Using Social Media and Personality Predictions to Anticipate Startup Success

Daniel Stenson

Master's thesis
2023:E73

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2023:E73
ISSN 1404-6342

LUTFMS-3492-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

ABSTRACT

This thesis explores the potential of integrating predicted founder personalities, based on the Big 5 Personality Framework, into Machine Learning (ML) models to enhance the accuracy of early-stage startup success predictions. Leveraging Natural Language Processing (NLP) techniques, we extracted personality insights from founders' tweets, focusing on US startups funded between 2013 and 2015. Our research utilized a range of models, including XGBoost, Random Forest, and Feed-forward Neural Network for personality predictions, and Logistic Regression, XGBoost, and Random Forest for startup success forecasts. Results indicated that most personality-predicting models outperformed the Naive baseline. In success predictions, XGBoost emerged as the top performer, showcasing the highest scores in Macro F1 and AUC for both Series B and Series C funding rounds. While the trait of Neuroticism was highlighted as significant for Series B predictions across models, Series C predictions emphasized the importance of Openness and Agreeableness. Our findings underline the value of integrating predicted personality traits into ML models for startup success forecasts. However, as with all research, our work had inherent limitations and suggested areas for further exploration and improvement.

Keywords: Machine Learning, Startup Success Predictions, Founder Personalities, Natural Language Processing, Social Media Analysis, Big 5 Personality Framework, Feed-forward Neural Network, XGBoost

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my friends and family, whose support and encouragement have been a constant source of strength throughout this journey.

A special thanks to my supervisor, Filip Tronarp, for his guidance, patience, and expertise. His insights and dedication have greatly contributed to my research.

I am also grateful to the few investors who have generously shared their time and knowledge with me. Their unbiased perspectives and emphasis on what truly matters have been instrumental in shaping both this thesis and my understanding of the field.

Thank you all for your invaluable contributions.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Aim and Scope

This thesis aims to ascertain the significance and potential value of integrating predicted founder personalities into ML models for enhancing the accuracy of early-stage startup success predictions. Hence, an integral part of this thesis is to construct well performing personality predictions models, that rely on Natural Language inputs from social media posts, or tweets.

For this project we use data from US startups who received seed-funding between the years 2013 and 2015. This time frame allows us to reliably gauge a startup's success, under the assumption that if they have not succeeded by now, they likely will not. For the personality predictions we restrict ourselves to only using the Big 5 personality Framework as both target variables for the personality models and as input variables for the startup success models.

We use and compare a feed-forward neural network, an XGBoost model, and a random forest model for the personality predictions that uses either TF-IDF or BERT embeddings as inputs. For the startup success predictions, we employ logistic regression models, XGBoost models, and random forest models, and compare their performance and reliance on personality features. The models are, to varying extents, tuned, and subjected to different forms of regularization, to achieve the best possible results.

1.2 Background

Startups are young businesses intending to disrupt industries [2]. Unlike other businesses, startups focus on speed and growth. They rely on large investments from institutional investors to bring new products and services to market, intending to outmaneuver their competition and grab large market shares quickly. Given the aggressive speed and growth orientation of startups, obtaining sufficient funding becomes a critical factor for success. This is where the concept of Venture Capital comes into play. Venture Capital (VC) is a form of financing or private equity to startup companies believed to have long-term growth potential [3]. VC can come from a myriad of investors, such as individual investors (known as angel investors,

or angels), VC firms, investments banks or other financial institutions. The goal for these investors is to see high returns, which is why the capital is typically provided at a very early stage in a company's development, often at the seed- or early-stage. VC firms collect and pool money from financially well-off people or institutions known as limited partners (LPs) and use these funds to invest in startups. The decision-making process within a VC firm can vary based on numerous factors, but it typically unfolds according to the following sequence: [4]

1. Origination: At this stage, VC firms seek out potential investment opportunities via either inbound or outbound sources.

2. Screening: This involves a two-fold assessment - VC Firm-specific and Generic. Firstly, it's necessary to ascertain whether the opportunity aligns with the VC firm's sector interests and investment thesis. Secondly, the investment potential is evaluated from a growth perspective, among other generic factors.

3. First-Phase Evaluation: In this phase, the management team of the potential investment interacts with the VC firm. An initial evaluation is conducted with the objective of gauging the potential appeal of the opportunity.

4. Second-Phase Evaluation: Should the opportunity prove to be promising, it advances to this phase, designed to rapidly amass as much knowledge as possible about the prospective venture. The objective here is to streamline the decision-making process and prevent the deal from being secured by other investors.

5. Closing: Following the second-phase evaluation, the deal proceeds to the closing stage. This is when the terms of the agreement and the legal obligations are established and finalized.

At every stage of the process, the VC Firm can decide that a deal is not interesting, and therefore opt-out.

When evaluating opportunities, also known as conducting due diligence, VC firms look at numerous different factors aiding them in making the right decisions. Generally, these factors include, but are not limited to, team, business model, product, market, industry, valuation, ability to add value, and fit [5].

