due to the high rate of failure among these businesses. Any investor often faces a high level of risk when investing in unicorn companies. In this project, my goal is to use supervised machine learning methods to identify key features that contribute to Unicorn success and predict a company's likelihood of success, to help investors, make informed decisions.

To address this problem, I propose a machine learning model that will predict the success of unicorn companies based on a variety of factors such as the company's, industry, and statues. The model will be trained on a dataset of past unicorn companies and their outcomes (successful or not) and will be able to make predictions on new unicorn companies with a high degree of accuracy. This model will be valuable to investors, entrepreneurs, and market analysts as it will provide a way to identify high potential unicorn companies and make informed decisions about investment and strategy. It will also be of interest to researchers studying the factors that contribute to the success of unicorn companies.

1.4. Objective

The main goal of this project is to create a classification model that can accurately predict the success of unicorn companies, with the aim of assisting investors and co-founders in making informed decisions.

To achieve this, I will develop and test classification algorithms that can accurately and reliably predict the success of a unicorn company. The results of this study will provide valuable insights and information to help investors and founders make informed and strategic decisions about these companies, with the goal of maximizing the chances of success for their unicorn company.

By identifying the most effective strategies and techniques for predicting the success of unicorn companies, I aim to provide valuable insights and guidance to help investors and founders navigate the complex and uncertain landscape of unicorn companies and make informed and strategic decisions about their investments and growth strategies.

The target of the project is to:

1. Identify and gather relevant data on unicorn companies from CB Insights, using both python and manual scraping, including financial statements, investor count, and similar companies.
2. Clean and pre-process the data for analysis and modeling.
3. Analyze unicorn behavior based on several variables.
4. Determine which variables have the greatest impact on unicorn success.
5. Develop and test a range of supervised machine learning models that can predict unicorn success, including RidgeClassifier, KNN, LDA, LR, NC, and SVC.
6. Evaluate the performance of the models using metrics such as accuracy, precision, and recall.
7. Identify key factors that contribute to the success or failure of unicorn companies and develop insights and recommendations for investors and entrepreneurs.

1.5. Research Question

To achieve the objectives, the following research questions were formulated:

1. How can relevant data on unicorn companies be gathered from CB Insights using both python and manual scraping?
2. What methods are most effective for cleaning and pre-processing data for analysis and modeling?
3. What are the key variables that drive unicorn behavior and how can they be analyzed?
4. What variables have the greatest impact on unicorn success and how can they be determined?
5. Which supervised machine learning models are most effective in predicting unicorn success and how can they be developed and tested?
6. How can the performance of machine learning models be evaluated using metrics such as accuracy, precision, and recall?
7. What are the key factors that contribute to the success or failure of unicorn companies and how can insights and recommendations be developed for investors and entrepreneurs?

1.6. Scope

This paper focuses on constructing ML models to predict the success of unicorn companies using the selected performance metrics. The scope of this study does not include a focus on the relative importance of each variable or the consideration of additional variables that are not included in the dataset and may affect the results. The implementation and deployment of the resulting models are also outside the scope of this study.

1.7. Outline

The structure of this report is as follows:

Chapter 1: This chapter contains the introduction to the problem in this study, as well as the aim, objectives, research questions, and scope of the study.

Chapter 2: This chapter elaborates on the literature reviews conducted for this study.

Chapter 3: This chapter explains the data exploration of the study, from data retrieval to Descriptive Statistics

Chapter 4: This chapter reports on the data preprocessing of the experiment performed started from data cleaning until Encoding

Chapter 5: This chapter explains the methodological framework of the study, from data retrieval to model evaluation.

Chapter 6: This chapter reports on the results of the experiment performed on the selected algorithms and the champion model, discusses the empirical findings, implications of the study and the visualization results

Chapter 7: This chapter contains recommendations for future work, a conclusion, limitations and the beneficial buyers.

2. Literature Review

The purpose of this literature review is to examine the use of techniques that can be employed to construct models that can accurately forecast the success of unicorn with minimal human intervention. This chapter presents a review of the literature on this topic, highlighting the machine learning algorithms that have been used in previous studies and their respective outcomes. Additionally, the chapter includes a review of various algorithms.

2.1. The Role of Machine Learning in Predicting the Success of Unicorn Companies

Unicorns are a significant topic in the economic policies of countries worldwide due to their potential for driving economic growth and creating wealth, as well as their impact on innovation and technological development. Their rapid growth, ability to quickly adopt innovative business models and advanced technologies, and lean management approach often make them disruptive players in the global economy, particularly as they often have a global reach. Their dynamic, sometimes unconventional approach to business can challenge traditional corporate and small and medium-sized enterprise models. Therefore, the success of unicorn is not just of interest to entrepreneurs, but also to other stakeholders such as investors, shareholders, suppliers, and customers/clients (Menon & James, 2022).

Accurate predictions of the success or failure of a unicorn can bring value to the entire unicorn ecosystem. For small and medium-sized enterprises, key stakeholders include managers who can use success prediction models to take preventive measures against future business challenges and avoid bankruptcy, sponsors, lenders, and investors who can maximize returns and minimize risk in their business portfolios by identifying healthy companies to invest in, and employees who can make informed career choices and avoid the costs of unemployment in the event of bankruptcy (Kaloferov, Predicting Startup Success Using Support Vector Machine, 2021) (Grant & Rahman, 2021).

For unicorn, the primary stakeholders are the entrepreneurs themselves, who can make informed decisions and pivot in a timely manner to address potential issues and save resources such as financial and human capital, which are often scarce in the early stages of a unicorn (Suroso J. S., Kaburuan, Jery, & Setyo, 2020).

Other important stakeholders in the success of unicorn include investors, who may benefit from prediction models by increasing their success rate with unicorn, and suppliers and customers/clients who may rely on the unicorn products or services and bear risks related to its success or failure (TorresToukoumidis, 2020).

Using classification algorithms to predict the success of unicorn companies involves creating predictive models using historical data that can classify a company as successful or unsuccessful. These models are trained using labeled data and can be used to predict a categorical outcome, such as success or failure (Fujita, Okudo, Nishino, & Nagane, 2021).

Predicting success using supervised learning algorithms, which use labeled training data, is a common approach in this process. To achieve accurate predictions, the data used for training must be relevant and

relevant features must be carefully selected. Unsupervised learning algorithms, which do not use labeled training data, may also be used for tasks such as clustering and dimensionality reduction (Arroyo, Corea, Jimenez-Diaz, & Recio-Garcia, 2019).

ML algorithms can be used to analyze historical data and identify patterns and trends that may be indicative of the future success of a unicorn company. These algorithms can be either supervised, which are trained on labeled data to make predictions about specific outcomes, or unsupervised, which do not use labeled data and are used for tasks such as clustering, dimensionality reduction, and association (Kaloferov, Predicting Startup Success Using Support Vector Machine, 2021).

Supervised learning algorithms are often used to build predictive models that can classify outcomes (categorical values) or predict continuous values (regression problems). To build an effective model for predicting the success of unicorn companies, it will be important to consider a wide range of factors that may impact a unicorn likelihood of success, and to carefully evaluate the data sources and inputs used to train and test the model (Hoang & Hoang, 2022).

2.2. History and Challenges of Predicting the Success of Unicorn Companies

Unicorn success prediction models aim to predict the likelihood of a company achieving success before any disruption occurs. However, it is important to note that all companies fail in their own unique way, so directly applying classification algorithms to this problem may not be effective. Instead, it is important to study and analyze as many failed companies as possible in order to identify key factors that contributed to their failure.

Bankruptcy prediction has been the subject of research for decades, with early studies focusing on statistical modeling to formalize the relationship between variables and make predictions under extreme uncertainty. Most research has focused on corporate bankruptcy and survival models for established companies and small and medium-sized enterprises (SMEs). The use of prediction models in predicting unicorn success dates to the 1950s and 60s, when early models used financial statements and ratios to make predictions. However, these early models did not consider the ability and experience of the management team. Success prediction models traditionally used data from both successful and

unsuccessful companies across different industries. The validity of these models is typically assessed using a confusion matrix.

The research on predicting the success of early-stage companies became more prevalent in the 1990s. One of the first non-financial models for predicting the failure of new ventures was the Lussier Model, which used qualitative variables in a regression model. The full model was based on 15 variables, including record keeping and financial controls, capital, industry experience, management experience, planning, professional advisory, education, staffing, product/service timing, economic timing, age, partners, parent company, minority business owner status, and marketing.

In addition to the Lussier Model, there have been numerous studies demonstrating the relationship between the success of a new venture and the skills and motivation of the management team (Malyy, Tekic, & Podladchikova, 2021).

Overall, it appears that machine learning algorithms can be useful for predicting the success of unicorn companies, although there is still room for improvement. Further research is needed to develop more accurate and robust prediction models.

2.3. Predicting the Success of Unicorn Companies Using Machine Learning Algorithms and Synthetic Minority Over-sampling Technique

Predicting the success of unicorn companies can be approached as a binary classification problem, where the goal is to classify data points into one of two categories: "success" or "failure". This approach allows us to use machine learning algorithms to predict whether a particular unicorn company will be successful (Malyy, Tekic, & Podladchikova, 2021).

According to research, there is no single machine learning model that is consistently better than all others in every situation. Therefore, it is important to test a variety of techniques to determine the most appropriate algorithm for the specific dataset being used.

In this study, six supervised learning algorithms were selected for their simplicity and effectiveness in binary classification problems: (1) RidgeClassifier Algorithm, (2) KNN Algorithm, (3) LDA Algorithm, (4) LR Algorithm, (5) NC Algorithm, and (6)SVC Algorithm. And SMOTE to imbalance the data to further understand the capabilities and limitations of these algorithms, I conducted a literature review on each one.

2.3.1. Synthetic Minority Over-sampling Technique

In this paper, I found an imbalance in the data after training and testing six algorithms. To address this, I used SMOTE. The minority class in the data had a very small number of instances, making it difficult to build a reliable classifier. By synthesizing new instances, SMOTE increases the size of the minority class and helps to balance the class distribution. This can improve the performance of the classifier, as many machine learning algorithms perform poorly when the classes are imbalanced (Erol, Uzbaş, Yücelbaş, & Yücelbaş, 2022).

2.3.2. RidgeClassifier Algorithm

RidgeClassifier is a linear classification algorithm that is based on Ridge Regression, which is a variant of Linear Regression that uses a regularization term to prevent overfitting. The regularization term is based on the L2 norm of the coefficients, which penalizes large coefficients and promotes a more parsimonious model (Czako, Sebestyen, & Hangan, 2021).

The general form of the RidgeClassifier algorithm can be expressed as (y = w*x + b) where y is the predicted class label, w is the weight vector, x is the input feature vector, and b is the bias term (Chang, Liu, Yao, & Zhao, 2022) (Goel, Goradia, & Kakelli, 2021).

The goal of the RidgeClassifier algorithm is to find the weight vector w and bias term b that minimize the loss function, subject to a constraint on the L2 norm of the weight vector. The loss function used in RidgeClassifier is the cross-entropy loss, which measures the difference between the predicted probabilities and the true probabilities. The optimization problem can be expressed as minimize (y - w*x* - *b)^2 + lambda*||w||^2 where lambda is the regularization parameter that controls the strength of the regularization term. The solution to this optimization problem can be found using a variety of optimization algorithms, such as gradient descent or the L-BFGS method. Once the solution is found, the weight vector w and bias term b can be used to make predictions for new data points by evaluating the equation y = w*x + b (Lichouri, 2021).

Overall, RidgeClassifier is a powerful and efficient classification algorithm that is well-suited for many real-world problems. It is a good choice when the data is relatively clean and well-behaved, and the

goal is to predict a binary outcome. However, it may not be the best choice for all situations, and it may be worth considering other algorithms as well.

2.3.3. K-Nearest Neighbors

KNN is a classification algorithm that works by finding the k closest training data points to a new data point and using the labels of those points to predict the label of the new data point. The value of k is a hyperparameter that is chosen by the user (Xi, et al., 2022) (Ren, et al., 2022).

The general form of the KNN algorithm can be expressed as follows:

- Choose a value for k and a distance function.
- For each new data point x, find the k training data points that are closest to x using the distance function.
- Predict the label of x by majority vote, where the label of a data point is counted as a vote. In step 2, the distance function is used to measure the similarity between the new data point x and the training data points. There are several distance functions that can be used, including Euclidean distance, Manhattan distance, and Hamming distance. The choice of distance function depends on the nature of the data and the specific problem being solved.

For example, Euclidean distance is a common choice for continuous variables and is defined as: d(x, y) = sqrt((x1 - y1)^2 + (x2 - y2)^2 + ... + (xn - yn)^2) where x and y are two data points with n features, and xi and yi are the values of the feature. Manhattan distance is another common choice for continuous variables and is defined as: The distance between points x and y can be represented by the equation: d(x, y) = ∑ |xi - yi|, where x and y are n-dimensional points and the sum is taken over all dimensions from 1 to n. Hamming distance is a common choice for categorical variables and is defined as: d(x, y) = sum(xi != yi) where xi and yi are the values of the feature, and the sum is taken over all n features (Sharma, Guleria, Tiwari, & Kumar, 2022).

Overall, KNN is a simple and intuitive classification algorithm that is well-suited for many real-world problems. It is a good choice when the data is relatively clean and well-behaved, and the goal is to predict a categorical outcome. However, it may not be the best choice for all situations, and it may be worth considering other algorithms as well.

2.3.4. Linear Discriminant Analysis

LDA is a classification algorithm that works by finding a linear combination of features that maximizes the separation between different classes. It is a supervised method, meaning that it requires

labeled training data to learn the relationship between the features and the class labels (Nguyen, et al., 2022) (Pinho-Gomes, et al., 2022).

The general form of the LDA algorithm can be expressed as follows:

1. Compute the means and covariance matrices for each class.
2. Find the linear combination of features that maximizes the separation between the means of the different classes.
3. Use the linear combination to project the data onto a lower-dimensional space and make predictions using a simple decision rule.

In step 1, the means and covariance matrices are computed using the training data. The mean is a measure of the central tendency of the data, and the covariance matrix is a measure of the dispersion of the data.

In step 2, the linear combination of features is found by solving the following optimization problem: maximize (m1 - m2)^T * S^-1 * (m1 - m2) where m1 and m2 are the means of the two classes, and S is the pooled covariance matrix, which is a combination of the individual covariance matrices of the two classes.

In step 3, the linear combination is used to project the data onto a lower-dimensional space, and a simple decision rule is used to make predictions. For example, a common decision rule is to assign a data point to the class with the closest mean in the projected space (Xi, et al., 2022) (Goschenhofer, et al., 2022).

Overall, LDA is a powerful and efficient classification algorithm that is well-suited for many realworld problems. It is a good choice when the data is relatively clean and well-behaved, and the goal is to predict a categorical outcome. However, it may not be the best choice for all situations, and it may be worth considering other algorithms as well.

2.3.5. Logistic Regression

LR is a classification algorithm that is used to predict a binary outcome (e.g., 0 or 1, yes or no, true or false). It is a supervised method, meaning that it requires labeled training data to learn the relationship

between the features and the class labels (Chen, et al., 2022). The general form of the LR algorithm can be expressed as follows (y = sigmoid(w*x + b)) where y is the predicted probability of the positive class, w is the weight vector, x is the input feature vector, and b is the bias term. The sigmoid function is defined as: sigmoid(z) = 1 / (1 + exp(-z))

The goal of the LR algorithm is to find the weight vector w and bias term b that maximize the likelihood of the training data, given the model. The likelihood is a measure of how well the model fits the data, and it is defined as the probability of observing the data given the model (Gulati, et al., 2021) (Hou, et al., 2022).

The optimization problem can be expressed as:
maximize sum(y * log(y_pred) + (1 - y) * log(1 - y_pred)) where y is the true class label and y_pred is the predicted probability of the positive class. The solution to this optimization problem can be found using a variety of optimization algorithms, such as gradient descent or the L-BFGS method. Once the solution is found, the weight vector w and bias term b can be used to make predictions for new data points by evaluating the equation y = sigmoid(w*x + b) (Hoang & Hoang, 2022) (Jackson & Adam, 2022).

Overall, LR is a powerful and efficient classification algorithm that is well-suited for many real-world problems. It is a good choice when the data is relatively clean and well-behaved, and the goal is to predict a binary outcome. However, it may not be the best choice for all situations, and it may be worth considering other algorithms as well.

2.3.6. NearestCentroid

Nearest Centroid is a simple and intuitive classification algorithm that works by representing each class by its centroid, which is the mean of all the points in the class. To make a prediction for a new data point, the algorithm calculates the distance from the point to the centroid of each class and assigns the point to the class with the closest centroid (Jaddi & Saniee Abadeh, 2022) (Chauhan, Kshetri, & Ahmad, 2022).

The general form of the Nearest Centroid algorithm can be expressed as follows:

1. Compute the centroid of each class using the training data.
2. For each new data point x, calculate the distance from x to the centroid of each class.

3. Predict the class label of x by assigning it to the class with the closest centroid.
   In step 2, the distance between the data point x and the centroid of each class is calculated using a distance function. There are several distance functions that can be used, including Euclidean distance, Manhattan distance, and Hamming distance. The choice of distance function depends on the nature of the data and the specific problem being solved.

For example, Euclidean distance is a common choice for continuous variables and is defined as: d(x, y) = sqrt((x1 - y1)^2 + (x2 - y2)^2 + ... + (xn - yn)^2) where x and y are two data points with n features, and xi and yi are the values of the ith feature.

Manhattan distance is another common choice for continuous variables and is defined as: The distance between two points x and y can be expressed as the sum of the absolute differences between their corresponding coordinates. Using this formula, d(x, y) = ∑ |xi - yi|, where x and y are n-dimensional points and the sum is taken over all dimensions from 1 to n.| Hamming distance is a common choice for categorical variables and is defined as: d(x, y) = sum(xi != yi) where xi and yi are the values of the ith feature, and the sum is taken over all n features (Gulati, et al., 2021).

Overall, Nearest Centroid is a simple and intuitive classification algorithm that is well-suited for many real-world problems. It is a good choice when the data is relatively clean and well-behaved, and the goal is to predict a categorical outcome. However, it may not be the best choice for all situations, and it may be worth considering other algorithms as well.

2.3.7. Support Vector Classification

SVC (SVMs) are a type of supervised learning algorithm that can be used for classification or regression tasks. The objective of an SVM is to identify the hyperplane in a high-dimensional space that maximally distinguishes between the various classes.

In a classification task, the goal is to predict the class of a given example based on its features. Given a set of training examples with known class labels, the SVM algorithm tries to find a hyperplane that maximally separates the different classes. The hyperplane is a decision boundary that defines the regions of the feature space where examples are classified as belonging to one class or the other (Painuli, Bhardwaj, & köse, 2022).

The SVM algorithm finds the hyperplane by maximizing the margin, which is the distance between the hyperplane and the nearest training data points. In other words, the SVM algorithm tries to find the hyperplane that has the maximum distance to the nearest training examples of any class, so that the decision boundary is as far away from the training examples as possible. This helps to ensure that the classifier is not overfitting to the training data and is able to generalize well to new examples.

The equation for the hyperplane in an SVM is:

$w^T x + b = 0$ where:

- w is the normal vector to the hyperplane, which is a weight vector that determines the orientation of the hyperplane
- x is a set of characteristics that describe an example
- b is the bias term, which determines the position of the hyperplane along the normal vector.
- The decision boundary is determined by the values of w and b that maximize the margin. If the value of $w^T x + b$ is positive, the example is classified as belonging to one class, and if it is negative, it is classified as belonging to the other class.
- To find the hyperplane that maximizes the margin, the SVM algorithm uses an optimization algorithm to solve the following optimization problem:
- maximize M subject to $y_i(w^T x_i + b) >= M$ for all i = 1, ..., n ● where:
- M is the margin
- x_i is the feature vector of the i-th training example.
- y_i is the class label of the i-th training example (either 1 or -1)
- The optimization problem is a constrained optimization problem, where the constraint $y_i(w^T x_i + b) >= M$ for all i ensures that the distance between the hyperplane and the nearest training examples is at least M.

In some cases, the optimization problem may not have a solution where the margin is maximized, and all the training examples are correctly classified. In this case, the SVM algorithm allows for a certain number of training examples to be misclassified, if the margin is still maximized. These misclassified examples are known as support vectors and are the examples that are closest to the decision boundary. The SVM algorithm can also be used for regression tasks, where the goal is to predict a continuous target variable based on the features. In this case, the optimization problem is slightly different, and the goal is to minimize the error between the predicted values and the true values (Kumar, Misra, & Chan, 2022).