Numerous studies have been done trying to identify the factors that lead to successful ventures; in [6] a literature review of success factors for IT startups was conducted. They identified three categories of success factors, what they named organizational, individual, and external factors. The organizational category includes factors such as organizational age and size, and location of a company

(which is tied to connections of potential customers, advisors and investors). Individual factors are related to the founders or founding team, their personalities, previous work and startup experience. Lastly, external factors contain factors such as market competition, innovation, and changes in legislation that drive performance and growth. Some of the internal factors identified in [6] are related to personalities and personality traits of the founding team, and therefore could support the notion of personalities having an effect on the success of a startup. This is especially relevant for the aim of this paper. These factors include leadership, motivation, independence, social skills, and business attitude of the founders.

Research in the field of personality psychology establishes a prominent framework known as the Five-Factor Model (FFM), also referred to as the Big Five personality traits [7]. This model identifies five key dimensions that define human personality and govern human behavior: Openness (to Experience), Conscientiousness, Extraversion, Agreeableness, and Neuroticism. These dimensions originate from peer rating scales [8] and are currently traced in various research areas such as questionnaires of needs and motives [9], and self-reports on trait-descriptive adjectives [10]. As a result, it stands as one of the most widely used frameworks for personality modeling. The framework typically assigns a number on a scale of 1-100, where each subject receives a score on all five dimensions, which together constitute their personality. Additionally, research on the FFM suggests that personalities change over time [11], a factor that will be crucial for data selection, as discussed later.

1.3 Previous Research

In previous research, various approaches have been employed to predict the success of startups. In 2016 [12] utilized Crunchbase data, supplemented with additional information from TechCrunch, to predict success at different startup stages from Pre-seed to Series G. Machine learning methods used included naive bayes, ADTrees, and random forest. The study reported high AUC, precision, and recall scores across different models, with the best model showing an AUC score of 87%.

In 2018 [13] employed a two-step approach that first identified key uncertainty factors before investing in startups. These factors were then assessed by a 'domain expert' and the assessments were used to train machine learning models. Algorithms used in this study included k-Nearest Neighbors (k-NN), naive bayes, and support vector machine (SVM), with the naive bayes model showing the most promising results, boasting a precision of 88% and a recall of 81%.

Further research in 2018 utilized Crunchbase data to predict whether a startup would be acquired or complete an Initial Public Offering (IPO) [14]. This study

employed the random forest algorithm and achieved a true positive rate of 94%.

Moreover, a 2018 paper expanded on data sources by including information from forums, social media sites, and news articles about companies in addition to Crunchbase data [15]. An advanced learning pipeline was used constituting of various algorithms including logistic regression, neural network, and CatBoost (a gradient boosting decision tree modification). Their best performing model achieved a ROC-AUC of 85%.

[16] used a hybrid intelligence approach using both hard information (team, capital raised, etc.,) as well as soft information (product innovation, proof of concept, etc.,.) where the latter was judged by a group of people, and then trained on various models such as logistic regression, naive bayes and SVM.

Another paper from 2019 [17] used a similar approach with Crunchbase data and machine learning models such as logistic regression, random forest, and eXtreme Gradient Boosting (XGB), reporting an accuracy of 95%.

Lastly, a 2021 study [18] also utilized Crunchbase data but tried to avoid look-ahead biased variables. This study employed logistic regression, SVM, and a Gradient Boosting classifier (GBM), with the last one performing best with a Macro F1 score of 43%.

Looking at the papers referenced above, several of them can be critiqued with several areas of improvement. Especially, the use of look-ahead factors in both [12] and [15] could have unknowingly influenced prediction outcomes. [13] which employed key uncertainty factors, raises concerns about the potential bias due to the limited dataset used for training and testing, comprising only 198 and 49 observations respectively. In [17], additional look-ahead bias might be displayed, due to the use of variables not known at the early stages of a startup, such as the number of funding rounds. Additionally, the claim of a 95% accuracy rate is questionable given the unbalanced data set, which only had 5% of successes. Lastly, [18] which to a certain extent pointed out these areas of improvement, can still be critiqued for the small number of variables, which might be one explanation for the relatively low F1-score (43%).

Moving on to previous research in the field of personality predictions; one of the pioneering papers in this field is [19] from 2011. In their study, they collect a set of users, have them perform Big 5 personality tests, and subsequently extract their tweets and Twitter user data. From this, they create a set of features including LIWC language features, MRC language features, sentiment, and more. They then feed these features into two different models to predict personalities: a ZeroR model and a Gaussian process model. They conclude that it is possible to predict individuals' personalities accurately using Twitter data, obtaining mean average errors (MAEs) of between 0.12-0.18 for the five different personality traits. However, a primary critique of this paper is the absence of a baseline or naive model,

which complicates the interpretation of their results. More recently, similar approaches employing advanced NLP techniques have been evaluated. In 2021, [20] used pre-trained language models such as BERT and RoBERTa, combined with NLP features like the number of tweets and sentiment, to predict personalities from Twitter and Facebook data. They choose to approach it as a classification problem, dividing the personality trait scores into two classes, 'low' and 'high'. For the Twitter dataset, they achieved a (macro) F1 score of 0.88, with the lowest being 0.74. Opting for a classification approach over a regression is intriguing but might overlook some nuances in the scores that a regression approach could capture.

1.4 Defining Startup Success

Defining success in the context of startups can be a complex task due to the many outcomes that may be considered "successful". In this paper, we propose a novel approach to define success, arguing that a startup's ability to raise subsequent funding rounds, specifically Series B or Series C and beyond, should be considered a strong measure of success. This metric can be seen as a reflection of the startup's ability to consistently demonstrate promising growth and traction, which persuades investors to provide further capital.

However, other commonly used indicators of success such as mergers and acquisitions (M&A) can be misleading. It is often difficult to discern whether an M&A event signals success or represents a final attempt to prevent failure.

Similarly, an Initial Public Offering (IPO) is a rare event among startups, with even successful ones often deferring this step for extended periods. Using IPOs as a marker of success might inadvertently exclude many successful startups that have not yet reached this milestone.

For these reasons, this study asserts that the ability to raise Series B or Series C funding is a more accurate and encompassing measure of startup success, as it directly correlates with demonstrated traction and growth.

1.5 Scientific Contributions

The research conducted in this thesis will help understand if and to what extent different personality traits affect the outcome of startups and give a practical way for researchers to incorporate personalities into models aiming to predict the success of startups. This thesis will not only aid researchers, but also assist VC firms, individual investors, and founders in making informed decisions for both investment and recruitment purposes.

1.6 Outline of the Thesis

This thesis is structured as follows: Chapter 2 introduces the project data, detailing its collection, pre-processing, and the final features used for modeling both personalities and startup success. Chapter 3 includes a comprehensive coverage of necessary theory, ranging from text embeddings to machine learning. Chapter 4 outlines the methods used in this thesis. Chapter 5 consolidates all results, offering a clear overview for the reader. Chapter 6 delves into discussions about the results and provides recommendations for future research on the topic. The thesis concludes with Chapter 7, which contains our final thoughts and insights.

1.7 Problem Formulation

We will start by formulating the problems of personality predictions and success predictions mathematically, to help understand what we want to accomplish.

Personality Prediction

A user's set of tweets is represented as $X = \{x_1, x_2, \ldots, x_n\}$ where x_i denotes the ith tweet and n is the total number of tweets by the user ($10 < n \leq 1000$). Each set of tweets can be represented as a single document, D. A function, t, is used to generate a numerical representation (text embedding) of the document,

$$T_D = t(D).$$

The corresponding personality score is represented as $Y = \{y_1, y_2, \ldots, y_5\}$ where y_i is the score for the ith personality trait in the Five-Factor Model.

The aim is then to construct a model to learn a function $f : T_D \to Y$ such that the squared error ϵ between the predicted personality score $f(T_d)$ and the true personality score Y is minimized, i.e.,

$$\min \epsilon^2 \left(Y, f(T_D)\right).$$

Startup Success Prediction

The predicted personality scores serve as input to a separate model for predicting startup success. The set of personality scores is represented as $P = \{p_1, p_2, ..., p_5\}$, where p_i denotes the mean personality score among the founders for the ith personality trait in the Five-Factor Model. Additionally, we represent the categorical variables as $C = \{c_1, c_2, ..., c_p\}$, where c_i is the ith categorical variable and p is the total number of categorical variables, and the numerical variables as $N = \{n_1, n_2, ..., n_q\}$, where n_i is the ith numerical variable and q is the total number of numerical variables.

The complete input to the model then becomes a combination of P, C, and N. This can be represented as a vector $I = (P, C, N)$, where I includes the mean personality traits among founders, additional categorical variables, and numerical variables.

The corresponding startup success label is denoted as $S \in \{0, 1\}$, where 0 represents failure and 1 represents success.

The overarching goal is not just to construct a model to learn a function $g : I \to S$ that outputs the probability of startup success, but also to minimize the binary cross-entropy loss between the predicted probability $g(I)$ and the true startup success label S:

$$L(S, g(I)) = -\left(S \log(g(I)) + (1 - S) \log(1 - g(I))\right)$$

$$\min L(S, g(I))$$

Further, we aim to analyze the weight and importance of P within this model, thereby understanding the role of founder personalities in predicting startup success.

2 Data

Before we go into detail on the data specifics, it is crucial to reiterate the primary objectives of this study. Firstly, we aim to develop a model capable of predicting individuals' personalities based on the Five-Factor Model. The subsequent goal is not just to create and assess models that predict startup success by incorporating these personality predictions, but also to critically evaluate the significance of these personality attributes in influencing the outcomes.

2.1 Data Collection

To collect the necessary data, we began by identifying Twitter users who had posted their Five-Factor Model Personality test scores on Twitter. For this purpose, we used Python and through code searched for keywords such as 'My Big 5 Results', among others. Given the multitude of websites offering Big 5 personality tests, we ensured that the users we identified had taken the test from a specific set of websites, which we had pre-determined as trustworthy and reputable. After identifying a set of users, we extracted either their 1,000 most recent tweets or all their tweets if they had published fewer than 1,000. Consequently, we accumulated a dataset comprising 1,708 Twitter users with an average of 724 tweets per user and an average tweet length of 71 words. For the company-specific data, we turned to Crunchbase, a data-as-a-service provider that hosts information about both private and public companies. Although the exact methods Crunchbase uses to collect data are not fully transparent, we considered it a reliable source for our research.