One of the key features of SVMs is the ability to use different kernel functions to project the data into a higher-dimensional space to find a hyperplane that can better separate the classes. Some popular choices for kernel functions in machine learning include linear, polynomial, and radial basis function (RBF). These functions can be used to transform data into a higher-dimensional space, allowing for the separation of classes that are not linearly separable in the original feature space. The choice of kernel function can have a significant impact on the performance of the SVM, so it is important to choose the kernel function that works best for a given dataset.

3. Data Preparation and Exploration

Data preparation is the process of getting the data ready for modeling and evaluation. This may include tasks such as Exploratory Visualization, Descriptive Statistics, constructing features from the raw data, and selecting an appropriate evaluation metric.

Data exploration, also known as data visualization or data discovery, is the process of analyzing and visualizing the data to understand its underlying patterns and relationships. This is often done using graphical and statistical techniques, and the goal is to identify trends, anomalies, and other features that may be useful for modeling or decision-making.

3.1. Web scraping

The process of web scraping involves collecting and extracting data from websites in a structured manner, using specific tools or code. In this case, data was collected from CB Insights, a website that provides information on unicorn that have achieved a valuation of over $1 billion, known as unicorn. The first step in the ML process for predicting unicorn success involved web scraping data from CB Insights to gather relevant information on this unicorn.

To web scrape the data, Python code was used to extract structured data from the CB Insights website. Structured data refers to data that is organized in a predetermined format, such as tables or lists. By using Python code, the data could be collected in a systematic and automated manner, saving time and resources compared to manual data collection.

However, not all the data on the CB Insights website was structured in a way that could be easily extracted using code. In these cases, manual web scraping was necessary to collect the desired data. Manual web scraping involves manually sifting through the website and collecting the desired data manually, rather than using code to automate the process.

Overall, the process of web scraping data from CB Insights was an important first step in the ML process for predicting unicorn success. By collecting relevant data on various unicorn companies, including information on their industry, location, and financial performance, the ML algorithm could be trained to make predictions on the success of these companies. In the figures below, we see the Python code that was used to web scrape the data using BeautifulSoup and auto scraper.

```
#import numpy packages & matplotlib.pyplot packages
import numpy as np
import matplotlib.pyplot as plt
#import panda packages for filename
import pandas as pd
#import schedules packages to automatically scraping the website
import schedule
#import times packages to specific the time of web scraping
import time
import requests



from bs4 import BeautifulSoup as bs

from autoscraper import AutoScraper
# importing necessary packages
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
```

```
page = 1
while page != 3:
      url = f"https://www.cbinsights.com/research-unicorn-companies"
      print(url)
      page = page + 1
```

```
https://www.cbinsights.com/research-unicorn-companies
https://www.cbinsights.com/research-unicorn-companies
```

```
page = 1
titles = []
while page != 3:
    url = f"https://www.cbinsights.com/research-unicorn-companies"
    response = requests.get(url)
    html = response.content
    soup = bs(html, "lxml")
    wanted_list = ['Select Investors']
    scraper = AutoScraper()
    result = scraper.build(url, wanted_list)
    for i in range(len(titles)):
        titles.append([titles[i].text])
print(result)
```

Figure 1 Web scrape python code

3.2. Data Retrieval

To begin the process of predicting unicorn success using machine learning (ML) techniques, it is necessary to install the relevant libraries and tools. This includes packages like scikit-learn, which provides a range of fast and efficient tools for ML and statistical modeling, particularly in classification. These libraries can easily be added to the Jupyter Notebook environment, which is a popular platform for data analysis and ML projects.

The dataset that will be used in this study was collected through web scraping of the CB Insights website, which provides detailed information on unicorn companies such as their industry, location, and financial performance. Each column of the dataset represents a specific attribute of a unicorn company, while each row represents an individual company. By using ML algorithms on this dataset, it will be

possible to build models that can accurately predict the success of unicorn companies and inform investment and growth decisions by both investors and founders.

This data includes various features related to unicorn companies. Here is a short explanation of each feature:

- Company: The name of the company.
- Valuation: The calculated value of the company.
- Date_Joined_Year: The year in which the company joined CB Insights' unicorn list.
- Date_Joined: The exact date on which the company joined CB Insights' unicorn list.
- Industry: Which filed the company operate in.
- City: The city in which the company is headquartered.
- Country: : where (country) the company operate.
- Continent: The continent on which the company is headquartered.
- Year_Founded: The year in which the company was founded.
- Funding: The total amount of funding the company has received.
- Select_Investors: A list of the investors who have invested in the company.
- Headquarters: The physical location of the company's main offices.
- Statues: The status of the company (e.g., active, inactive, acquired).
- last_Funding_Type: The type of funding the company received in its most recent funding.
- Exit_Investor_Stage: The stage at which the company's investors exited (if applicable).
- Investors_Count: The number of investors who have invested in the company.
- Deal: The total number of deals the company has completed.
- Portfolio_Exits: The number of companies in the investor's portfolio that have exited.
- Funding_Time_Year: The year in which the company received its most recent funding.
- Funding_Time: The exact date on which the company received its most recent funding.
- Acq: Whether the company has been acquired.
- Number_Of_Funding_Round: The number of funding rounds the company has completed.
- Competitor: The company's main competitors.
- Employee_Number: The number of employees the company has.

This dataset contains 2180 observations and 25 features.

3.3. Feature Selection

Feature selection is the process of identifying and selecting a subset of the most relevant features in a dataset for use in building a ML model. This process is important because it can help improve the performance of the model by reducing overfitting, improving generalization, and increasing interpretability.

From the heatmap from the below figure 2, the correlation coefficient between two features measures how strongly they are related. A correlation coefficient of ±1 indicates that the features are linearly dependent, meaning that one feature can be used to predict the other. If the correlation coefficient is 0, it means that the features are not correlated and provide no information about each other. A correlation of 0.7 is a high level of correlation. When two features are highly correlated, it means that they provide similar information and one of the features can often be removed from the dataset. However, in this dataset, there are no features that are highly correlated and therefore no features have been eliminated.

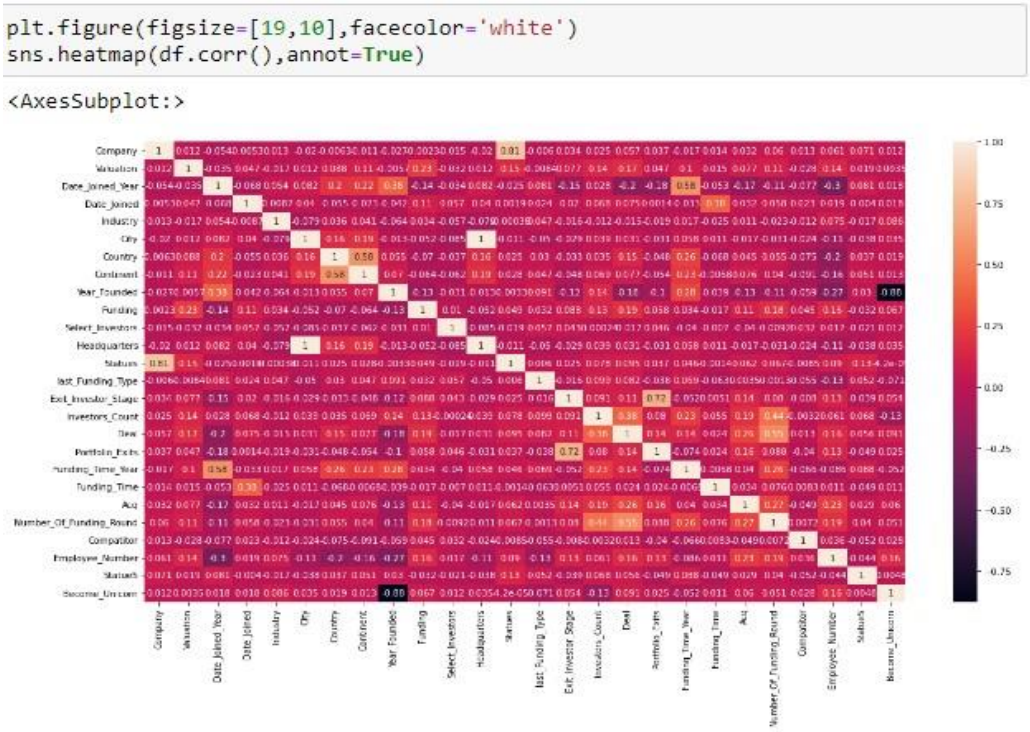


Figure 2 Heatmap Confusion matrix

3.4. Descriptive Statistics

Exploratory data analysis is a crucial step in the data science process as it allows me to understand and visualize the characteristics of my data. By using summary statistics and graphical representations, I can uncover patterns, spot anomalies, test hypotheses, and check assumptions.

In this case, I used summary statistics to understand the characteristics of my unicorn dataset, including the mean, median, and standard deviation for each feature. This helped me to get a better understanding of the data and identify any potential issues or outliers.

In addition, I used graphical representations such as histograms and scatter plots to visualize the data and gain a more intuitive understanding of the relationships between different features. Overall, the exploratory data analysis helped me to get a better understanding of the characteristics of my unicorn data and prepare it for further analysis and modeling.

In the figures 3 below, we see the statistical table. The statistical summary here is important to be analyzed because based on these results, meaningful conclusions can be generated while the data is being predicted and trained. In figure 4 below, we see the distribution of the features in the data. In the figures 4 we see the importance features distribution.

```
# data quantative describe
df.describe()
```

	Date_Joined_Year	Year_Founded	Investors_Count	Deal	Portfolio_Exits	Funding_Tin
count	2180.000000	2180.000000	2180.000000	2180.000000	2180.000000	2180
mean	2020.011009	2013.032110	13.919266	2.811468	0.048624	2020
std	1.922516	5.185045	9.193144	1.979880	0.289677	1
min	2011.000000	1919.000000	0.000000	0.000000	0.000000	2008
25%	2019.000000	2011.000000	8.000000	1.000000	0.000000	2021
50%	2021.000000	2014.000000	12.000000	2.000000	0.000000	2021
75%	2021.000000	2016.000000	18.000000	4.000000	0.000000	2022
max	2022.000000	2021.000000	91.000000	19.000000	5.000000	2022

Figure 3 Descriptive Statics

```
df.hist(bins=25, figsize=(10,10))
plt.show()
```

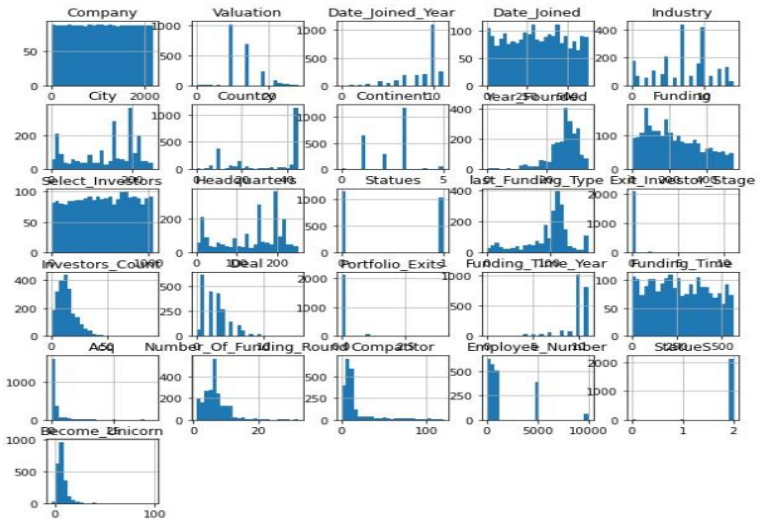


Figure 4 Histogram Distribution

4. Data Pre-processing

In this chapter I will explain data preprocessing is an important step in the ML process. It involves cleaning, formatting, and preparing the data for use in building a model. This can include tasks such as: Data Splitting, Data Wrangling, Data Transformation, Feature Engineering

Data preprocessing is a crucial step in the machine learning process, as it involves cleaning and organizing the data in a way that is suitable for analysis and modeling. This includes managing null values, encoding categorical data, and selecting relevant features.

Encoding categorical data involves converting categorical variables, such as company or industry, into numerical values that can be processed by machine learning algorithms. Feature selection involves identifying the most relevant features in the dataset and selecting only those for further analysis and modeling. Data preparation is important because it helps to ensure that the data is accurate, consistent, and ready for analysis and modeling.

4.1. Manage Null Values

The next step in the process of predicting unicorn success using machine learning (ML) is data preparation, also known as data wrangling or cleansing. This involves managing null values, encoding categorical data, and selecting relevant features for the model. In this dataset, there were some null values, but this is a common issue that can arise for a variety of reasons.

In my case the website didn't have access to all the specific company information, or the company may not have made all its data available to the public. To identify missing values in the data, the Pandas library provides the isna() and isnull() methods. Once any missing values have been identified and dealt with, In Figure 5 below, we see that there is a missing value in the data after web scraping. As illustrated in Figure 6, there are no missing values after the cleaning process has been completed. As shown in Figure 7, the code for cleaning null values is demonstrated.

```
# to know the sum of the NAN value
df.isnull().sum()
```

```
Company                      0
Valuation                 2990
Date_Joined_Year          1514
Date_Joined               1022
Industry                  3424
City                      1146
Country                   3452
Continent                 4421
Year_Founded              2155
Funding                    680
Select_Investors           184
Headquarters               481
StatueS                     52
last_Funding_Type          556
Exit_Investor_Stage       4929
Investors_Count           1113
Deal                       183
Portfolio_Exits           4894
Funding_Time_Year          242
Funding_Time               333
Acq                       3249
Number_Of_Funding_Round   1248
Compatitor                1010
Employee_Number           1660
Year_Founded.1             483
Funding.1                  797
Statues                   4938
dtype: int64
```

Figure 5 missing values check before clean

```
# to know the sum of the NAN value
df.isnull().sum()
```

```
Company                   0
Valuation                 0
Date_Joined_Year          0
Date_Joined               0
Industry                  0
City                      0
Country                   0
Continent                 0
Year_Founded              0
Funding                   0
Select_Investors          0
Headquarters              0
Statues                   0
last_Funding_Type         0
Exit_Investor_Stage       0
Investors_Count           0
Deal                      0
Portfolio_Exits           0
Funding_Time_Year         0
Funding_Time              0
Acq                       0
Number_Of_Funding_Round   0
Compatitor                0
Employee_Number           0
StatueS                   0
dtype: int64
```

Figure 6 missing values check after clean

```
df.dropna(inplace=True)
```

```
df.to_csv('finalclean.csv', index=False)
```

Figure 7 Drop null value python code

4.1.1. Encoding Categorical Data

Encoding categorical data is an important step in the data preparation process, as many machine learning algorithms are unable to handle categorical data in their raw form. To transform categorical. In this case, I encountered difficulty using the categorical data for my analysis. To overcome this, I applied encoding techniques to convert all the categorical data into numerical form for easier processing.

This is typically done using techniques such as one-hot encoding, which creates a new column for each unique category and assigns a binary value to indicate the presence or absence of the category. Finally, feature selection involves selecting the most relevant and useful features for the model, which can help to improve its performance and reduce the risk of overfitting.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2180 entries, 0 to 2202
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Company                 2180 non-null   object
 1   Valuation               2180 non-null   object
 2   Date_Joined_Year        2180 non-null   int64
 3   Date_Joined             2180 non-null   object
 4   Industry                2180 non-null   object
 5   City                    2180 non-null   object
 6   Country                 2180 non-null   object
 7   Continent               2180 non-null   object
 8   Year_Founded            2180 non-null   int64
 9   Funding                 2180 non-null   object
 10  Select_Investors        2180 non-null   object
 11  Headquarters            2180 non-null   object
 12  Statues                 2180 non-null   object
 13  last_Funding_Type       2180 non-null   object
 14  Exit_Investor_Stage     2180 non-null   object
 15  Investors_Count         2180 non-null   int64
 16  Deal                    2180 non-null   int64
 17  Portfolio_Exits         2180 non-null   int64
 18  Funding_Time_Year       2180 non-null   int64
 19  Funding_Time            2180 non-null   object
 20  Acq                     2180 non-null   int64
 21  Number_Of_Funding_Round 2180 non-null   int64
 22  Compatitor              2180 non-null   int64
 23  Employee_Number         2180 non-null   int64
 24  StatueS                 2180 non-null   int64
dtypes: int64(11), object(14)
memory usage: 442.8+ KB
```

Figure 8 Data before encoding