Our data extraction process from Crunchbase was straightforward. We defined our criteria: U.S.-founded companies that received seed funding between 2013 and 2015. Subsequently, we extracted the relevant data. In total, we obtained three distinct datasets: the Companies dataset, the Fundings dataset, and the Founders dataset. Each dataset was interlinked through specific identifiers: the founded company's ID (`organization_id`) and the founders' IDs (`founder_ids`). The relationships between these datasets can be observed in 2.1.

With the Founders dataset in hand, we identified those who had Twitter accounts. We then proceeded to gather all their tweets and combined this data with

the Founders dataset.

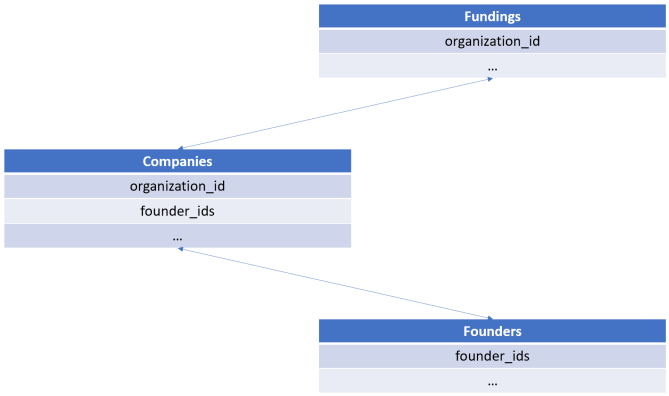

Figure 2.1: Overview of the connections between the three datasets from Crunchbase: Companies, Fundings, and Founders. This diagram only highlights the features used to interconnect the datasets.

2.2 Data Description

Overviews of features and use-cases for the three distinct datasets were provided in tables 2.1-2.3. As highlighted in the tables, not all features were utilized, with some being employed for either feature extraction or target extraction (to define success).

The Companies dataset contained a total of 9,065 rows, the Fundings dataset encompassed 26,642 rows, and the Founders dataset comprised 16,553 rows. After combining the datasets, the resulting set matched the number of rows in the Companies dataset, standing at 9,065. This was because the other two datasets could contain multiple entries related to a single company. As previously mentioned, the Tweets dataset consisted of 1,708 unique users, each contributing between 5 to 1,000 tweets, and their personality scores. The distribution of scores in the Tweet dataset can be seen in figure 2.2. As can be noted, the distribution is fairly left skewed, especially for the trait Openness and Agreeableness.

As touched upon in the Background section of the Introduction Chapter, [11] showed that personalities evolve during adulthood. Considering this, we deliberately chose to include only those companies whose founders tweeted at least five times before their company secured a seed round. This decision reduced the dataset to 299 companies. However, we assessed that this dataset was still sufficiently extensive to yield insightful and meaningful results.

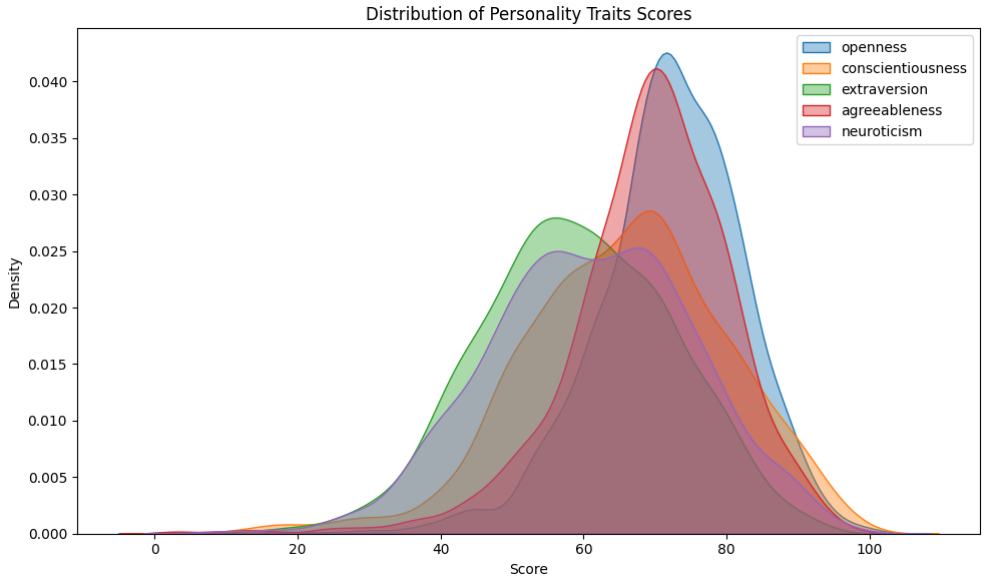

Figure 2.2: Distribution of Personality Scores for the Five Different Personality Traits: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism among the Tweet dataset.

2.3 Data Pre-Processing

As described in the Data Collection section, we only gathered data on US-founded companies, and we thus assumed that most founders were US-born. With this assumption in mind, we deliberately chose to filter out all non-English tweets from the Tweet dataset. To achieve this, we employed the language detector from `nltk`. Before language detection, we pre-processed the tweets: usernames, symbols, hashtags, emojis, and URLs were removed. After this, we used the language detector on each tweet, retaining only those identified as English. However, we preserved their original versions alongside the processed ones. This pre-processing led to a reduced Tweet dataset of 965 observations, highlighting the significant number of non-English tweets.

Handling Missing Data

In the three datasets we collected from Crunchbase, missing data presented a significant challenge. We addressed this primarily through zero replacements or median replacements. For the Fundings Data set, we replaced missing values in the `money_raised_usd` feature with zeros when the `investment_stage` was 'angel'. However, when the `investment_stage` was 'seed', we opted for median

replacement, considering only 'seed' investments. Likewise, we applied median replacement to the `num_investors` feature and to the `date_announced_on` feature, where we determined the median days between two consecutive rounds or between founding dates using the `founded_on_date` feature. Initially, we considered using the `number_trademarks` and `number_patents` features for modeling. Although we later decided against their inclusion (reasons discussed subsequently), we had replaced their missing values with zeros.

Feature Extraction

To construct capable models, we had to extract as much information as possible from the datasets we had collected. At the same time, we had to format the features so that the models could make the best possible predictions. As a result, we conducted supervised feature extraction. For the tweets data, we extracted four features: `nr_tweets`, `avg_tweet_length`, `avg_nr_mentions`, and `avg_nr_emojis`. For the other sets, we started by using the data in the Company dataset and the Fundings dataset. From these, we extracted the following new features: `seed_funding`, `days_to_seed`, `nr_seed_investors`, `seed_investors`, `angel_funding`, `days_to_angel`, all of which made use of `investment_stage`, `date_announced_on`, `money_raised_usd`, and `investor_identifiers` as described in table 2.2. We then appended these features to the Company Data set. Moreover, we took the `linkedin_url`, `facebook_url`, and `twitter_url` features from the original Company Data set and converted them into binary features: `has_linkedin`, `has_facebook`, and `has_twitter`. Finally, we used the Founders Data set to extract the features: `is_serial_founders_avg` and `founders_schools`. For `is_serial_founders_avg`, we looked at the `number_founded_orgs` to determine how many companies each founder had started previously. We then calculated an average score for the founders: founders who had started other companies in the past were given a 1; all others received a 0.

Additionally, the `founders_schools` feature, along with `city`, `region`, and `categories`, was encoded using one-hot-encoding, resulting in a total of 1,534 unique features.

2.4 Features for Modeling

After the pre-processing and feature extraction were completed, all features were appended to the Company Data set, which was then referred to as the Final Company dataset. These features are subsequently utilized to predict two distinct target variables: Series B and Series C funding success.

All the features used for modeling (except for the personality predictions) are shown in table 2.4, comprising 1,544 unique features (including the features from

the one-hot-encodings of the categorical features).

Excluding the categorical features, the final features utilized only a small subset of the available features. This choice might surprise some readers. However, one of the important goals of this thesis was to maintain the credibility of our work by ensuring that the success prediction models were not biased. For instance, the features `num_patents` and `num_trademarks` represented the company's current number of patents and trademarks. Including these would have biased the models since they would have been strong predictors of success, but such data would not be available at the time of an investment decision. The same reasoning applied to features like `revenue_range`, `number_of_employees_range`, and many more shown in tables 2.1-2.3. Consequently, we excluded these from the final feature set.

Feature	Description	Use-case
number_of_founders	Count of company founders.	Modeling
categories	Company's associated categories.	Modeling
category_groups	Broad category groupings.	Modeling
location_identifiers	IDs and names of office locations.	Modeling
founded_on_date	Company's founding date.	Feature Extraction
linkedin_url	Company's LinkedIn page URL.	Feature Extraction
facebook_url	Company's Facebook page URL.	Feature Extraction
twitter_url	Company's Twitter page URL.	Feature Extraction
operating_status	Current operational status.	Target Extraction
acquisition_status	Company's acquisition status.	Not used
website_url	Company's website URL.	Not used
description	Detailed company description.	Not used
number_trademarks	Count of registered trademarks.	Not used
number_patents	Count of registered patents.	Not used
short_description	Brief company description.	Not used
revenue_range	Estimated revenue range.	Not used
number_of_investors	Count of total investors.	Not used
number_of_lead_investors	Count of lead investors.	Not used
number_of_employees_range	Employee count in ranges.	Not used
number_of_funding_rounds	Count of total funding rounds.	Not used
investor_identifiers	IDs and names of investors.	Not used
equity_funding_usd	Total equity funding received.	Not used
last_funding_type	Type of latest funding.	Not used
last_funding_date	Date of latest funding round.	Not used
last_equity_funding_usd	Amount of latest equity funding.	Not used
last_equity_funding_date	Date of latest equity funding round.	Not used
crunchbase_company_rank	Crunchbase ranking for the company.	Not used
funding_stage	Current funding stage.	Not used
last_funding_usd	Latest round's funding amount.	Not used
number_funding_rounds	Total funding rounds count.	Not used
acquisition_price_usd	Acquisition price, if applicable.	Not used
ipo_status	Company's IPO status.	Not used
company_valuation_range	Estimated company valuation.	Not used
funding_usd	Total funding received.	Not used