```
: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2180 entries, 0 to 2202
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Company                 2180 non-null   int32
 1   Valuation               2180 non-null   int32
 2   Date_Joined_Year        2180 non-null   int64
 3   Date_Joined             2180 non-null   int32
 4   Industry                2180 non-null   int32
 5   City                    2180 non-null   int32
 6   Country                 2180 non-null   int32
 7   Continent               2180 non-null   int32
 8   Year_Founded            2180 non-null   int64
 9   Funding                 2180 non-null   int32
 10  Select_Investors        2180 non-null   int32
 11  Headquarters            2180 non-null   int32
 12  Statues                 2180 non-null   int32
 13  last_Funding_Type       2180 non-null   int32
 14  Exit_Investor_Stage     2180 non-null   int32
 15  Investors_Count         2180 non-null   int64
 16  Deal                    2180 non-null   int64
 17  Portfolio_Exits         2180 non-null   int64
 18  Funding_Time_Year       2180 non-null   int64
 19  Funding_Time            2180 non-null   int32
 20  Acq                     2180 non-null   int64
 21  Number_Of_Funding_Round 2180 non-null   int64
 22  Compatitor              2180 non-null   int64
 23  Employee_Number         2180 non-null   int64
 24  StatueS                 2180 non-null   int64
 25  Become_Unicorn          2180 non-null   int64
dtypes: int32(14), int64(12)
memory usage: 405.2 KB
```

Figure 9 Data after encoding

4.1.2. Data Splitting

Data preprocessing is an important step in the ML process, as it helps to ensure that the data is in a suitable format for the model to effectively learn from. One common technique for preprocessing data is feature selection, which involves identifying and selecting the most relevant features in the dataset for the model to use. In this case, the heatmap was used to identify any highly correlated features that might provide similar information and therefore be redundant.

Another important step in preprocessing is splitting the data into training and testing sets, which allows for the creation of models and the estimation of their performance on unseen data. The train_test_split function was used to split the dataset into a 75:25 train to test ratio, with the training set used to create and refine models and the testing set reserved for final evaluation.

Dataset	Percentage	Instance
Train	75	1635
Test	25	545

Table 1 Train and Test table

5. Methodology

The methodology for this project involved the use of classification algorithms and Power BI visualization on a computer equipped with a 2.71 GHz Intel(R) Core (TM) CPU processor and 8 GB of RAM running the Windows 10 operating system and Jupyter Notebook (6.4.5).

Python programming language was used for machine learning and Power BI for visualization. The project was completed in five steps: web scraping, data retrieval, preprocessing, data preparation and exploration, modeling, and evaluation.

5.1. Data Modelling

The pseudocode for the ML algorithms in this study outlines the steps for building a classification model using a specific algorithm. The pseudocode for the six classification models mentioned in this paragraph includes steps such as importing the necessary libraries and modules, loading, and preprocessing the data, selecting, and training the model, making predictions using the trained model, and evaluating the model's performance.

These steps may vary slightly depending on the specific algorithm being used, but they generally involve preparing the data, choosing, and training the model, making predictions, and evaluating the model's performance. There are 6 classification models built in this section which includes RidgeClassifier, KNN, LDA, LR, NC, and SVC.

were implemented and trained using the .fit() function on the training set (consisting of X-train and y_train data, where X represents the features and y represents the target). Once trained, these models were used to make predictions on the test set (X_test) using the predict() function. These classification models were imported from the sklearn library into Python, and I used the append method to add them to my data. To decide, I generated the accuracy and f1 scores for each algorithm and recorded them in the table below. use classification algorithms to predict the success of unicorn companies and visualize the results using Power BI. The performance of the model is evaluated using various metrics such as the classification report, Accuracy, and the AUC and receiver operating characteristics curve. The results of the analysis are then visualized using Power BI to better understand and communicate the findings.

5.2. Model Evaluation

TP, FP, FN, and TN are terms used in the field of ML and statistics, particularly in relation to evaluating the performance of a classification model.

- TP (True Positive) refers to the number of correct positive predictions made by the model.
- FP (False Positive) refers to the number of incorrect positive predictions made by the model.
- FN (False Negative) refers to the number of incorrect negative predictions made by the model.
- TN (True Negative) refers to the number of correct negative predictions made by the model.

These terms are often used to compute various evaluation metrics such as precision, recall, and accuracy.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Table 2 Confusion matrix

The performance metrics that used to recognize the classifier performance will explain in the Model evaluation is the process of measuring the performance of a ML model using a set of metrics. It helps to determine how well the model can make predictions on unseen data, and to identify any potential issues with the model. There are a variety of metrics that can be used for model evaluation, depending on the type of problem being addressed. Some commonly used metrics include.

Accuracy: is a measure of how well a model correctly predicts the output for a given input. In a classification problem, it is defined as the proportion of correct predictions made by the model out of all the predictions made. Mathematically, it can be expressed as follows: Accuracy = (Number of correct predictions) / (Total number of predictions) (N.Irino(3), 2020) .

Precision is the proportion of true positive predictions made by the model out of all the positive predictions made. It can be calculated as follows: Precision = (True positive) / (True positive + False positive)

Recall is the proportion of true positive predictions made by the model out of all the actual positive samples in the dataset. It can be calculated as follows: Recall = (True positive) / (True positive + False negative)

The F1 score is the harmonic mean of precision and recall, and it can be calculated as follows: F1 score = 2 * (Precision * Recall) / (Precision + Recall) .The F1 score ranges from 0 to 1, with a higher value indicating better performance. It is a balance between precision and recall and is often used as a metric when there is an uneven class distribution in the dataset, or when it is important to avoid false positives or false negatives.

	Actual spam	**Actual not spam**
Predicted spam	TP=100	FP=50
Predicted not spam	FN=10	TN=140

Table 3 F1 score matrix.

Using the equations above, I can calculate the precision, recall, and F1 score of the model as follows:

Precision = (TP) / (TP + FP) = 100 / (100 + 50) = 0.67

Recall = (TP) / (TP + FN) = 100 / (100 + 10) = 0.91

F1 score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.67 * 0.91) / (0.67 + 0.91) = 0.77 (N.Irino(3), 2020).

AUC or ROC curve is a metric used to evaluate the performance of a binary classification model. It plots the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds, and the AUC is the area under the curve. The AUC ranges from 0 to 1, with a higher value indicating better performance. A model with an AUC of 1 is a perfect classifier, while a model with an AUC of 0.5 is a random classifier.

	Actual spam	**Actual not spam**
Predicted spam	TP=100	FP=50
Predicted not spam	FN=10	TN=140

Table 4 AUC confusion matrix

Using the equations above, I can calculate the TPR and FPR at different classification thresholds as follows:

- At a classification threshold of 0.9: TPR = (TP) / (TP + FN) = 100 / (100 + 10) = 0.91, FPR = (FP) / (FP + TN) = 50 / (50 + 140) = 0.26
- At a classification threshold of 0.8: TPR = (TP) / (TP + FN) = 100 / (100 + 10) = 0.91, FPR = (FP) / (FP + TN) = 50 / (50 + 140) = 0.26
- At a classification threshold of 0.7: TPR = (TP) / (TP + FN) = 90 / (90 + 10) = 0.90, FPR = (FP) / (FP + TN) = 60 / (60 + 140) = 0.30
- At a classification threshold of 0.6: TPR = (TP) / (TP + FN) = 80 / (80 + 10) = 0.89, FPR = (FP) / (FP + TN) = 70 / (70 + 140) = 0.33

I can plot the TPR and FPR values on a graph to visualize the ROC curve. The AUC is then calculated as the area under this curve. In this example, the AUC might be around 0.8, indicating that the model has good performance (N.Irino(3), 2020).

6. Result and Discussion

In this chapter, I will discuss the visualization techniques and machine learning algorithms that is used to predict "unicorn success" and the features that have an impact on it. I will also present the "champion model," which is the model with the best performance among all the models that we trained and evaluated. My goal is to understand how to accurately predict "unicorn success" and to identify any patterns or trends in the data that could be helpful in this task.

6.1. Visualization Result

In this section, I will present the results of our data visualization efforts to gain a deeper understanding of the factors that influence "unicorn success," which can be either successful or unsuccessful. The visualizations in the appendix, under the Data Visualization section, show the elements that influenced the outcome, represented as either Yes (for successful) or No (for unsuccessful) in the target column.

I have visualized the features to identify the factors that contribute to "unicorn success." You can find the visualizations of categorical and numerical features in Appendix A.

6.1.1. The industry with highest valuation

As shown in Figure 4, the fintech industry appears to be the most highly valued among the options presented in the chart, with a valuation of approximately 996 billion. This industry is the most attractive to investors compared to other industries, such as the travel industry which has a lower valuation. While a high valuation may indicate strong growth potential or investor interest, it is important to note that this is just one factor to consider when evaluating investment opportunities. Other crucial considerations include the financial performance and stability of companies within the sector, the competitive landscape, macroeconomic conditions, and the investor's own risk tolerance and investment objectives.

To make well-informed investment decisions, it is crucial to conduct thorough research and due diligence on available opportunities, rather than solely relying on the valuation of an industry.

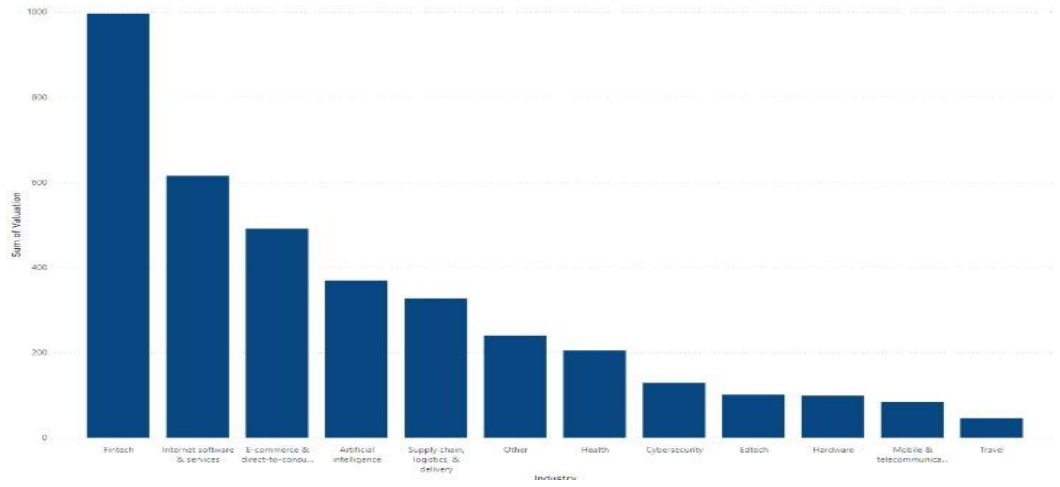


Figure 10 Industry with higher valuation

6.1.2. The industry that has higher protein in the market

Figure 5 illustrates that the technology industry holds the highest percentage in the fintech industry at 24.13%, followed by the internet software and services industry at 19.25%, and then e-commerce and direct-to-consumer delivery at 12.11%.

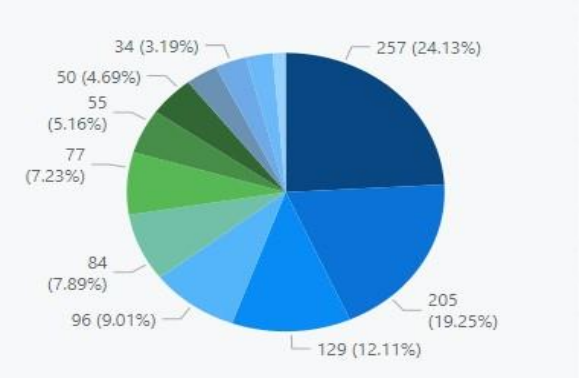


Figure 11 The industry with higher protein in the market

6.1.3. When the Co-founder should sell the company

According to the visualization, the best time for a co-founder to sell their unicorn company is when it has an average valuation of $3.5 billion and has experienced a 6% exit of investors. exit of investors. If the company experiences a higher rate of investor exit, it may be at risk of being acquired or forced to close.

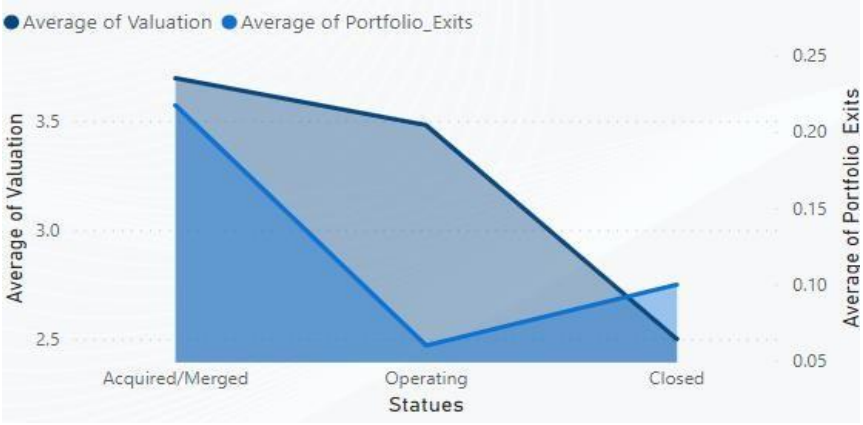


Figure 12 when the co-founder should sell the company

6.2. ML algorithm

The table below shows the accuracy, F1-score, and ROC AUC score for different models. The KNN model has the highest accuracy of 95%, ROC of 95%, and F1-score of 95%.

To ensure the best model, I apply SMOTE (Synthetic Minority Over-sampling Technique), which is an oversampling method used to address class imbalance in a classification dataset. Instead of simply duplicating existing samples, SMOTE generates synthetic samples of the minority class by selecting a minority class sample and one of its nearest neighbors, then interpolating new synthetic samples along the line between the two samples, choosing a random number of steps between them. These synthetic samples are added to the original dataset, which can help improve the performance of classification algorithms when the original dataset is imbalanced.

ML	Accuracy	ROC	F1-score
RidgeClassifier	0.94	94.5 %	0.94
KNN	0.95	95.0 %	0.95
LDA	0.94	94.5 %	0.94
LR	0.93	93.0 %	0.92
NC	0.70	68.9 %	0.64
SVC	0.97	96.9	0.97

Table 4 ML algorithm score

By using SMOTE, I observed that all the models increased in accuracy, ROC, and F1-score, except for the NC model.

The following figures compare the confusion matrices of the six algorithms I used after applying SMOTE.

The results show that the SVC classifier had the highest accuracy, with a test sample accuracy of 97% and a train sample accuracy of 97%. The confusion matrix heatmap for the SVC classifier, using a test sample of 25% (totaling 1635 out of 2180 total samples).

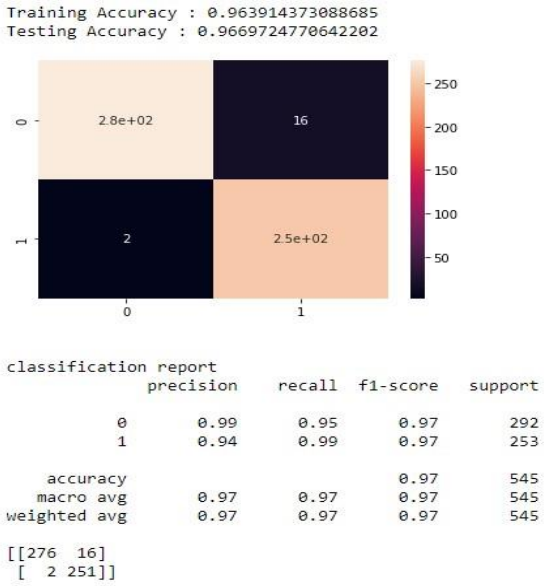


Figure 13 SVC confusion matrix

The results show that the KNN classifier had the second highest accuracy, with a test sample accuracy of 95% and a train sample accuracy of 95%. The confusion matrix heatmap for the KNN classifier, using a test sample of 25% (totaling 1635 out of 2180 total samples)

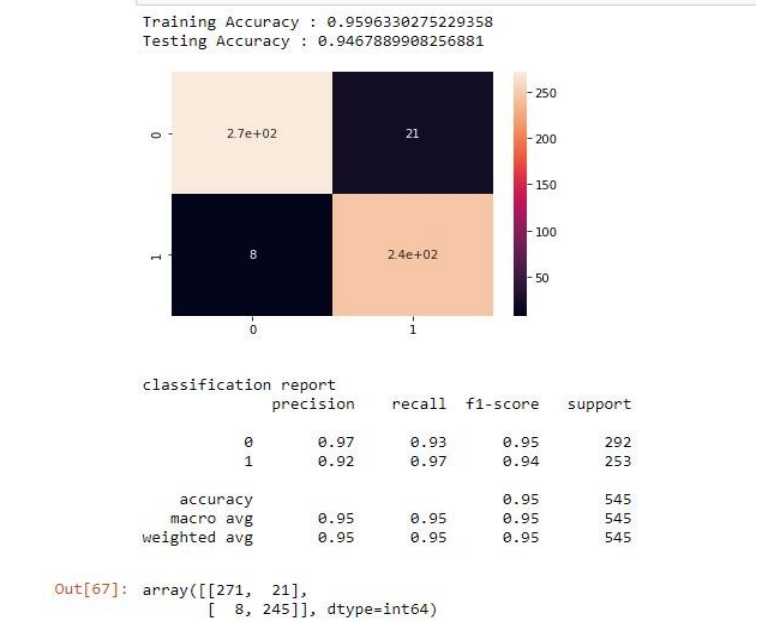


Figure 14 KNN confusion matrix

The third matrix was for the RidgeClassifier and LDA algorithms, which had a test sample accuracy of 94.5% and a train sample accuracy of 94%. The confusion matrix heatmap for these two models, using a test sample of 25% (totaling 1635 out of 2180 total samples)

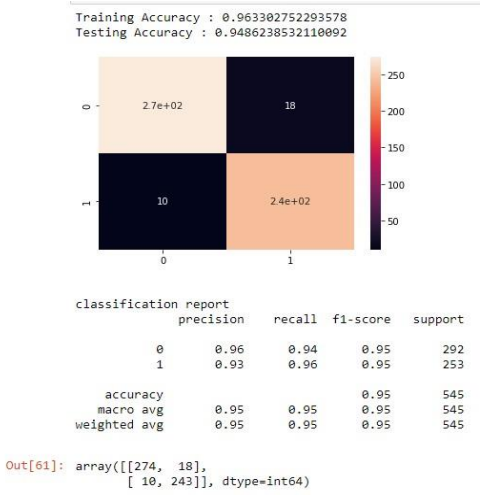


Figure 15 RidgeClassifier confusion matrix

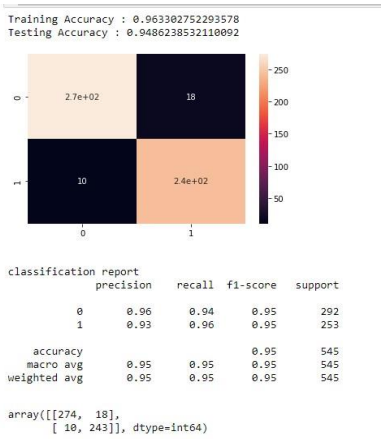


Figure 16 LDA confusion matrix

The LR algorithm had a test sample accuracy of 93% and a train sample accuracy of 93%. The confusion matrix heatmap for this model, using a test sample of 25% (totaling 1635 out of 2180 total samples), shows that the classifier is correct 93% of the time.

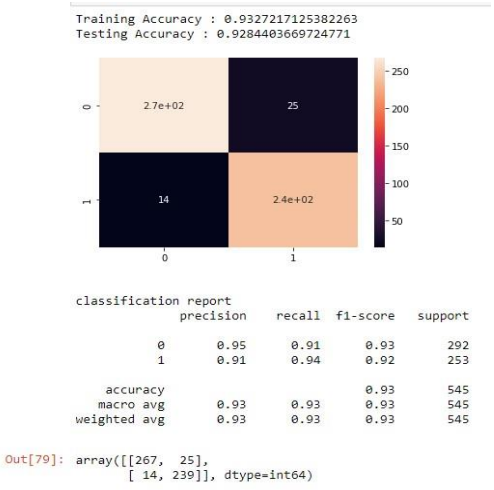


Figure 17 LR confusion matrix

The NC algorithm had a test sample accuracy of 70% and a train sample accuracy of 72%. The confusion matrix heatmap for this model, using a test sample of 25% (totaling 1635 out of 2180 total samples).

Figure 18 NC confusion matrix

6.3. Champion Model

After careful analysis and evaluation, I have determined that the SVC classification model is the most effective for predicting unicorn success. This decision was based on its performance metrics, which were found to be the highest among the models tested. Specifically, the accuracy score of the SVC model increased to 97% after applying the SMOTE method, which is a technique for dealing with class imbalance in the dataset. Additionally, both the ROC and F1-score of the SVC model achieved a score of 97%, which are widely accepted performance metrics that indicate the model's ability to correctly classify positive and negative cases.

These outstanding results demonstrate that the SVC model is capable of effectively identifying patterns in the data and making accurate predictions about unicorn success. Furthermore, its high accuracy, combined with its robustness to class imbalance, make it the ideal model for this task. The SVC model was also found to be more robust to variations in the size and complexity of the dataset, which highlights its ability to generalize well to new data. Overall, these impressive results clearly establish the SVC model as the best choice for predicting unicorn success, and further research can be done on how to optimize this model for better results.

6.4. Discussion

In this research, I set out to predict the success of unicorn companies using machine learning techniques. The first step in this process was to identify and gather relevant data on unicorn companies from CB Insights, using both python and manual scraping. This data included financial statements, investor count, and similar companies.

Once the data was collected, I then cleaned and pre-processed it for analysis and modeling. During this step, I discovered that the dataset was imbalanced, with many negative cases and a small number of positive cases. To overcome this issue, I applied the SMOTE (Synthetic Minority Over-sampling Technique) method to balance the dataset and ensure that the models had an equal number of positive and negative cases to learn from.

Next, I analyzed unicorn behavior based on several variables and determined which variables had the greatest impact on unicorn success. I also developed and tested a range of supervised machine learning models, including RidgeClassifier, KNN, LDA, LR, NC, and SVC. These models were chosen based on their effectiveness and popularity in the field of machine learning.

Finally, I evaluated the performance of the models using metrics such as accuracy, precision, and recall. After careful analysis, I found that the K-Nearest Neighbors (KNN) model had the highest performance among the models tested, with an accuracy score of 97%. The model also had a high ROC score and F1-score which indicate the model's ability to correctly classify positive and negative cases.

7. Conclusion

Accurate prediction of unicorn success is crucial, as the consequences of making incorrect decisions about which unicorn will be successful can be costly in terms of both money and opportunities. To address this challenge, this paper presents an empirical analysis of various machine learning algorithms. The

sample for this study was created by web scraping data from a crowd-sourced database called CBinsight.com. While the sample for this study is larger than many other studies on this topic, it is also subject to selection bias, as most of the firms in the sample are successful.

To address the class imbalance between successful and failed companies, I applied oversampling to the minority class (failed companies) using the SMOTE method. Six different models were tested: LR, SVC, KNN, RidgeClassifier, LDA, and NC. I found that the KNN model performed the best across a range of metrics and outperformed the NC model in terms of predictive ability.

7.1. Limitation

There were limitations in this study.

Firstly, the dataset was imbalanced, with a significantly larger number of observations in the majority class compared to the minority class. This made it more challenging for the models to accurately predict the minority class, as it was harder for them to learn the characteristics of the minority class samples. However, I applied the SMOTE method to address this issue, and there are other techniques that could have been used to handle imbalanced datasets that were not explored in this study.

Secondly, because the idea is new, there was no perfect data available, so I had to collect the data from CB Insights. This was a time-consuming process as much of the data was unstructured, which may have introduced some bias or limitations to the study.

Thirdly, our use of logistic regression in this study may not have been the most suitable choice due to its limited predictive ability. However, I chose to utilize logistic regression because our focus was on establishing a strong theoretical foundation and creating a transparent and easily understandable machine learning model.

fourthly, while the focus of this research was on determining the potential for machine learning techniques to predict the success of unicorn, it is important to note that success can be difficult to define and measure. In this study, I chose to focus on specific metrics that are commonly used to assess the success of unicorn, such as industry, status, and valuation. However, there are many other factors that can contribute to the overall performance and success of a unicorn, such as customer satisfaction, employee satisfaction, and innovation. These factors are also important to consider when evaluating the success of a unicorn, and future research may benefit from incorporating these additional measures of performance into the analysis.

7.2. Recommendations

The ever-changing nature of the unicorn company industry necessitates the periodic replication of this study using the most up-to-date data available to maintain the accuracy of these predictions in realworld situations. Incorporating the model into a system that can process live data will also allow for realtime prediction capabilities.

In future iterations of this study, it may be beneficial to investigate other methods for dealing with imbalanced datasets, such as modifying existing machine learning algorithms or using alternative resampling techniques like random under-sampling and over-sampling. Although this study testing additional methods may improve the model's ability to handle imbalanced classification issues. It would also be useful to evaluate the model using various performance evaluation metrics to better understand its performance and identify areas for improvement.

using classification algorithms to predict the success of unicorn, is to carefully select and pre-process the data used for training and testing the model. Ensuring that the data is representative of the target and free from errors and biases can significantly improve the accuracy and reliability of the model's predictions.

Should work to further develop and refine the theoretical framework outlined in this research, as this will allow for a more structured and systematic approach to the topic. Additionally, it will be valuable to gather more and more comprehensive data on unicorn, which can be achieved by accessing a variety of databases and combining them. Furthermore, the use of advanced machine learning techniques on this data could lead to more accurate predictions.

It would also be beneficial to combine both performance indicators and success metrics to generate more complete forecasts of unicorn success. The research presented here provides a foundation for analyzing the performance of unicorn and serves as a steppingstone towards identifying key factors that contribute to unicorn success. This research has laid the foundation for a theoretical framework for analyzing unicorn performance, and future studies will bring us closer to identifying the key predictors of unicorn success. This research has provided a valuable starting point for understanding how to predict unicorn performance, and there is significant potential for future research to build upon this work and

make even more significant strides in this area. By continuing to develop and refine our understanding of the factors that influence unicorn success, I can move closer to the goal of identifying the next unicorn.

Overall, using classification algorithms to predict the success of unicorn has the potential to be a valuable tool for identifying and investing in the next generation of unicorns. By following best practices and using a combination of techniques and approaches, it is possible to achieve reliable and accurate predictions of unicorn success, this study provides a useful approach for predicting the success of unicorn firms using machine learning techniques and highlights the importance of addressing class imbalances in such analyses.

7.3. Beneficial buyers

Using machine learning to predict the success of unicorn companies can be highly beneficial for both co-founders and investors. For co-founders, machine learning can help identify key factors that contribute to the success of unicorn companies, allowing them to make more informed decisions and increase the chances of success for their own ventures. This can include identifying trends in financial performance, investor interest, and similar companies, as well as understanding the competitive landscape.

For investors, machine learning can be used to identify and evaluate potential investment opportunities, allowing them to make more informed decisions about where to invest their capital. By analyzing the data of successful unicorn companies and developing models that can predict success, investors can identify high-growth industries and companies that are most likely to yield a strong return on investment. This can help them to mitigate risk and make more profitable investments. Also, using ML can also help both co-founders and investors to identify and overcome class imbalance in the data. By using techniques such as SMOTE, it can help to balance the data and increase the accuracy of predictions, which can help to identify opportunities that may have been overlooked.

References

Arroyo, J., Corea, F., Jimenez-Diaz, G., & Recio-Garcia, J. A. (2019). IEEE Access. *Assessment of machine learning performance for decision support in venture capital investments*, 124233 - 124243.

cbinsights. (2022). *cbinsights*. Retrieved from cbinsights: https://www.cbinsights.com/research-unicorncompanies

Chang, C., Liu, N., Yao, L., & Zhao, X. (2022). A semi-supervised classification RBM with an improved fMRI representation algorithm. *Computer Methods and Programs in Biomedicine*.

Chauhan, P. S., Kshetri, N., & Ahmad, N. (2022). The Role of Data and Artificial Intelligence in Driving Diversity, Equity, and Inclusion. *Computer*.

Chen, J., Sun, B., Wang, L., Fang, B., Chang, Y., Li, Y., . . . Chen, G. (2022). Semi-supervised semantic segmentation framework with pseudo supervisions for land-use/land-cover mapping in coastal areas. *International Journal of Applied Earth Observation and Geoinformation*.

Czako, Z., Sebestyen, G., & Hangan, A. (2021). AutomaticAI – A hybrid approach for automatic artificial intelligence algorithm selection and hyperparameter tuning. *Expert Systems with Applications*.

Emir Hidayat, S., Bamahriz, O., Hidayati, N., Sari, C. A., & Dewandaru, G. (2022). Value drivers of startup valuation from venture capital equity-based investing: A global analysis with a focus on technological factors. *Borsa Istanbul Review*, 653 - 667.

Erol, G., Uzbaş, B., Yücelbaş, C., & Yücelbaş, Ş. (2022). Analyzing the effect of data preprocessing techniques using machine learning algorithms on the diagnosis of COVID-19. *Concurrency and Computation: Practice and Experience*.

Fujita, M., Okudo, T., Nishino, N., & Nagane, H. (2021). Analyzing startup ecosystem through corporate networks based on investment relation of venture capitals in unicorns. *Procedia CIRP*.

Goel, I., Goradia, S. R., & Kakelli, A. K. (2021). Predicting the Presence of Amphibians Near Road Construction Sites Using Emerging Machine Learning Algorithms. *Proceedings - 2021 1st IEEE International Conference on Artificial Intelligence and Machine Vision, AIMV 2021*.

Goschenhofer, J., Hvingelby, R., Ruegamer, D., Thomas, J., Wagner, M., & Bischl, B. (2022). Deep Semi-supervised Learning for Time Series Classification. *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*.

Grant, K., & Rahman, S. (2021). Assessing the participation and success of women entrepreneurs in unicorn startups. *Proceedings of the European Conference on Innovation and Entrepreneurship, ECIE*, 397 - 406.

Gulati, K., S., S. K., Sarath Kumar Boddu, R., Sarvakar, K., Kumar Sharma, D., & M.Z.M., N. (2021). Comparative analysis of machine learning-based classification models using sentiment classification of tweets related to COVID-19 pandemic. *Materials Today: Proceedings*.

Hacklin, F., Björkdahl, J., & Wallin, M. W. (2018). Strategies for business model innovation: How firms reel in migrating value. *Long Range Planning*, 82-110.

Hoang, D. B., & Hoang, S. (2022). DEEP LEARNING-CANCER GENETICS AND APPLICATION OF DEEP LEARNING TO CANCER ONCOLOGY. *Vietnam Journal of Science and Technology*, 885 - 928.

Hou, X., Xiong, X., Li, X., Bi, J., Xu, G., Wang, Y., & Jiang, S. (2022). Predictive value of cardiac magnetic resonance mechanical parameters for myocardial fibrosis in hypertrophic

cardiomyopathy with preserved left ventricular ejection fraction. *Frontiers in Cardiovascular Medicine*.

Jackson, C. M., & Adam, E. (2022). A machine learning approach to mapping canopy gaps in an indigenous tropical submontane forest using WorldView-3 multispectral satellite imagery. *Environmental Conservation*.

Jaddi, N. S., & Saniee Abadeh, M. (2022). Cell separation algorithm with enhanced search behaviour in miRNA feature selection for cancer diagnosis. *Information Systems*.

Kaloferov, G. (2021). Predicting Startup Success Using Support Vector Machine. *AIP Conference Proceedings*.

Kumar, A., Misra, S. C., & Chan, F. T. (2022). Leveraging AI for advanced analytics to forecast altered tourism industry parameters: A COVID-19 motivated study. *Expert Systems with Applications*.

Li, X., Liu, J., Dong, J., Lu, J., & Lu, L. (2021). Exploring Impact Factors of Risk Contagion in Venture Capital Markets: A Complex Network Approach. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 4268 - 4277.

Lichouri, M. A. (2021). Arabic Dialect Identification based on Weighted Concatenation of TF-IDF Features. *WANLP 2021 - 6th Arabic Natural Language Processing Workshop, Proceedings of the Workshop*, 282 - 286.

Malyy, M., Tekic, Z., & Podladchikova, T. (2021). The value of big data for analyzing growth dynamics of technology-based new ventures. *Technological Forecasting and Social Change*.

Menon, R., & James, L. (2022). Understanding Startup Valuation and its Impact on Startup Ecosystem. *Journal of Business Valuation and Economic Loss Analysis*, 101 - 114.

N.Irino(3), M. a. (2020). A vision-based machine accuracy measurement method. *CIRP Annals*.

Nguyen, T. K., Nguyen, T. N., Nguyen, H. V., Tran, L. T., Ngo, T. X., Pham, P. T., & Tran, M. H. (2022). Machine learning-based screening of MCF-7 human breast cancer cells and molecular docking analysis of essential oils from Ocimum basilicum against breast cancer. *Journal of Molecular Structure*.

Painuli, D., Bhardwaj, S., & köse, U. (2022). Recent advancement in cancer diagnosis using machine learning and deep learning techniques: A comprehensive review. *Computers in Biology and Medicine*.

Pinho-Gomes, A.-C., Yoo, S.-H., Allen, A., Maiden, H., Shah, K., & Toolan, M. (2022). Incorporating environmental and sustainability considerations into health technology assessment and clinical and public health guidelines: a scoping review. *International Journal of Technology Assessment in Health Care*.

Ren, G., Liu, L., Zhang, P., Xie, Z., Wang, P., Zhang, W., . . . Wang, J. (2022). Machine Learning Predicts Recurrent Lumbar Disc Herniation Following Percutaneous Endoscopic Lumbar Discectomy. *Global Spine Journal*.

Sharma, S., Guleria, K., Tiwari, S., & Kumar, S. (2022). A deep learning based convolutional neural network model with VGG16 feature extractor for the detection of Alzheimer Disease using MRI scans. *Measurement: Sensors*.

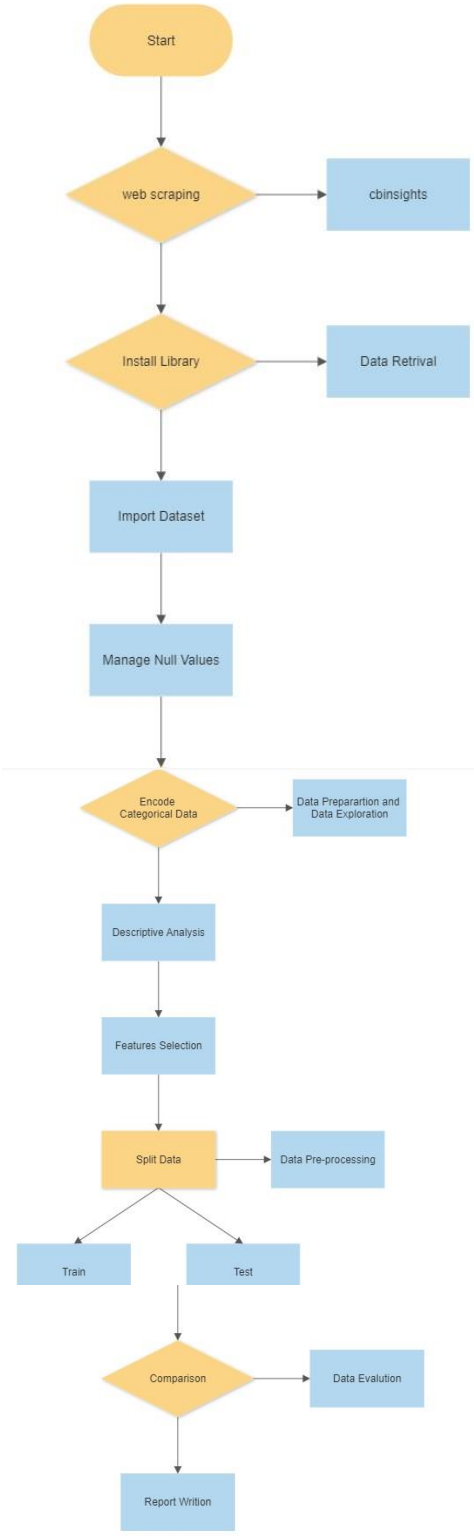
Singhal, J., Rane, C., Wadalkar, Y., Deshpande, A., & Joshi, M. (2022). Data Driven Analysis for Startup Investments for Venture Capitalists. *2022 International Conference for Advancement in Technology, ICONAT 2022*.

Suroso, J. S., Kaburuan, E. R., Setyo, J., & Jery. (2020). Identification of e-commerce business approaches to improve customer satisfaction. *2020 8th International Conference on Orange Technology, ICOT 2020*.

Torres-Toukoumidis, A. C. (2020). Gamified marketing in unicorn startups companies. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, 314 - 324.

Xi, L., Yun, Z., Liu, H., Wang, R., Huang, X., & Fan, H. (2022). Semi-supervised Time Series Classification Model with Self-supervised Learning. *Engineering Applications of Artificial Intelligence*.

Appendix


Start
web scraping
cbinsights
Install Library
Data Retrival
Import Dataset
Manage Null Values
Encode Categorical Data
Data Preparartion and Data Exploration
Descriptive Analysis
Features Selection
Split Data
Data Pre-processing
Train
Test
Comparison
Data Evalution
Report Writion

This appendix consists of the machine learning code, including the following steps:

1) importing necessary libraries,

2) importing the CSV file,

3) calculating probability,

4) performing univariate exploration,

5) encoding the data,

6) analyzing the distribution of features.

7) machine learning algorithm.

8) Power BI visualizations

1. Import necessary library

In [1]:
```
#Part I - Preliminary Wrangling
```

In [2]:
```
# import the library
import warnings
import pandas as pd
import numpy as np
import random
import seaborn as sns
import matplotlib.pyplot as plt
import squarify
warnings.filterwarnings('ignore')
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from pandas import Timestamp
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
from sklearn.metrics import roc_auc_score
from sklearn import preprocessing
from sklearn.metrics import classification_report
import pickle
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn import tree
```