Table 2.1: Overview of all initial features available from the Crunchbase Companies Data Set. Use-case descriptions: 'Feature Extraction' denotes features used to generate additional features, 'Modeling' refers to features directly used in model training, 'Target Extraction' refers to features that define the target variable for the predictive model, and 'Not used' indicates features not utilized in the project.

Feature	Description	Use-case
number_of_investors	Total count of investors.	Modeling
investor_identifiers	Identifiers for the investors.	Modeling
lead_investor_identifiers	Identifiers for the lead investors.	Modeling
date_announced_on	Date when the funding was announced.	Feature Extraction
investment_type	Type of investment (e.g., Equity, Debt-financing, etc.).	Feature Extraction
money_raised_usd	Amount of money raised in USD.	Feature Extraction
investment_stage	Stage of investment (e.g., Seed, Series A, etc.).	Feature Extraction
pre_money_valuation_usd	Pre-money valuation of the company in USD.	Feature Extraction
is_equity_funding	Indicator if the funding was equity funding.	Not used

Table 2.2: Overview of all initial features available from the Crunchbase Fundings Data Set. Use-case descriptions: 'Feature Extraction' denotes features used to generate additional features, 'Modeling' refers to features directly used in model training, and 'Not used' indicates features not utilized in the project.

Feature	Description	Use-case
number_founded_orgs	Total count of organizations founded.	Modeling
attended_schools	Schools attended by the founder.	Modeling
gender	Gender of the founder.	Modeling
facebook_url	Facebook page URL of the founder.	Feature Extraction
linkedin_url	LinkedIn profile URL of the founder.	Feature Extraction
twitter_url	Twitter handle URL of the founder.	Feature Extraction
primary_org	Primary organization associated with the founder.	Connecting Data
current_org	Current organizations associated with the founder.	Connecting Data
name	Name of the founder.	Not used
crunchbase_person_rank	Crunchbase rank of the founder.	Not used
location_identifier	Identifier for the founder's location.	Not used

Table 2.3: Overview of all initial features available from the Crunchbase Founders Data Set. Use-case descriptions: 'Feature Extraction' denotes features used to generate additional features, 'Modeling' refers to features directly used in model training, 'Connecting Data' means features used to connect different datasets, and 'Not used' indicates features not utilized in the project.

Feature	Type	Description
seed_funding	Continuous	$USD in seed funding.
days_to_seed	Continuous	Number of days from founding to seed round.
nr_seed_investors	Discrete	Count of investors at seed stage.
seed_investors	Discrete	One-hot encoded list of seed stage investors (744 unique).
angel_funding	Continuous	Amount in $USD raised in angel funding.
days_to_angel	Continuous	Days from founding to angel round, if funding was raised.
num_founders	Discrete	Count of company founders.
is_serial_founders_avg	Continuous	Average count of serial founders among the founders.
founders_schools	Categorical	One-hot encoded list of founder's attended schools (357 unique).
city	Categorical	City where company was founded (98 unique cities).
region	Categorical	Region or US state where company was founded (30 unique regions).
has_twitter	Discrete	Indicator if company has/had a Twitter page.
has_facebook	Discrete	Indicator if company has/had a Facebook page.
has_linkedin	Discrete	Indicator if company has/had a LinkedIn page.
categories	Categorical	Crunchbase Product Categories associated with the company (305 unique).

Table 2.4: Overview of final features used for model training. The table showcases the feature name, its type, and a brief description.

Feature	Type	Description
tweets	Text	Raw tweet texts.
nr_tweets	Discrete	Tweet count by user.
avg_tweet_length	Continuous	Average tweet length.
avg_nr_mentions	Continuous	Average mentions per tweet.
avg_nr_emojis	Continuous	Average emojis per tweet.
openness	Continuous	Average prediction score for the Openness trait of the founding team.
conscientiousness	Continuous	Average prediction score for the Conscientiousness trait of the founding team.
extroversion	Continuous	Score for the Extroversion trait.
agreeableness	Continuous	Score for the Agreeableness trait.
neuroticism	Continuous	Score for the Neuroticism trait.

Table 2.5: Overview of features used for training the personality models (before embedding the tweets). The table showcases the feature name, its type, and a brief description.

3 Artificial Intelligence

This section offers a foundational understanding of key theoretical concepts relevant for this thesis. We touch upon the expansive domain of Artificial Intelligence, delving into Machine Learning, Natural Language Processing, and the nuances of Deep Learning. Text embeddings and their significance in contextualizing information are explored, followed by a review of various modeling techniques. We will also address the challenges and solutions associated with high-dimensional and unbalanced datasets. This understanding is crucial for comprehending the methods used in this study.