```
from sklearn.model_selection import cross_val_score, KFold

from sklearn.linear_model import RidgeClassifier

from imblearn.over_sampling import SMOTE

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import NearestCentroid

from sklearn.svm import SVC
```

In [3]:
```
# import csv file
df = pd.read_csv('finalclean.csv', encoding='cp1252')
```

In [4]:
```
#to show all columns because the 22 column
pd.set_option('display.max_columns', None)
# viewing the first lines
df.head()
```

Out[4]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent
0	Bytedance	$180B	2017	4/7/2017	Artificial intelligence	Beijing	China	Asia
1	SpaceX	$100B	2012	12/1/2012	Other	Hawthorne	United States	North America
2	SHEIN	$100B	2018	7/3/2018	E-commerce & direct-to-consumer	Shenzhen	China	Asia
3	Stripe	$95B	2014	1/23/2014	Fintech	San Francisco	United States	North America
4	Klarna	$46B	2011	12/12/2011	Fintech	Stockholm	Sweden	Europe

2. Import CSV file

In [3]:
```
# import csv file
df = pd.read_csv('finalclean.csv', encoding='cp1252')
```

In [4]:
```
#to show all columns because the 22 column
pd.set_option('display.max_columns', None)
# viewing the first lines
df.head()
```

Out[4]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent
0	Bytedance	$180B	2017	4/7/2017	Artificial intelligence	Beijing	China	Asia
1	SpaceX	$100B	2012	12/1/2012	Other	Hawthorne	United States	North America
2	SHEIN	$100B	2018	7/3/2018	E-commerce & direct-to-consumer	Shenzhen	China	Asia
3	Stripe	$95B	2014	1/23/2014	Fintech	San Francisco	United States	North America
4	Klarna	$46B	2011	12/12/2011	Fintech	Stockholm	Sweden	Europe

3. calculating probability

In [6]:
```
df = df[df['Statues'] != 'Acquired/Merged']
```

In [7]:
```
df['Statues'].value_counts()
```

Out[7]:
```
Closed       1148
Operation    1032
Name: Statues, dtype: int64
```

In [8]:
```
#to know if the sum of the duplicate
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
#info on the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2180 entries, 0 to 2202
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Company                 2180 non-null   object
 1   Valuation               2180 non-null   object
 2   Date_Joined_Year        2180 non-null   int64
 3   Date_Joined             2180 non-null   object
 4   Industry                2180 non-null   object
 5   City                    2180 non-null   object
 6   Country                 2180 non-null   object
 7   Continent               2180 non-null   object
 8   Year_Founded            2180 non-null   int64
 9   Funding                 2180 non-null   object
 10  Select_Investors        2180 non-null   object
 11  Headquarters            2180 non-null   object
 12  Statues                 2180 non-null   object
 13  last_Funding_Type       2180 non-null   object
 14  Exit_Investor_Stage     2180 non-null   object
 15  Investors_Count         2180 non-null   int64
 16  Deal                    2180 non-null   int64
 17  Portfolio_Exits         2180 non-null   int64
 18  Funding_Time_Year       2180 non-null   int64
 19  Funding_Time            2180 non-null   object
 20  Acq                     2180 non-null   int64
 21  Number_Of_Funding_Round 2180 non-null   int64
 22  Compatitor              2180 non-null   int64
 23  Employee_Number         2180 non-null   int64
 24  StatueS                 2180 non-null   int64
dtypes: int64(11), object(14)
memory usage: 442.8+ KB
```

In [10]:
```
#to know the number of unique values in the data in each row
df.nunique()
```

Out[10]:
```
Company                   2173
Valuation                   29
Date_Joined_Year            12
Date_Joined                632
Industry                    15
City                       253
Country                     46
Continent                    6
Year_Founded                35
Funding                    541
Select_Investors          1045
Headquarters               253
Statues                      2
last_Funding_Type          162
Exit_Investor_Stage         12
Investors_Count             55
Deal                        16
Portfolio_Exits              5
Funding_Time_Year           12
Funding_Time               564
Acq                         22
Number_Of_Funding_Round     26
Compatitor                 121
Employee_Number             10
StatueS                      3
dtype: int64
```

In [11]:
```
df.value_counts()
```

Out[11]:
```
Company              Valuation  Date_Joined_Year  Date_Joined  Industry
City           Country        Continent      Year_Founded  Funding  Select_Inve
stors                                                    Headquarters   Statues
last_Funding_Type       Exit_Investor_Stage  Investors_Count  Deal  Portfolio_E
xits  Funding_Time_Year  Funding_Time  Acq  Number_Of_Funding_Round  Compatitor
Employee_Number  StatueS
#waywire             $4B        2021              1/7/2021     Internet softwar
e & services     Mississauga    Canada         North America  2000          $23
0M    Dragoneer Investment Group, Hellman & Friedman, JMI Equity         Missis
sauga    Closed     Private Equity | Alive  0                    3
1     0               2022              4/1/2022      5    5
16          5000             2          1
FullStory            $2B        2021              8/4/2021     Internet softwar
e & services     Atlanta        United States  North America  2014          $17
```

In [12]:
```
# to know the sum of the NAN value
df.isnull().sum()
```

Out[12]:
```
Company                  0
Valuation                0
Date_Joined_Year         0
Date_Joined              0
Industry                 0
City                     0
Country                  0
Continent                0
Year_Founded             0
Funding                  0
Select_Investors         0
Headquarters             0
Statues                  0
last_Funding_Type        0
Exit_Investor_Stage      0
Investors_Count          0
Deal                     0
Portfolio_Exits          0
Funding_Time_Year        0
Funding_Time             0
Acq                      0
Number_Of_Funding_Round  0
Compatitor               0
Employee_Number          0
StatueS                  0
dtype: int64
```

In [13]:
```
# data quantative describe
df.describe()
```

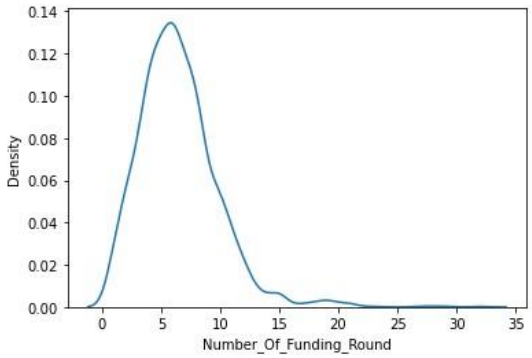
Out[13]:

	Date_Joined_Year	Year_Founded	Investors_Count	Deal	Portfolio_Exits	Funding_Tin
count	2180.000000	2180.000000	2180.000000	2180.000000	2180.000000	2180
mean	2020.011009	2013.032110	13.919266	2.811468	0.048624	2020
std	1.922516	5.185045	9.193144	1.979880	0.289677	1
min	2011.000000	1919.000000	0.000000	0.000000	0.000000	2008
25%	2019.000000	2011.000000	8.000000	1.000000	0.000000	2021
50%	2021.000000	2014.000000	12.000000	2.000000	0.000000	2021
75%	2021.000000	2016.000000	18.000000	4.000000	0.000000	2022
max	2022.000000	2021.000000	91.000000	19.000000	5.000000	2022

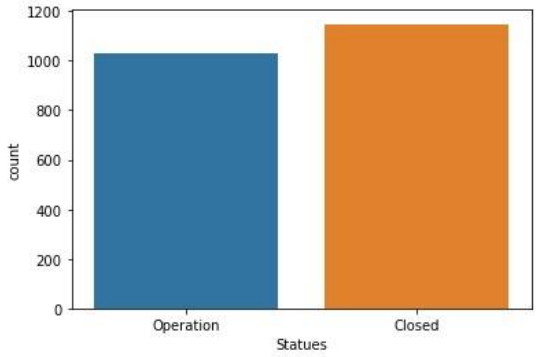
```
In [14]: df.shape
Out[14]: (2180, 25)