3.1 Background

Artificial Intelligence (AI) is a broad field in computer science that aims to simulate and enhance human cognitive functions within machines. Its purpose is to create systems that can perform tasks typically requiring human intelligence. These tasks include, but are not limited to, image recognition, speech recognition, decision-making, and natural language understanding [21].

The author in [21] defined AI as "The study of the computations that make it possible to perceive, reason, and act." This definition underscores machines' abilities to understand their environment, make decisions, and pursue specific goals.

Figure 3.1 illustrates the various subfields and applications of AI.

Machine Learning

Machine learning (ML) is a sub-field of AI that focuses on creating models and algorithms that allow machines to learn from and make decisions or predictions based on data [22]. ML tasks can be divided into three categories: supervised learning, unsupervised learning, and reinforcement learning [23].

In supervised learning, models are trained on data that contain explicit target variables or labels corresponding to some input data[22]. As an analogy, consider training an ML model to distinguish apples from oranges using a set of images. In the case of supervised learning, each image is labeled as either an apple or orange, to help the model learn.

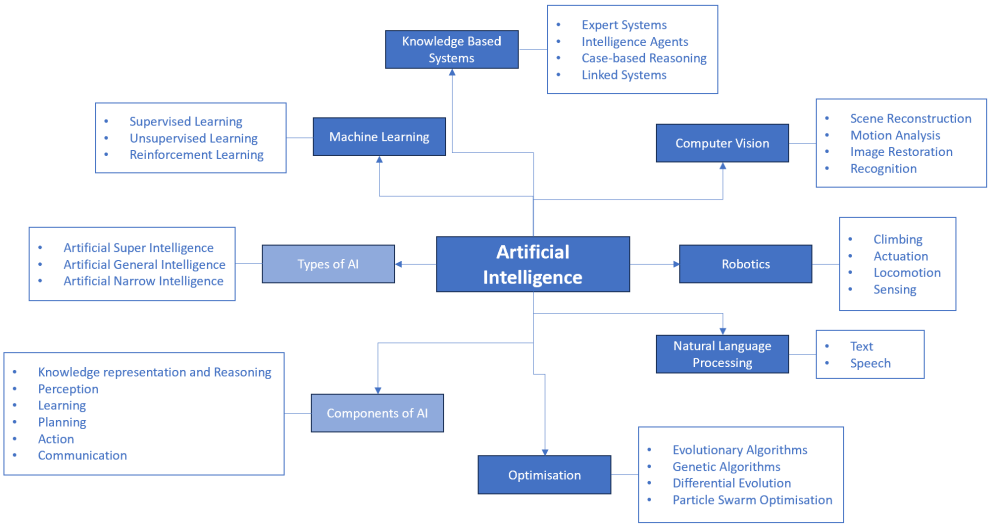

Figure 3.1: Illustration of various subfields and applications of AI. Derived from [1].

Unsupervised learning, on the other hand, seeks to identify inherent structures or patterns within data that has not been labeled. Common methods used in unsupervised learning include K-Mean Clustering, Principal Component Analysis (PCA), and Autoencoders [24].

Lastly, reinforcement learning operates on a principle of trial and error, where an agent learns to navigate within a given environment, getting rewarded or punished for its actions and adjusting thereafter [23]. For instance, consider a computer agent learning to play the game Snake. The agent receives rewards for consuming "snacks" and incurs penalties for colliding with its tail. Through this process of trial-and-error, the agent can learn the optimal strategy to successfully navigate the game environment.

In general, ML is a very important part of AI as it enables systems to automatically learn and improve from experience without being explicitly programmed [25].

Natural Language Processing

Natural Language Processing (NLP) is a sub-field within AI that focuses on the interaction between computers and humans through natural language [26]. Leveraging ML techniques, NLP aims to enable computers to understand, interpret, and utilize human language in meaningful ways.

NLP applications encompass a wide range of tasks including machine translation, sentiment analysis, named entity recognition, and speech recognition. A core

aspect of these tasks involves transforming raw, unstructured text into structured, numerical representations. These representations, also known as embeddings, enable computers to understand the semantic meaning of text [27].

NLP employs ML techniques to convert vast amounts of human language data into numerical representations. This transformation allows ML models to analyze and derive insights from the data, thereby enhancing the capabilities of AI systems in handling language-based tasks [26].

These embeddings, coupled with ML techniques, unlock a myriad of possibilities in understanding and deriving insights from natural language data. This combination forms the backbone of NLP [27].

Deep Learning and Neural Networks

Deep learning, a subset of ML, harnesses deep neural networks to model and process data structures within extensive datasets [28]. While traditional ML methods often require manual feature engineering, deep learning can automatically extract relevant features from raw data [29]. As such, it has fostered breakthroughs in several areas, notably computer vision via convolutional neural networks [30], and natural language processing through the utilization of recurrent neural networks and transformers [31].

However, the architecture and design of deep neural networks sometimes lead to challenges surrounding interpretability [29]. Additionally, these models require abundant data for training and considerable computational power [28]. Yet, within AI, deep learning stands as a critical tool, with its ability to engage with vast datasets and learn complex patterns from data, which has considerably advanced current AI systems [29].