In [15]: df.columns
Out[15]: Index(['Company', 'Valuation', 'Date_Joined_Year', 'Date_Joined', 'Industry',
                'City', 'Country', 'Continent', 'Year_Founded', 'Funding',
                'Select_Investors', 'Headquarters', 'Statues', 'last_Funding_Type',
                'Exit_Investor_Stage', 'Investors_Count', 'Deal', 'Portfolio_Exits',
                'Funding_Time_Year', 'Funding_Time', 'Acq', 'Number_Of_Funding_Round',
                'Compatitor', 'Employee_Number', 'StatueS'],
               dtype='object')

In [16]: df.to_csv('finalcleanencoding.csv', index=False)
```

4. Univariate Exploration

```
In [18]: sns.kdeplot(data=df, x='Number_Of_Funding_Round')
Out[18]: <AxesSubplot:xlabel='Number_Of_Funding_Round', ylabel='Density'>
```

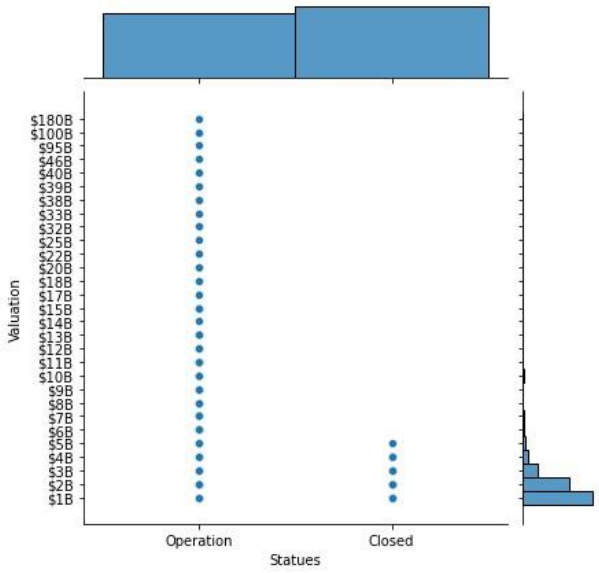



```
In [19]: sns.countplot(df['Statues'])
Out[19]: <AxesSubplot:xlabel='Statues', ylabel='count'>
```




```
In [20]: # the private company is the highest precentage
```

```
In [20]: # the private company is the highest precentage

In [21]: sns.jointplot(x='Statues', y='Valuation', data=df)

Out[21]: <seaborn.axisgrid.JointGrid at 0x2276cdf3e80>
```




```
In [22]: # there a positive relationship the private had a highest valuion and the closed

In [23]: Date_Joined_Year = df["Date_Joined_Year"]
         Year_Founded = df["Year_Founded"]
         df['Become_Unicorn']= Date_Joined_Year - Year_Founded

In [24]: df.to_csv('finalcleanencoding111.csv', index=False)
```

5. Encoding data

In [28]:
```
label_encoder = preprocessing.LabelEncoder()

df['Valuation']= label_encoder.fit_transform(df['Valuation'])

df['Valuation'].unique()

df.head()
```

Out[28]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent
0	Bytedance	8	2017	4/7/2017	Artificial intelligence	Beijing	China	Asia
1	SpaceX	0	2012	12/1/2012	Other	Hawthorne	United States	North America
2	SHEIN	0	2018	7/3/2018	E-commerce & direct-to-consumer	Shenzhen	China	Asia
3	Stripe	27	2014	1/23/2014	Fintech	San Francisco	United States	North America
4	Klarna	21	2011	12/12/2011	Fintech	Stockholm	Sweden	Europe

In [29]:
```
label_encoder = preprocessing.LabelEncoder()

df['Industry']= label_encoder.fit_transform(df['Industry'])

df['Industry'].unique()

df.head()
```

Out[29]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent

In [29]:
```
label_encoder = preprocessing.LabelEncoder()

df['Industry']= label_encoder.fit_transform(df['Industry'])

df['Industry'].unique()

df.head()
```

Out[29]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Y
0	Bytedance	8	2017	4/7/2017	0	Beijing	China	Asia	
1	SpaceX	0	2012	12/1/2012	12	Hawthorne	United States	North America	
2	SHEIN	0	2018	7/3/2018	5	Shenzhen	China	Asia	
3	Stripe	27	2014	1/23/2014	7	San Francisco	United States	North America	
4	Klarna	21	2011	12/12/2011	7	Stockholm	Sweden	Europe	

In [30]:
```
label_encoder = preprocessing.LabelEncoder()

df['Funding']= label_encoder.fit_transform(df['Funding'])

df['Funding'].unique()

df.head()
```

Out[30]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Y
0	Bytedance	8	2017	4/7/2017	0	Beijing	China	Asia	
1	SpaceX	0	2012	12/1/2012	12	Hawthorne	United States	North America	

In [30]:
```
label_encoder = preprocessing.LabelEncoder()

df['Funding']= label_encoder.fit_transform(df['Funding'])

df['Funding'].unique()

df.head()
```

Out[30]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Y
0	Bytedance	8	2017	4/7/2017	0	Beijing	China	Asia	
1	SpaceX	0	2012	12/1/2012	12	Hawthorne	United States	North America	
2	SHEIN	0	2018	7/3/2018	5	Shenzhen	China	Asia	
3	Stripe	27	2014	1/23/2014	7	San Francisco	United States	North America	
4	Klarna	21	2011	12/12/2011	7	Stockholm	Sweden	Europe	

In [31]:
```
label_encoder = preprocessing.LabelEncoder()

df['City']= label_encoder.fit_transform(df['City'])

df['City'].unique()

df.head()
```

Out[31]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	China	Asia	
1	SpaceX	0	2012	12/1/2012	12	96	United States	North America	

In [31]:
```
label_encoder = preprocessing.LabelEncoder()

df['City']= label_encoder.fit_transform(df['City'])

df['City'].unique()

df.head()
```

Out[31]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	China	Asia	
1	SpaceX	0	2012	12/1/2012	12	96	United States	North America	
2	SHEIN	0	2018	7/3/2018	5	213	China	Asia	
3	Stripe	27	2014	1/23/2014	7	197	United States	North America	
4	Klarna	21	2011	12/12/2011	7	220	Sweden	Europe	

In [32]:
```
label_encoder = preprocessing.LabelEncoder()

df['Country']= label_encoder.fit_transform(df['Country'])

df['Country'].unique()

df.head()
```

Out[32]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	Asia	
1	SpaceX	0	2012	12/1/2012	12	96	44	North America	

In [32]:
```
label_encoder = preprocessing.LabelEncoder()
df['Country']= label_encoder.fit_transform(df['Country'])
df['Country'].unique()
df.head()
```

Out[32]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	Asia	
1	SpaceX	0	2012	12/1/2012	12	96	44	North America	
2	SHEIN	0	2018	7/3/2018	5	213	9	Asia	
3	Stripe	27	2014	1/23/2014	7	197	44	North America	
4	Klarna	21	2011	12/12/2011	7	220	38	Europe	

In [33]:
```
label_encoder = preprocessing.LabelEncoder()
df['Continent']= label_encoder.fit_transform(df['Continent'])
df['Continent'].unique()
df.head()
```

Out[33]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	

In [34]:
```
label_encoder = preprocessing.LabelEncoder()

df['Select_Investors']= label_encoder.fit_transform(df['Select_Investors'])

df['Select_Investors'].unique()

df.head()
```

Out[34]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [35]:
```
label_encoder = preprocessing.LabelEncoder()

df['Statues']= label_encoder.fit_transform(df['Statues'])

df['Statues'].unique()

df.head()
```

Out[35]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

```
df['last_Funding_Type']= label_encoder.fit_transform(df['last_Funding_Type'])

df['last_Funding_Type'].unique()

df.head()
```

Out[36]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [37]:
```
label_encoder = preprocessing.LabelEncoder()

df['Exit_Investor_Stage']= label_encoder.fit_transform(df['Exit_Investor_Stage']

df['Exit_Investor_Stage'].unique()

df.head()
```

Out[37]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [38]:
```
label_encoder = preprocessing.LabelEncoder()

df['Headquarters']= label_encoder.fit_transform(df['Headquarters'])

df['Headquarters'].unique()

df.head()
```

Out[38]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F

In [38]:
```
label_encoder = preprocessing.LabelEncoder()

df['Headquarters']= label_encoder.fit_transform(df['Headquarters'])

df['Headquarters'].unique()

df.head()
```

Out[38]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [39]:
```
label_encoder = preprocessing.LabelEncoder()

df['Deal']= label_encoder.fit_transform(df['Deal'])

df['Deal'].unique()

df.head()
```

Out[39]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [40]:
```
label_encoder = preprocessing.LabelEncoder()

df['Portfolio_Exits']= label_encoder.fit_transform(df['Portfolio_Exits'])

df['Portfolio_Exits'].unique()

df.head()
```

In [40]:
```
label_encoder = preprocessing.LabelEncoder()

df['Portfolio_Exits']= label_encoder.fit_transform(df['Portfolio_Exits'])

df['Portfolio_Exits'].unique()

df.head()
```

Out[40]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [41]:
```
label_encoder = preprocessing.LabelEncoder()

df['Compatitor']= label_encoder.fit_transform(df['Compatitor'])

df['Compatitor'].unique()

df.head()
```

Out[41]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	Bytedance	8	2017	4/7/2017	0	14	9	1	
1	SpaceX	0	2012	12/1/2012	12	96	44	3	
2	SHEIN	0	2018	7/3/2018	5	213	9	1	
3	Stripe	27	2014	1/23/2014	7	197	44	3	
4	Klarna	21	2011	12/12/2011	7	220	38	2	

In [42]:
```
label_encoder = preprocessing.LabelEncoder()

df['Company']= label_encoder.fit_transform(df['Company'])

df['Company'].unique()

df.head()
```

Out[42]:

In [42]:
```
label_encoder = preprocessing.LabelEncoder()

df['Company']= label_encoder.fit_transform(df['Company'])

df['Company'].unique()

df.head()
```

Out[42]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	2017	4/7/2017	0	14	9	1	
1	1932	0	2012	12/1/2012	12	96	44	3	
2	1870	0	2018	7/3/2018	5	213	9	1	
3	1950	27	2014	1/23/2014	7	197	44	3	
4	1594	21	2011	12/12/2011	7	220	38	2	

In [43]:
```
label_encoder = preprocessing.LabelEncoder()

df['Date_Joined_Year']= label_encoder.fit_transform(df['Date_Joined_Year'])

df['Date_Joined_Year'].unique()

df.head()
```

Out[43]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	6	4/7/2017	0	14	9	1	
1	1932	0	1	12/1/2012	12	96	44	3	
2	1870	0	7	7/3/2018	5	213	9	1	
3	1950	27	3	1/23/2014	7	197	44	3	
4	1594	21	0	12/12/2011	7	220	38	2	

In [44]:
```
label_encoder = preprocessing.LabelEncoder()

df['Funding_Time_Year']= label_encoder.fit_transform(df['Funding_Time_Year'])
```

In [44]:
```
label_encoder = preprocessing.LabelEncoder()

df['Funding_Time_Year']= label_encoder.fit_transform(df['Funding_Time_Year'])

df['Funding_Time_Year'].unique()

df.head()
```

Out[44]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	6	4/7/2017	0	14	9	1	
1	1932	0	1	12/1/2012	12	96	44	3	
2	1870	0	7	7/3/2018	5	213	9	1	
3	1950	27	3	1/23/2014	7	197	44	3	
4	1594	21	0	12/12/2011	7	220	38	2	

In [45]:
```
label_encoder = preprocessing.LabelEncoder()

df['Year_Founded']= label_encoder.fit_transform(df['Year_Founded'])

df['Year_Founded'].unique()

df.head()
```

Out[45]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	6	4/7/2017	0	14	9	1	
1	1932	0	1	12/1/2012	12	96	44	3	
2	1870	0	7	7/3/2018	5	213	9	1	
3	1950	27	3	1/23/2014	7	197	44	3	
4	1594	21	0	12/12/2011	7	220	38	2	

In [46]:
```
label_encoder = preprocessing.LabelEncoder()

df['Date_Joined']= label_encoder.fit_transform(df['Date_Joined'])

df['Date_Joined'].unique()

df.head()
```

Out[46]:

In [46]:
```
label_encoder = preprocessing.LabelEncoder()

df['Date_Joined']= label_encoder.fit_transform(df['Date_Joined'])

df['Date_Joined'].unique()

df.head()
```

Out[46]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	6	373	0	14	9	1	
1	1932	0	1	171	12	96	44	3	
2	1870	0	7	518	5	213	9	1	
3	1950	27	3	25	7	197	44	3	
4	1594	21	0	179	7	220	38	2	

In [47]:
```
label_encoder = preprocessing.LabelEncoder()

df['Funding_Time']= label_encoder.fit_transform(df['Funding_Time'])

df['Funding_Time'].unique()

df.head()
```

Out[47]:

	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Country	Continent	Year_F
0	1240	8	6	373	0	14	9	1	
1	1932	0	1	171	12	96	44	3	
2	1870	0	7	518	5	213	9	1	
3	1950	27	3	25	7	197	44	3	
4	1594	21	0	179	7	220	38	2	

In [48]:
```
df.describe()
```

Out[48]:

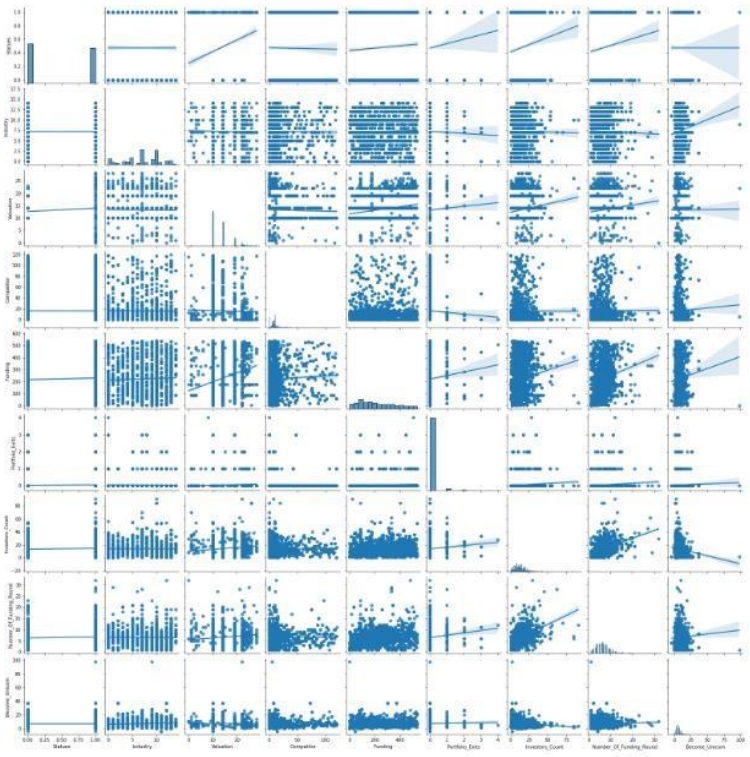
	Company	Valuation	Date_Joined_Year	Date_Joined	Industry	City	Co
count	2180.000000	2180.000000	2180.000000	2180.000000	2180.000000	2180.000000	2180.0
mean	1085.033028	13.242661	9.011009	311.365596	7.197248	137.614220	31.4

```
In [49]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2180 entries, 0 to 2202
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Company                 2180 non-null   int32
 1   Valuation               2180 non-null   int32
 2   Date_Joined_Year        2180 non-null   int64
 3   Date_Joined             2180 non-null   int32
 4   Industry                2180 non-null   int32
 5   City                    2180 non-null   int32
 6   Country                 2180 non-null   int32
 7   Continent               2180 non-null   int32
 8   Year_Founded            2180 non-null   int64
 9   Funding                 2180 non-null   int32
 10  Select_Investors        2180 non-null   int32
 11  Headquarters            2180 non-null   int32
 12  Statues                 2180 non-null   int32
 13  last_Funding_Type       2180 non-null   int32
 14  Exit_Investor_Stage     2180 non-null   int32
 15  Investors_Count         2180 non-null   int64
 16  Deal                    2180 non-null   int64
 17  Portfolio_Exits         2180 non-null   int64
 18  Funding_Time_Year       2180 non-null   int64
 19  Funding_Time            2180 non-null   int32
 20  Acq                     2180 non-null   int64
 21  Number_Of_Funding_Round 2180 non-null   int64
 22  Compatitor              2180 non-null   int64
 23  Employee_Number         2180 non-null   int64
 24  StatueS                 2180 non-null   int64
 25  Become_Unicorn          2180 non-null   int64
dtypes: int32(14), int64(12)
memory usage: 405.2 KB

In [50]: df.to_csv('finalcleanencoding.csv', index=False)

In [51]: sns.pairplot(df[["Statues", "Industry","Valuation","Compatitor","Funding","Portf

Out[51]: <seaborn.axisgrid.PairGrid at 0x2276dffca60>
```

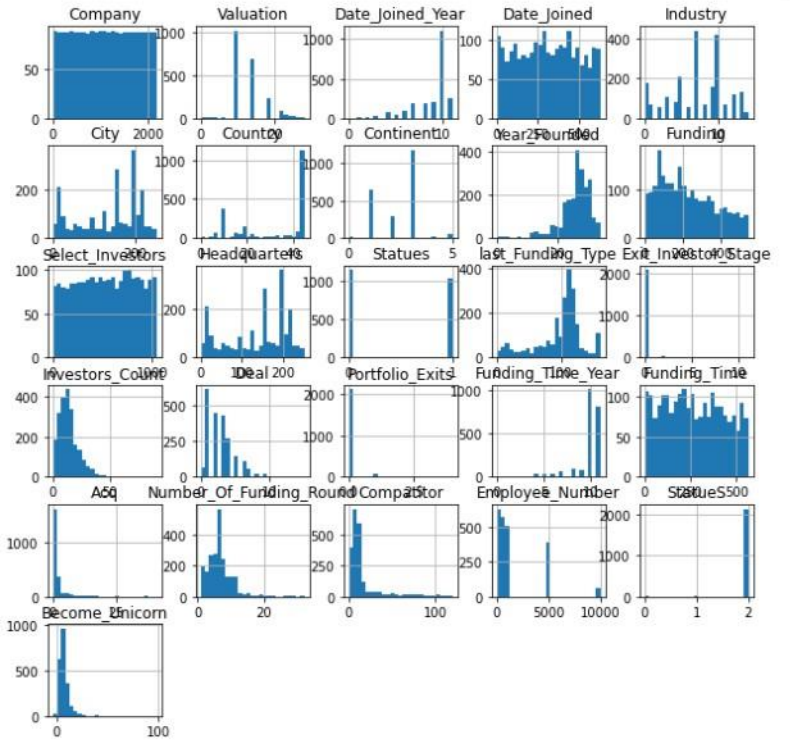

```
In [50]: df.to_csv('finalcleanencoding.csv', index=False)
```


6. Feature distribution

Out[51]: <seaborn.axisgrid.PairGrid at 0x2276dffca60>



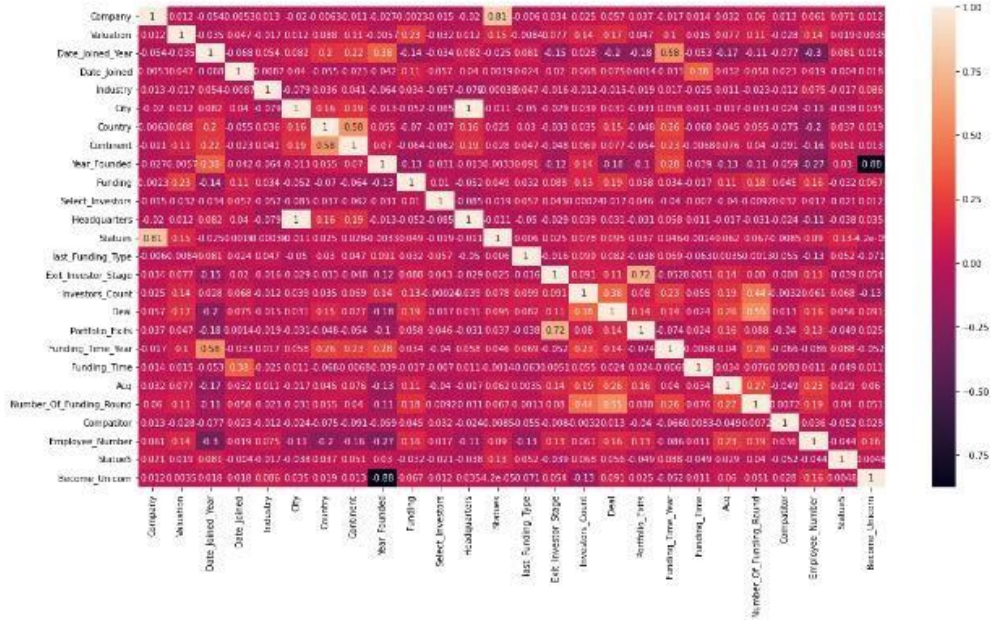
In [52]:
```
df.hist(bins=25, figsize=(10,10))
plt.show()
```



7. Machine learning

In [53]:
```
plt.figure(figsize=[19,10],facecolor='white')
sns.heatmap(df.corr(),annot=True)
```

Out[53]: <AxesSubplot:>



In [56]:
```
X = df.drop(["Statues"],axis = 1 )
y = df['Statues']
```

In [57]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, shuffl
```

In [58]:
```
# lets print the shapes again
print("Shape of the X Train :", X_train.shape)
print("Shape of the y Train :", y_train.shape)
print("Shape of the X test :", X_test.shape)
print("Shape of the y test :", y_test.shape)
```

```
Shape of the X Train : (1635, 25)
Shape of the y Train : (1635,)
Shape of the X test : (545, 25)
Shape of the y test : (545,)
```

RidgeClassifier Algorithm first model (perfect)

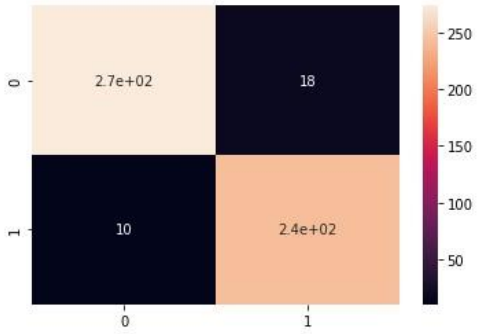
In [96]:
```
sm = SMOTE()
xres,yres = sm.fit_resample(X_train, y_train)
```

In [60]:
```
Ridge = RidgeClassifier()
Ridge.fit(xres,yres)
validation = cross_val_score(Ridge, X_train, y_train)

print("cross-validation mean score: %.2f" % validation.mean())
kfold = cross_val_score(Ridge, X_train, y_train)

print("average score kfold: %.2f" % kfold.mean())
#Predicting on test
prediction=Ridge.predict(X_test)
```

```
cross-validation mean score: 0.96
average score kfold: 0.96
```

In [61]:
```
print("Training Accuracy :", Ridge.score(X_train, y_train))
print("Testing Accuracy :", Ridge.score(X_test, y_test))

confusion = confusion_matrix(y_test, prediction)

sns.heatmap(confusion, annot = True)
plt.show()
print('\nclassification report')

#Predicting on test
print(classification_report(y_test, prediction))
confusion_matrix(y_test, prediction)
```

```
Training Accuracy : 0.963302752293578
Testing Accuracy : 0.9486238532110092
```




```
classification report
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       292
           1       0.93      0.96      0.95       253

    accuracy                           0.95       545
   macro avg       0.95      0.95      0.95       545
weighted avg       0.95      0.95      0.95       545
```

Out[61]:
```
array([[274,  18],
       [ 10, 243]], dtype=int64)
```

In [62]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,predictio
roc_auc = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc)

precision, recall, thresholds = precision_recall_curve(y_test, prediction)
f1 = f1_score(y_test, prediction)
Precision_Recall_rfs = auc(recall, precision)
print("Precision-Recall Curves =",Precision_Recall_rfs)
```

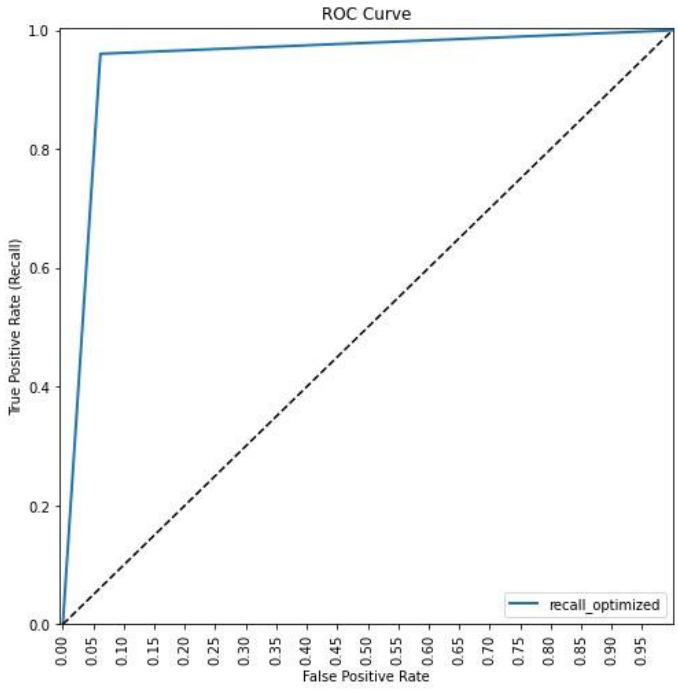
```
ROC Curves= 0.9494152363419784
Precision-Recall Curves = 0.9549287074561135
```

In [63]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, prediction
print('Random Forest: Recall = ',str(round(recall_score(y_test, prediction,pos_l
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, prediction)*
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, prediction,pos_lab
```

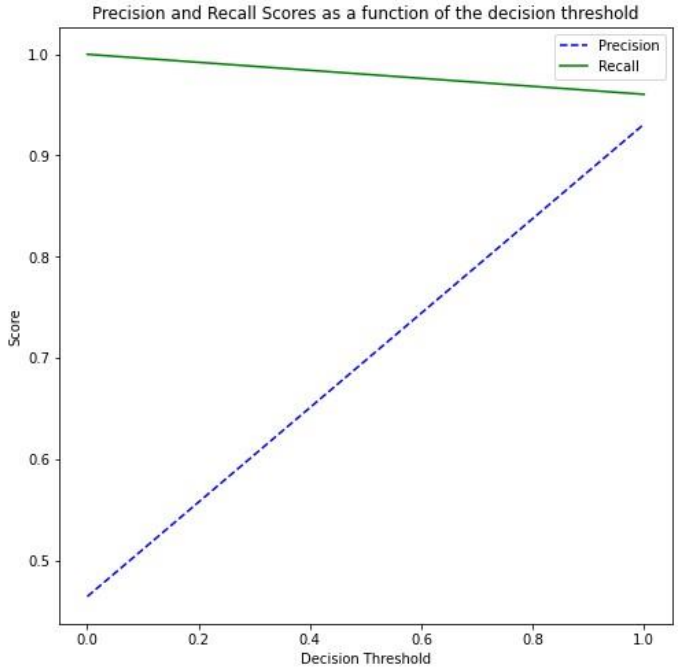
```
Random Forest: Precision =  94.9 %
Random Forest: Recall =  94.9 %
Random Forest: Accuracy =  94.9 %
Random Forest: F1-Score =  94.9 %
```

In [64]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, prediction)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

```
ROC AUC Score:  94.9 %
```



In [65]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, prediction)
plot_precision_recall_vs_threshold(p, r, thresholds)
```



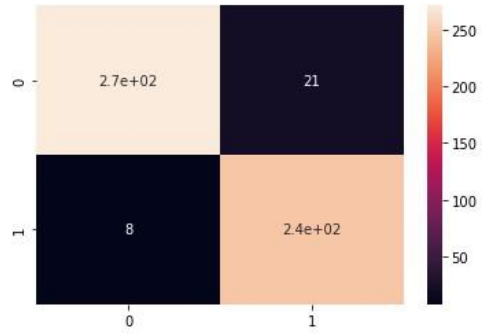
KNeighborsClassifier 2rd algorithm model (perfect)

In [66]:
```
KNeighbors = KNeighborsClassifier()
#Fitting the training data
KNeighbors.fit(xres,yres)
Validation = cross_val_score(KNeighbors, X_train, y_train)

print("cross-validation mean score: %.2f" % Validation.mean())
kfold_score = cross_val_score(KNeighbors, X_train, y_train)

print("average score kfold: %.2f" % kfold_score.mean())
#Predicting on test
predi=KNeighbors.predict(X_test)
```

```
cross-validation mean score: 0.95
average score kfold: 0.95
```

In [67]:
```
print("Training Accuracy :", KNeighbors.score(X_train, y_train))
print("Testing Accuracy :", KNeighbors.score(X_test, y_test))
MATRIX = confusion_matrix(y_test, predi)

sns.heatmap(MATRIX, annot = True)
plt.show()
print('\nclassification report')
print(classification_report(y_test, predi))
confusion_matrix(y_test, predi)
```

```
Training Accuracy : 0.9596330275229358
Testing Accuracy : 0.9467889908256881
```




```
classification report
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       292
           1       0.92      0.97      0.94       253

    accuracy                           0.95       545
   macro avg       0.95      0.95      0.95       545
weighted avg       0.95      0.95      0.95       545
```

Out[67]:
```
array([[271,  21],
       [  8, 245]], dtype=int64)
```

In [68]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,predi)
roc_auc1 = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc1)

precision, recall, thresholds = precision_recall_curve(y_test, predi)
f1 = f1_score(y_test, predi)
Precision = auc(recall, precision)
print("Precision-Recall Curves =",Precision)
```

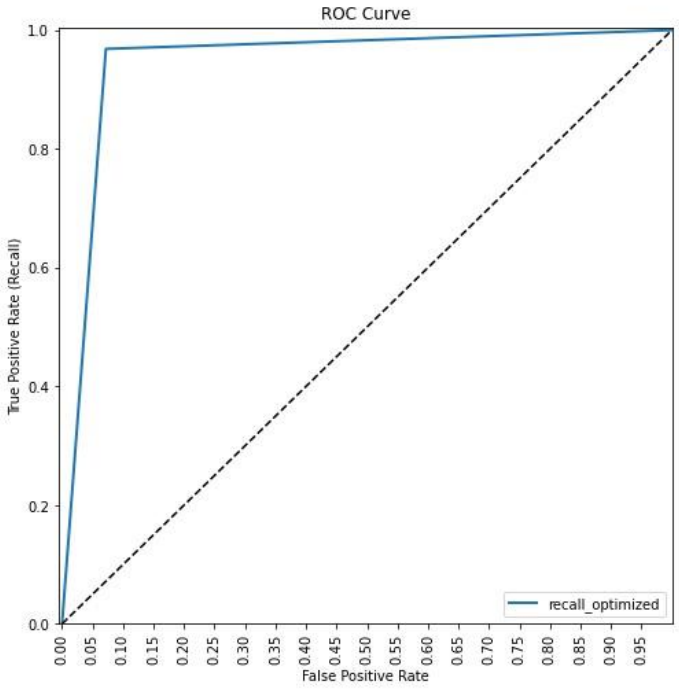
```
ROC Curves= 0.9482308192105691
Precision-Recall Curves = 0.9520554886509162
```

In [69]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, predi,pos_
print('Random Forest: Recall = ',str(round(recall_score(y_test, predi,pos_label=
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, predi)*100,1
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, predi,pos_label='p
```

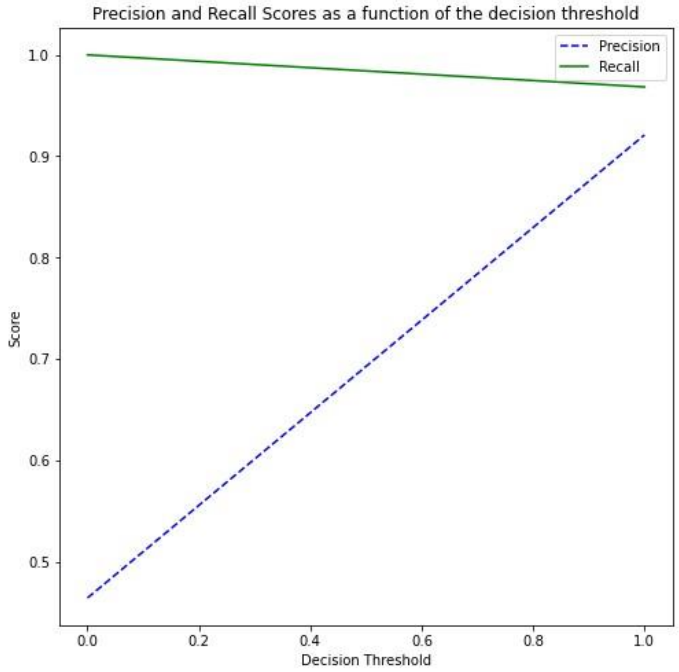
```
Random Forest: Precision =  94.7 %
Random Forest: Recall =  94.7 %
Random Forest: Accuracy =  94.7 %
Random Forest: F1-Score =  94.7 %
```

In [70]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, predi)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

```
ROC AUC Score:  94.8 %
```



In [71]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, predi)
plot_precision_recall_vs_threshold(p, r, thresholds)
```



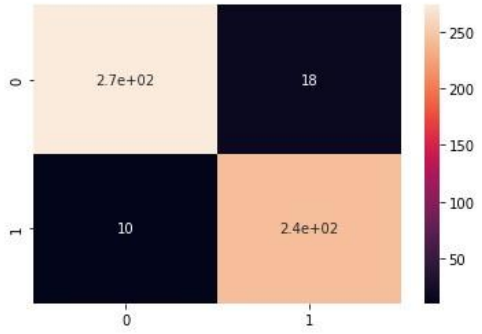
LinearDiscriminantAnalysis 3nd algorithm model (perfect)

In [72]:
```
LinearDiscriminantAnalysis = LinearDiscriminantAnalysis()
#Fitting the training data
LinearDiscriminantAnalysis.fit(xres,yres)
Validationn = cross_val_score(LinearDiscriminantAnalysis, X_train, y_train)

print("cross-validation mean score: %.2f" % Validationn.mean())
kfold_scoree = cross_val_score(LinearDiscriminantAnalysis, X_train, y_train)

print("average score kfold: %.2f" % kfold_scoree.mean())
#Predicting on test
predii=LinearDiscriminantAnalysis.predict(X_test)
```

```
cross-validation mean score: 0.96
average score kfold: 0.96
```

In [73]:
```
print("Training Accuracy :", LinearDiscriminantAnalysis.score(X_train, y_train))
print("Testing Accuracy :", LinearDiscriminantAnalysis.score(X_test, y_test))
MATRIXX = confusion_matrix(y_test, predii)

sns.heatmap(MATRIXX, annot = True)
plt.show()
print('\nclassification report')
print(classification_report(y_test, predii))
confusion_matrix(y_test, predii)
```

```
Training Accuracy : 0.963302752293578
Testing Accuracy : 0.9486238532110092
```




```
classification report
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       292
           1       0.93      0.96      0.95       253

    accuracy                           0.95       545
   macro avg       0.95      0.95      0.95       545
weighted avg       0.95      0.95      0.95       545
```

Out[73]:
```
array([[274,  18],
       [ 10, 243]], dtype=int64)
```

In [74]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,predii)
roc_auc2 = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc2)

precision, recall, thresholds = precision_recall_curve(y_test, predii)
f1 = f1_score(y_test, predii,pos_label='positive',average='micro')
Precision_Recall_rfs = auc(recall, precision)
print("Precision-Recall Curves =",Precision_Recall_rfs)
```

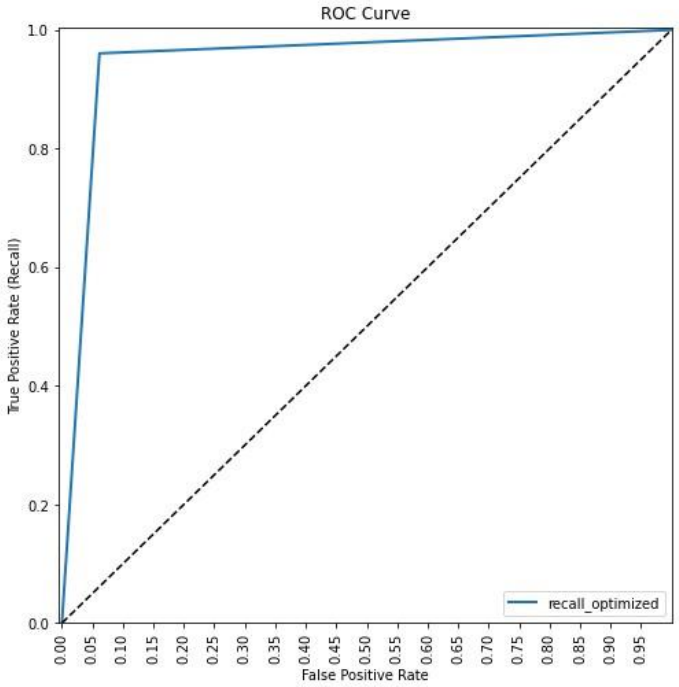
```
ROC Curves= 0.9494152363419784
Precision-Recall Curves = 0.9549287074561135
```

In [75]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, predii,pos
print('Random Forest: Recall = ',str(round(recall_score(y_test, predii,pos_label
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, predii)*100,
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, predii,pos_label='
```

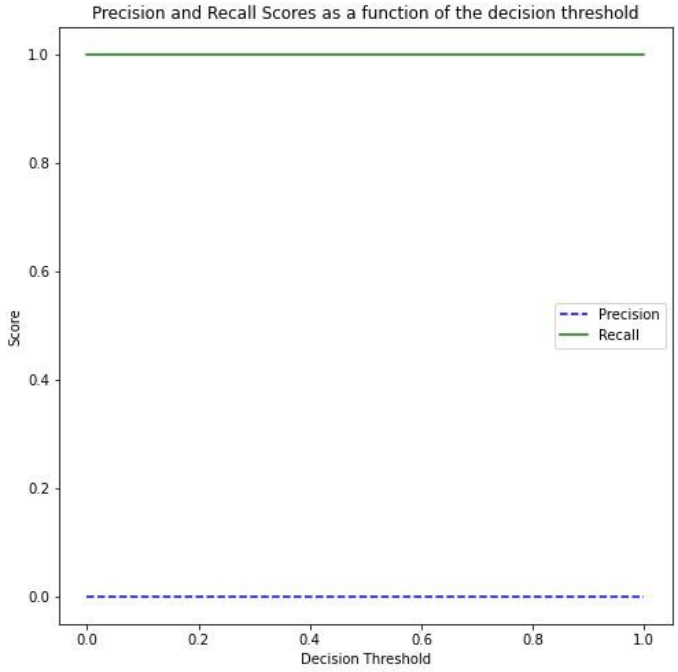
```
Random Forest: Precision =  94.9 %
Random Forest: Recall =  94.9 %
Random Forest: Accuracy =  94.9 %
Random Forest: F1-Score =  94.9 %
```

In [76]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, predii)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

```
ROC AUC Score:  94.9 %
```



In [77]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, predii, pos_label =2)
plot_precision_recall_vs_threshold(p, r, thresholds)
```



In [78]:
```
from sklearn.linear_model import LogisticRegression

# Incorrect
LogisticRegression = LogisticRegression()

#Fitting the training data
LR =LogisticRegression.fit(X_train, y_train)
Validatio = cross_val_score(LR, X_train, y_train)

print("cross-validation mean score: %.2f" % Validatio.mean())
kfol = cross_val_score(LR, X_train, y_train)

print("average score kfold: %.2f" % kfol.mean())
#Predicting on test
prediU=LR.predict(X_test)
```

```
cross-validation mean score: 0.93
average score kfold: 0.93
```

LogisticRegression 4 algorithm model (perfect)

In [78]:
```
from sklearn.linear_model import LogisticRegression

# Incorrect
LogisticRegression = LogisticRegression()

#Fitting the training data
LR =LogisticRegression.fit(X_train, y_train)
Validatio = cross_val_score(LR, X_train, y_train)

print("cross-validation mean score: %.2f" % Validatio.mean())
kfol = cross_val_score(LR, X_train, y_train)

print("average score kfold: %.2f" % kfol.mean())
#Predicting on test
prediU=LR.predict(X_test)
```

```
cross-validation mean score: 0.93
average score kfold: 0.93
```

In [80]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,prediU)
roc_auc = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc)

precision, recall, thresholds = precision_recall_curve(y_test, prediU)
f1 = f1_score(y_test, predii,pos_label='positive',average='micro')
Precision_Recall_rfs = auc(recall, precision)
print("Precision-Recall Curves =",Precision_Recall_rfs)
```

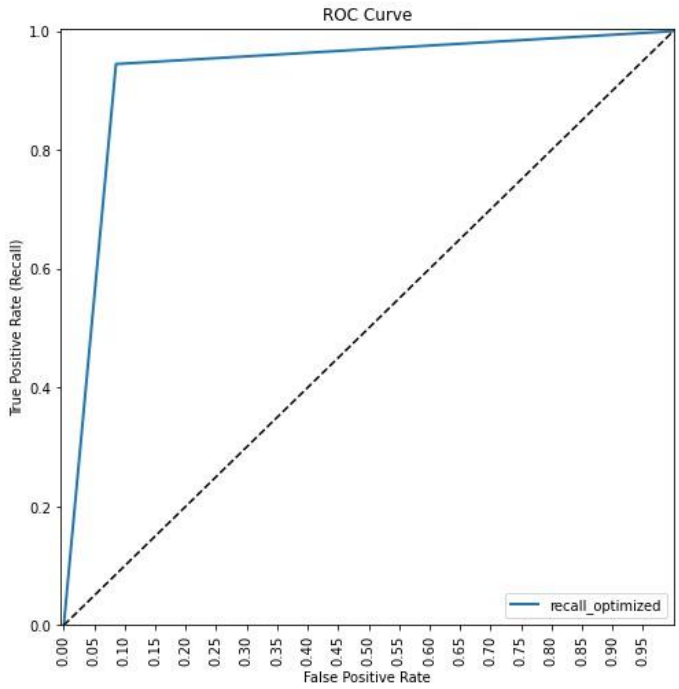
```
ROC Curves= 0.9295237966321944
Precision-Recall Curves = 0.9378275676590395
```

In [81]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, prediU,pos
print('Random Forest: Recall = ',str(round(recall_score(y_test, prediU,pos_label
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, prediU)*100,
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, prediU,pos_label='
```

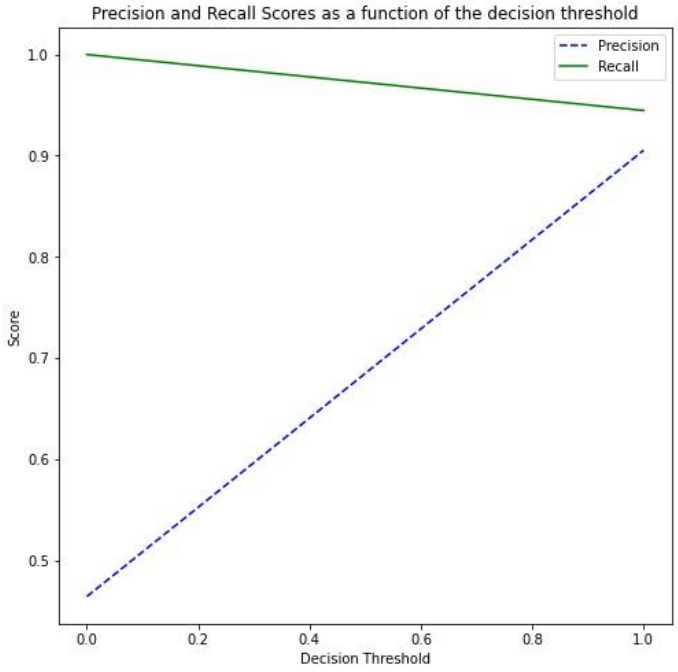
```
Random Forest: Precision =  92.8 %
Random Forest: Recall =  92.8 %
Random Forest: Accuracy =  92.8 %
Random Forest: F1-Score =  92.8 %
```

In [82]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, prediU)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

```
ROC AUC Score:  93.0 %
```



In [83]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, prediU)
plot_precision_recall_vs_threshold(p, r, thresholds)
```



NearestCentroid algorithm 5 model (perfect)

In [84]:

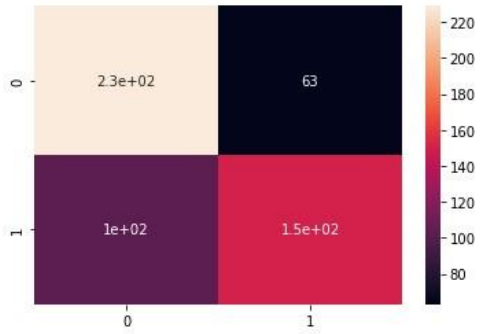
```
NearestCentroid = NearestCentroid()
#Fitting the training data
NearestCentroid.fit(xres,yres)
Cross = cross_val_score(NearestCentroid, X_train, y_train)

print("cross-validation mean score: %.2f" % Cross.mean())
Cross_K = cross_val_score(NearestCentroid, X_train, y_train)

print("average score kfold: %.2f" % Cross_K.mean())
#Predicting on test
#Predicting on test
y_prre=NearestCentroid.predict(X_test)
```

```
cross-validation mean score: 0.71
average score kfold: 0.71
```

In [85]:

```
print("Training Accuracy :", NearestCentroid.score(X_train, y_train))
print("Testing Accuracy :", NearestCentroid.score(X_test, y_test))

uuy = confusion_matrix(y_test, y_prre)

sns.heatmap(uuy, annot = True)
plt.show()
print('\nclassification report')
print(classification_report(y_test, y_prre))
print(confusion_matrix(y_test, y_prre))
```

```
Training Accuracy : 0.7162079510703364
Testing Accuracy : 0.6954128440366972
```




```
classification report
              precision    recall  f1-score   support

           0       0.69      0.78      0.73       292
           1       0.70      0.59      0.64       253

    accuracy                           0.70       545
   macro avg       0.70      0.69      0.69       545
weighted avg       0.70      0.70      0.69       545

[[229  63]
 [103 150]]
```

In [86]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,y_prre)
roc_auc = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc)

precision, recall, thresholds = precision_recall_curve(y_test, y_prre)
f1 = f1_score(y_test, y_prre,pos_label='positive',average='micro')
Precision_Recall_rfs = auc(recall, precision)
print("Precision-Recall Curves =",Precision_Recall_rfs)
```

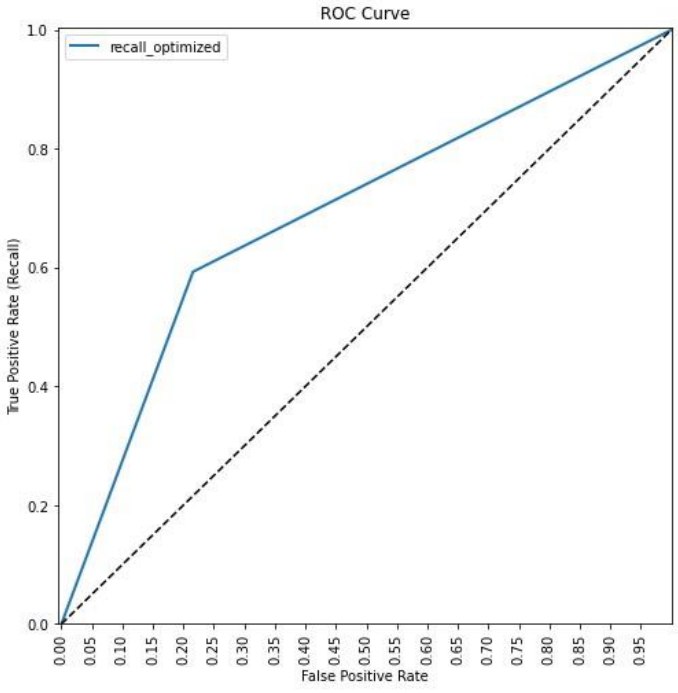
```
ROC Curves= 0.6885659754182686
Precision-Recall Curves = 0.7430507766474103
```

In [87]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, y_prre,pos
print('Random Forest: Recall = ',str(round(recall_score(y_test, y_prre,pos_label
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, y_prre)*100,
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, y_prre,pos_label='
```

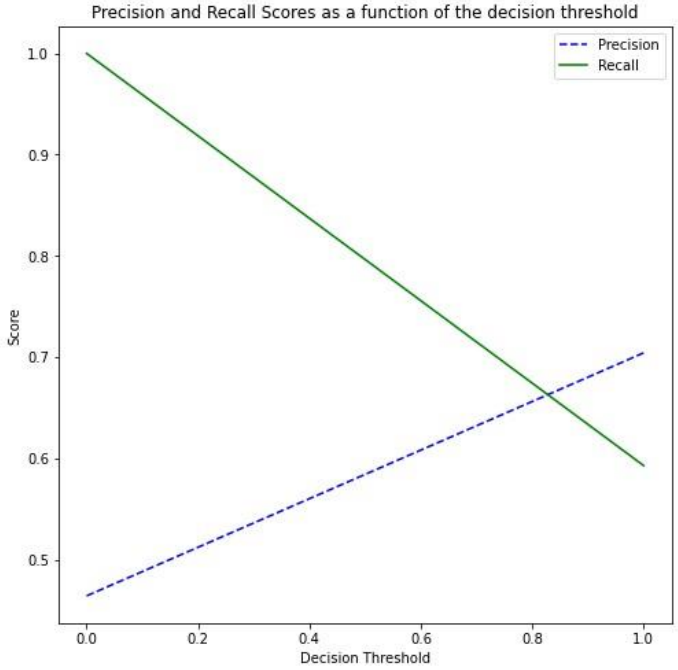
```
Random Forest: Precision =  69.5 %
Random Forest: Recall =  69.5 %
Random Forest: Accuracy =  69.5 %
Random Forest: F1-Score =  69.5 %
```

In [88]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, y_prre)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

```
ROC AUC Score:  68.9 %
```



In [89]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, y_prre)
plot_precision_recall_vs_threshold(p, r, thresholds)
```



SVC 6 algorithm model

In [90]:
```
from sklearn.svm import SVC
SVC = SVC()

SV =SVC.fit(xres,yres)
Cros = cross_val_score(SV, X_train, y_train)

print("cross-validation mean score: %.2f" % Cros.mean())
Cros_K = cross_val_score(SV, X_train, y_train)

print("average score kfold: %.2f" % Cross_K.mean())
#Predicting on test
#Predicting on test
y_pre33=SV.predict(X_test)
```

```
cross-validation mean score: 0.96
average score kfold: 0.71
```

In [91]:
```
print("Training Accuracy :", SVC.score(X_train, y_train))
print("Testing Accuracy :", SVC.score(X_test, y_test))

mm = confusion_matrix(y_test, y_pre33)

sns.heatmap(mm, annot = True)
plt.show()
print('\nclassification report')

print(classification_report(y_test, y_pre33))
print(confusion_matrix(y_test, y_pre33))
```

```
Training Accuracy : 0.963914373088685
Testing Accuracy : 0.9669724770642202
```


```
classification report
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       292
           1       0.94      0.99      0.97       253

    accuracy                           0.97       545
   macro avg       0.97      0.97      0.97       545
weighted avg       0.97      0.97      0.97       545

[[276  16]
 [  2 251]]
```

In [92]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,y_pre33)
roc_auc = auc(false_positive_rate, true_positive_rate)
print("ROC Curves=",roc_auc)

precision, recall, thresholds = precision_recall_curve(y_test, y_pre33)
f1 = f1_score(y_test, y_pre33,pos_label='positive',average='micro')
Precision_Recall_rfs = auc(recall, precision)
print("Precision-Recall Curves =",Precision_Recall_rfs)
```

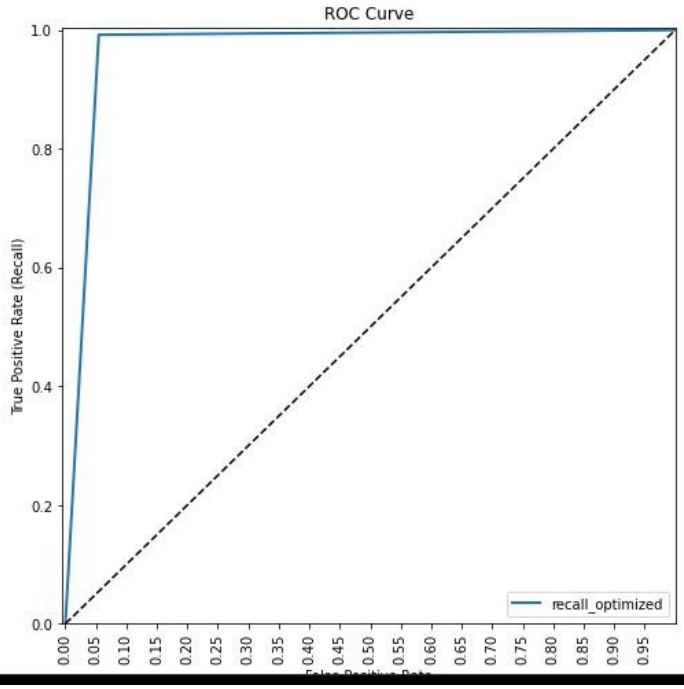
```
ROC Curves= 0.968650170556067
Precision-Recall Curves = 0.9679197463988812
```

In [93]:
```
print('Random Forest: Precision = ',str(round(precision_score(y_test, y_pre33,po
print('Random Forest: Recall = ',str(round(recall_score(y_test, y_pre33,pos_labe
print('Random Forest: Accuracy = ',str(round(accuracy_score(y_test, y_pre33)*100
print('Random Forest: F1-Score = ',str(round(f1_score(y_test, y_pre33,pos_label=
```

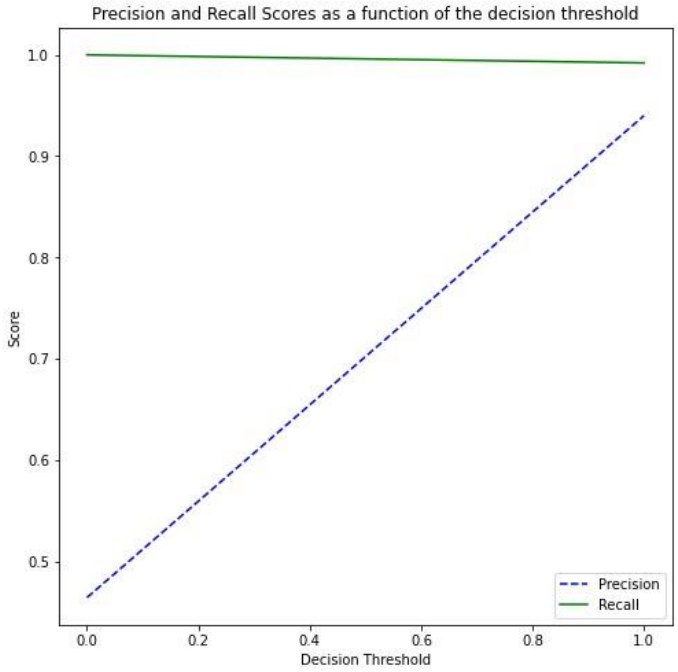
```
Random Forest: Precision =  96.7 %
Random Forest: Recall =  96.7 %
Random Forest: Accuracy =  96.7 %
Random Forest: F1-Score =  96.7 %
```

In [94]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
    plt.legend(loc='best')

fpr, tpr, auc_thresholds = roc_curve(y_test, y_pre33)
print('ROC AUC Score: ',str(round(auc(fpr, tpr)*100,1)), '%') # AUC of ROC
plot_roc_curve(fpr, tpr, 'recall_optimized')
```

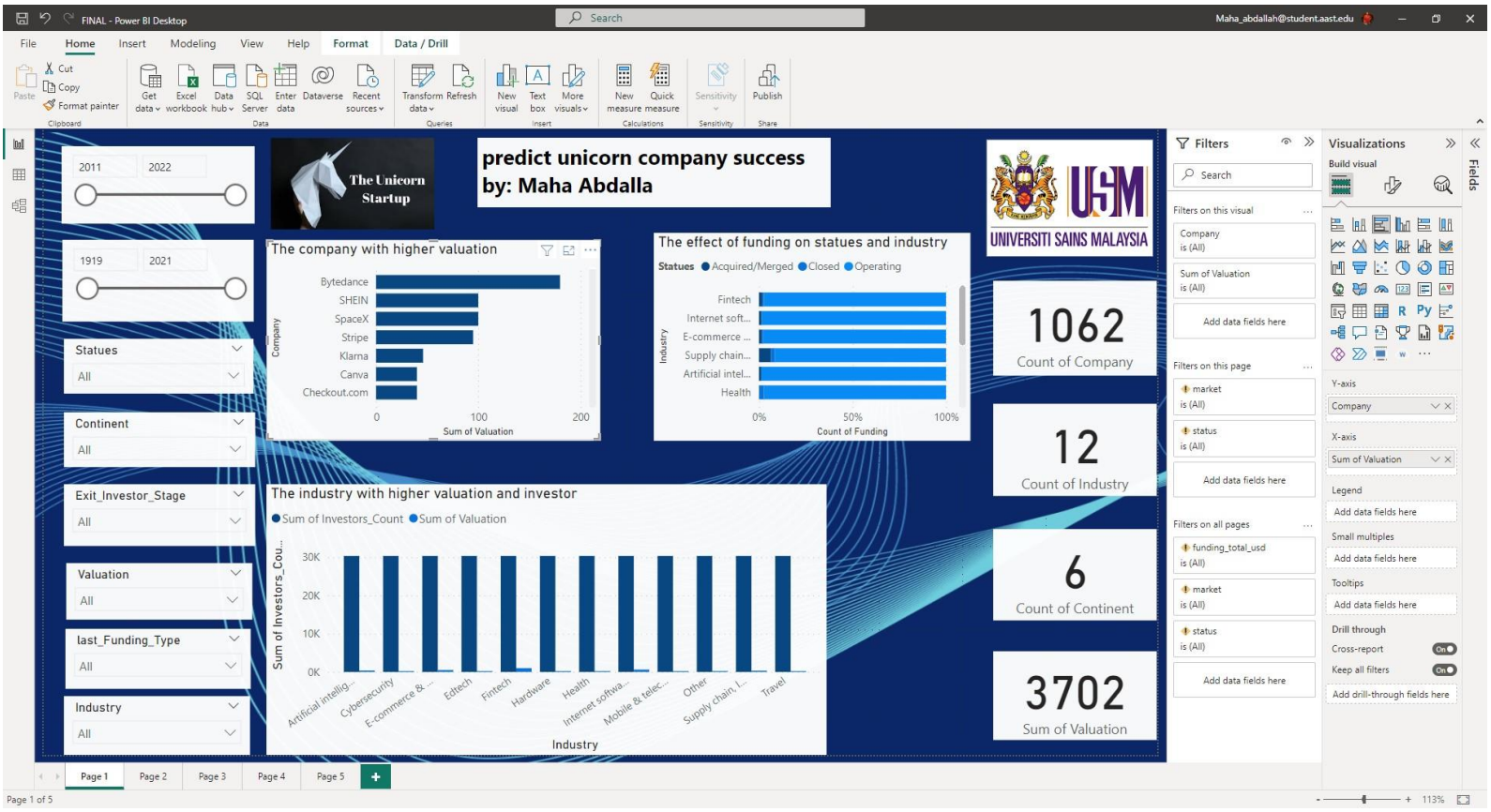
```
ROC AUC Score:  96.9 %
```

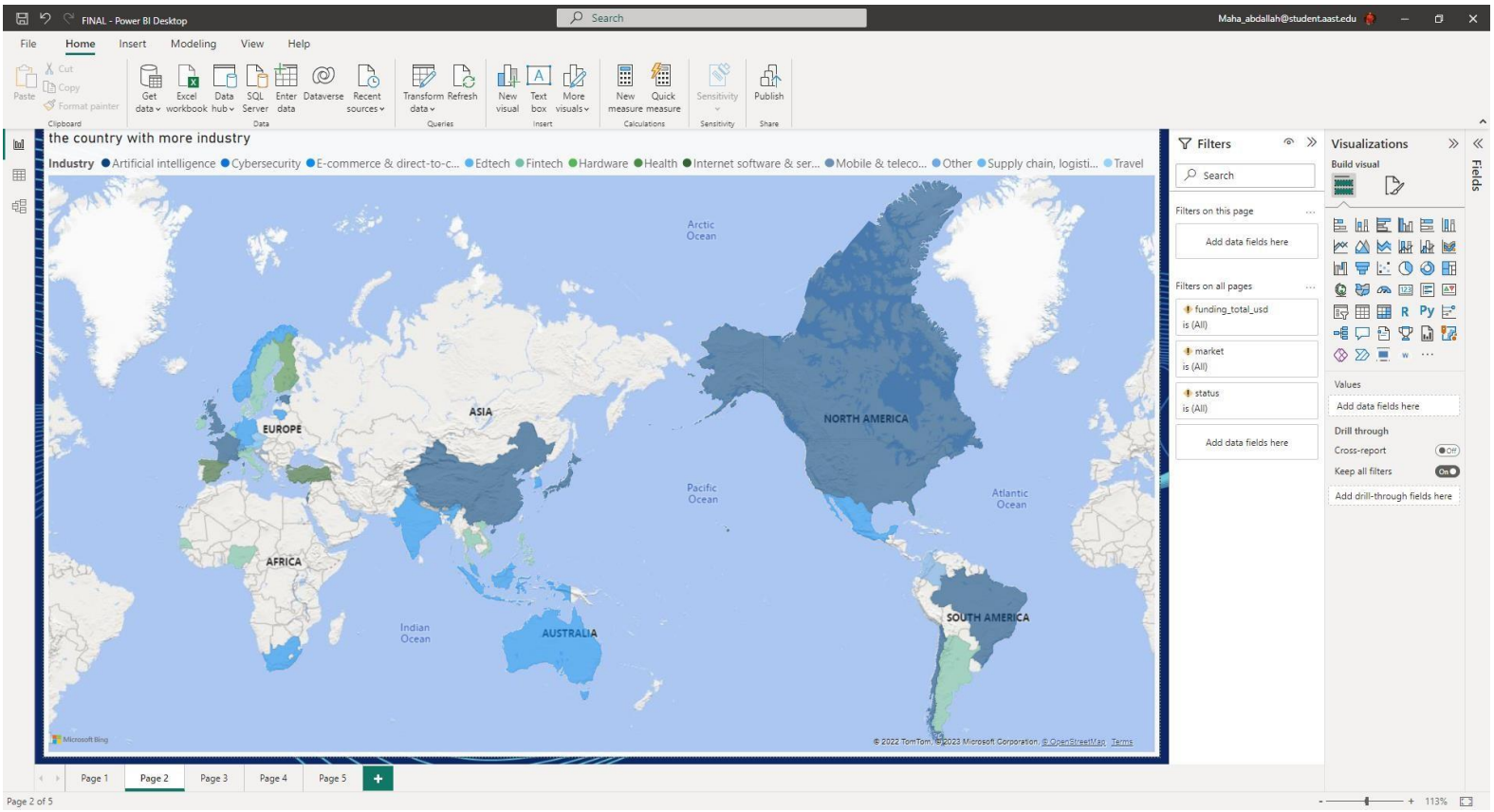


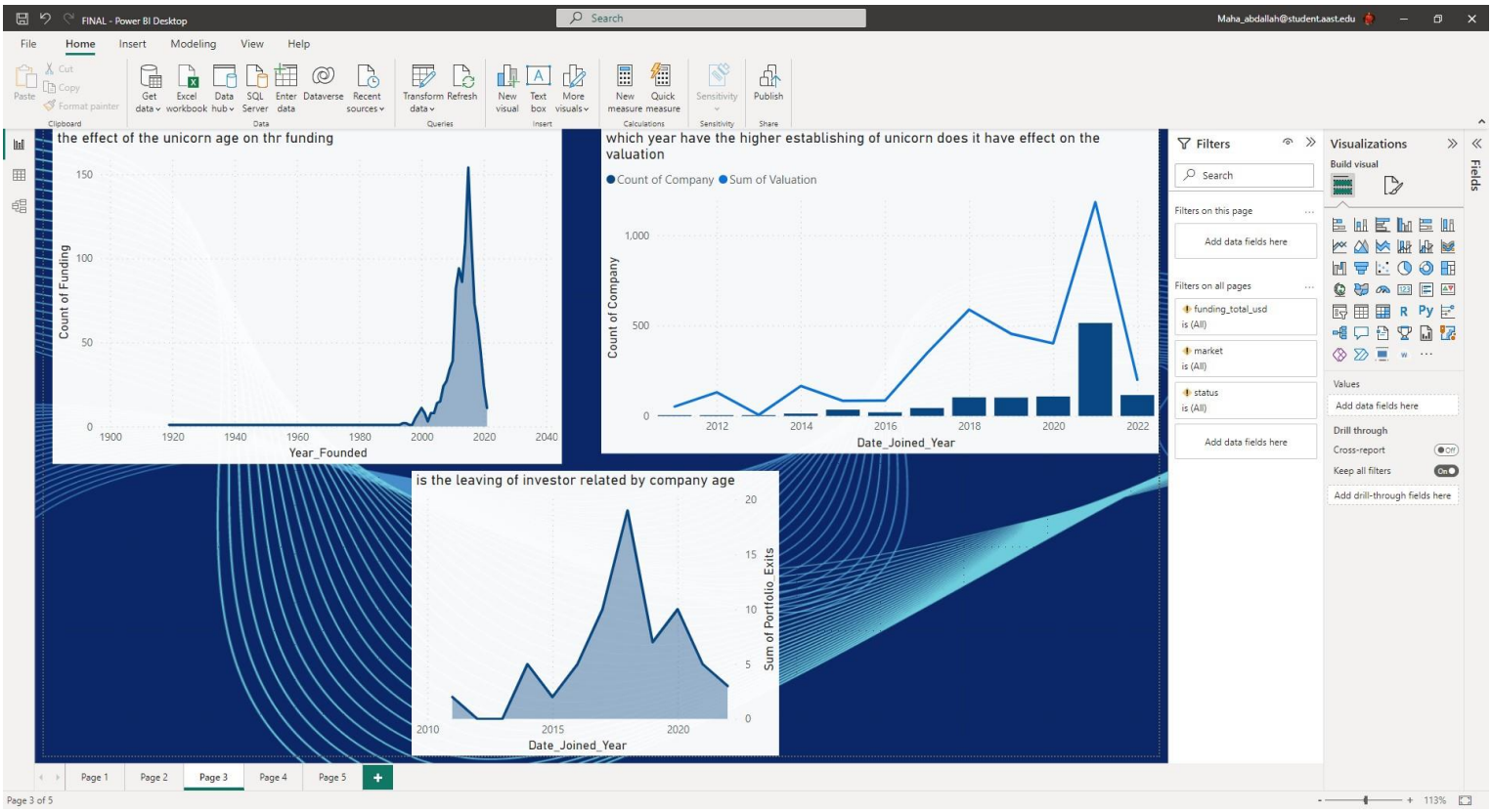
In [95]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):

    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision thresho
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')

p, r, thresholds = precision_recall_curve(y_test, y_pre33)
plot_precision_recall_vs_threshold(p, r, thresholds)
```

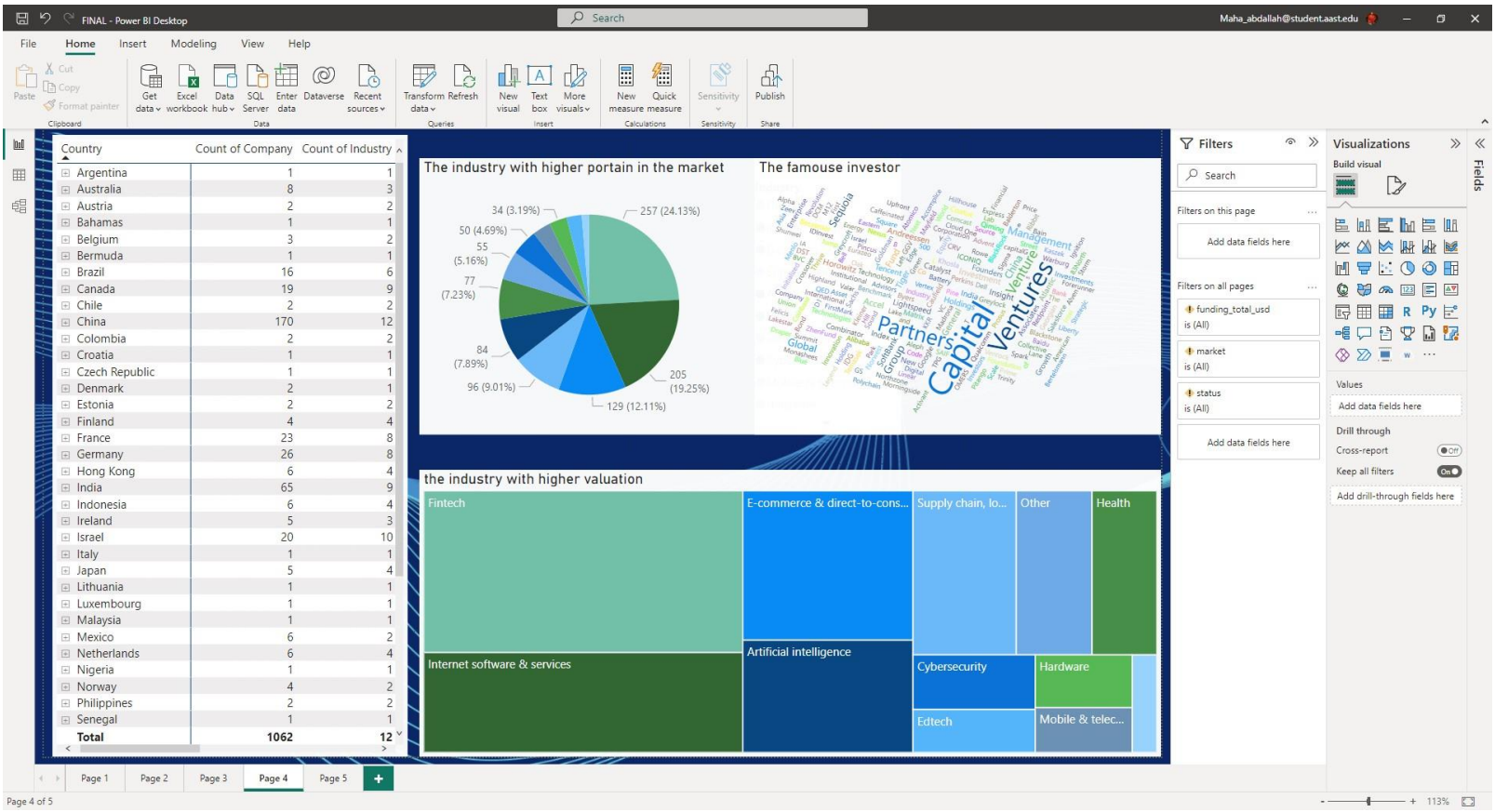


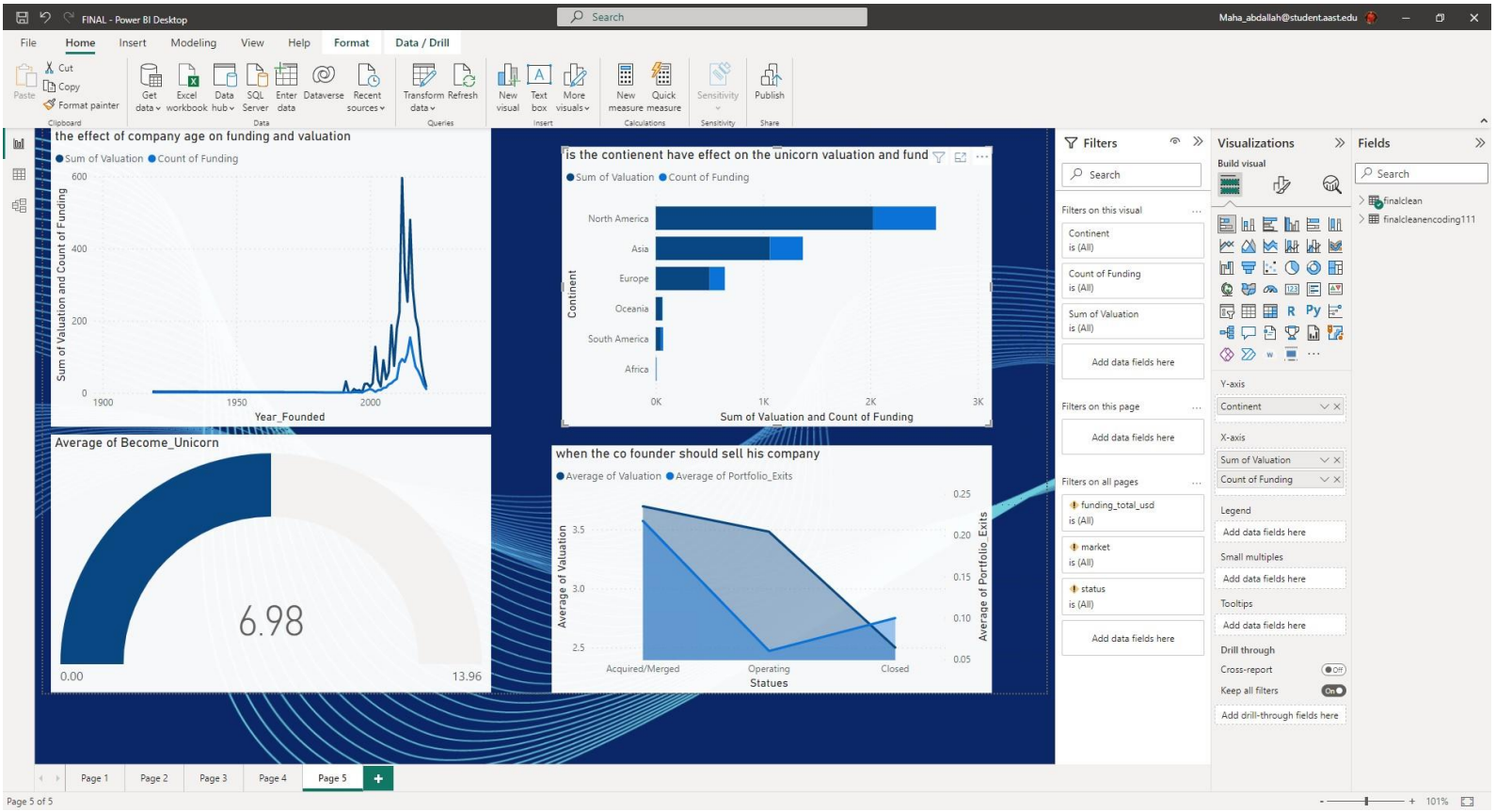
8. Power BI visualization

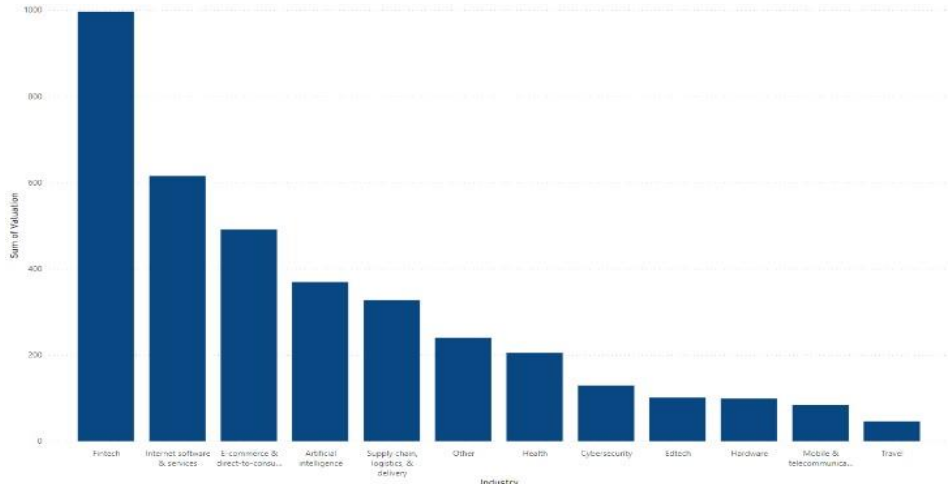



FINAL - Power BI Desktop
Search
Maha_abdallah@student.aast.edu
File
Home
Insert
Modeling
View
Help
the country with more industry
Industry
Artificial intelligence
Cybersecurity
E-commerce & direct-to-c...
Edtech
Fintech
Hardware
Health
Internet software & ser...
Mobile & teleco...
Other
Supply chain, logisti...
Travel
Arctic Ocean
EUROPE
ASIA
NORTH AMERICA
Pacific Ocean
Atlantic Ocean
AFRICA
Indian Ocean
AUSTRALIA
SOUTH AMERICA
Microsoft Bing
© 2022 TomTom, © 2023 Microsoft Corporation, © OpenStreetMap Terms
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
funding_total_usd
is (All)
market
is (All)
status
is (All)
Add data fields here
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Add drill-through fields here
Fields
Page 1
Page 2
Page 3
Page 4
Page 5
Page 2 of 5
113%


FINAL - Power BI Desktop
Search
Maha_abdallah@student.aast.edu
File
Home
Insert
Modeling
View
Help
the effect of the unicorn age on thr funding
Count of Funding
Year_Founded
which year have the higher establishing of unicorn does it have effect on the valuation
Count of Company
Sum of Valuation
Count of Company
Date_Joined_Year
is the leaving of investor related by company age
Sum of Portfolio_Exits
Date_Joined_Year
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
funding_total_usd
is (All)
market
is (All)
status
is (All)
Add data fields here
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Add drill-through fields here
Fields
Page 1
Page 2
Page 3
Page 4
Page 5
Page 3 of 5
113%


FINAL - Power BI Desktop
Search
Maha_abdallah@student.aast.edu
File
Home
Insert
Modeling
View
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
Data hub
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Publish
Share
Country
Count of Company
Count of Industry
Argentina 1 1
Australia 8 3
Austria 2 2
Bahamas 1 1
Belgium 3 2
Bermuda 1 1
Brazil 16 6
Canada 19 9
Chile 2 2
China 170 12
Colombia 2 2
Croatia 1 1
Czech Republic 1 1
Denmark 2 1
Estonia 2 2
Finland 4 4
France 23 8
Germany 26 8
Hong Kong 6 4
India 65 9
Indonesia 6 4
Ireland 5 3
Israel 20 10
Italy 1 1
Japan 5 4
Lithuania 1 1
Luxembourg 1 1
Malaysia 1 1
Mexico 6 2
Netherlands 6 4
Nigeria 1 1
Norway 4 2
Philippines 2 2
Senegal 1 1
Total 1062 12
The industry with higher portain in the market
34 (3.19%)
257 (24.13%)
50 (4.69%)
55 (5.16%)
77 (7.23%)
84 (7.89%)
96 (9.01%)
205 (19.25%)
129 (12.11%)
The famouse investor
Capital
Partners
Ventures
the industry with higher valuation
Fintech
E-commerce & direct-to-cons...
Supply chain, lo...
Other
Health
Internet software & services
Artificial intelligence
Cybersecurity
Hardware
Edtech
Mobile & telec...
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
funding_total_usd is (All)
market is (All)
status is (All)
Add data fields here
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Add drill-through fields here
Fields
Page 1
Page 2
Page 3
Page 4
Page 5
Page 4 of 5
113%


FINAL - Power BI Desktop
Search
Maha_abdallah@student.aast.edu
File
Home
Insert
Modeling
View
Help
Format
Data / Drill
the effect of company age on funding and valuation
Sum of Valuation
Count of Funding
Sum of Valuation and Count of Funding
Year_Founded
is the contienent have effect on the unicorn valuation and fund
North America
Asia
Europe
Oceania
South America
Africa
Continent
Sum of Valuation and Count of Funding
Average of Become_Unicorn
6.98
0.00
13.96
when the co founder should sell his company
Average of Valuation
Average of Portfolio_Exits
Acquired/Merged
Operating
Closed
Statues
Filters
Visualizations
Fields
Page 1
Page 2
Page 3
Page 4
Page 5


Sum of Valuation
Fintech
Internet software & services
E-commerce & direct-to-consu...
Artificial intelligence
Supply chain, logistics, & delivery
Other
Health
Cybersecurity
Edtech
Hardware
Mobile & telecommunica...
Travel