3.2 Text Embeddings

Text embeddings are numerical vector representations of text [27]. These are fundamental to natural language processing (NLP) as they allow for machines to interpret and understand meaningful relationships within text [26]. There are numerous methods to construct text embeddings, some of which will be described below.

Tokenization

An important step in creating text embeddings is to turn raw text into words or tokens. A token is a single unit of meaning, most typically a word [26]. However, depending on the specifies of the NLP task, a token could also refer to a single character, a syllable a sentence or even an entire document of text [26]. While not

all text embeddings techniques explicitly require this step, it generally forms an integral part of NLP. The tokenization of the sentence "This is a very large tree" can be seen in Figure 3.2, where each token refers to a single word.

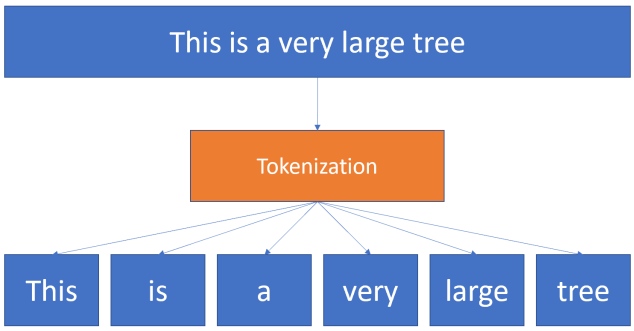

Figure 3.2: Tokenization example of the sentence "This is a very large tree".

Encodings

Once raw text has been tokenized, the subsequent step involves encoding these tokens to produce embeddings [27]. To illustrate, let us use the one-hot encoding method on our previous example sentence, "This is a very large tree." In one-hot encoding, each unique token in the corpus is associated with a distinctive binary vector [27]. In this vector, one position corresponds to '1' (denoting the presence of the token), while all other positions are marked as '0'. Table 3.1 shows the one-hot encoding method. While one-hot encoding provides a simple, straightforward way

Token	One-hot vector
"This"	[1, 0, 0, 0, 0, 0]
"is"	[0, 1, 0, 0, 0, 0]
"a"	[0, 0, 1, 0, 0, 0]
"very"	[0, 0, 0, 1, 0, 0]
"large"	[0, 0, 0, 0, 1, 0]
"tree"	[0, 0, 0, 0, 0, 1]

Table 3.1: One-hot encoding of the tokens in the sentence "This is a very large tree."

to represent text numerically, it fails to capture any semantic relationships between words, which is often crucial in NLP tasks. In search for techniques that can better account for semantic relevancy, we turn to more advanced methods, such as Term Frequency-Inverse Document Frequency (TF-IDF), and Transformers.

Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that determines the relevancy of a word or a phrase (term) in a document within a corpus [32]. The relevancy increases proportionally to how many times a given term appears within a document but is offset by the frequency of the term within the corpus. Thus, TF-IDF assumes that terms which appear less frequently in a set of documents carry more information, and hence gets assigned greater weights. Because of this assumption, TF-IDF is a very common and effective technique for tasks such as information retrieval and text mining. The Term Frequency (TF) for a specific term t in a document d is calculated as:

$$TF(t, d) = \frac{n_{t,d}}{N_d}$$

where $n_{t,d}$ is the number of times term t appears in document d and N_d is the total number of terms in document d. The Inverse Document Frequency (IDF) for term t in the corpus of documents D is:

$$IDF(t, D) = \log\left(\frac{N_D}{n_{t,D}}\right)$$

where N_D is the total number of documents in corpus D and $n_{t,D}$ is the number of documents with term t in corpus D. By combining TF and IDF, we get the final formula for Term Frequency-Inverse Document Frequency (TF-IDF):

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Although TF-IDF effectively captures term relevancy, it lacks contextual understanding and does not capture semantic relationships between words. To address these limitations, newer models like BERT, which consider context and capture semantic nuances, have been developed [33]. This will be the focus of the following section.

Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is an NLP model developed and open source by Google in 2018 [33]. As the name suggests, BERT is a transformer-based model, meaning it is grounded in the transformer architecture seen in Figure 3.3, a model architecture based on the attention mechanism. The purpose of the attention mechanism is to weigh the influence of different input elements when producing an output. In simpler terms, when BERT generates an output (such as translating a word from one language to another) the

attention mechanism helps the model decide which input elements to "pay attention to", given the surrounding input elements. The formula for attention (more formally, Scaled Dot-Product Attention) is given by the following,

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where Q (Query), K (Key) and V (Value) are vectors derived from the input data, and d_k refers to the dimensionality of Q and K. The attention score determines how much focus to place on a given element, where the score is scaled down by the square root of the dimension of the query and key given by the input element for computational stability. Then a softmax function is applied, to have the attention scores sum to one, which makes the score interpretable as a probability. The softmax function is defined as follows,

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=i}^{N} e^{z_j}}.$$

Apart from the transformer architecture and attention mechanism, another important aspect of BERT, is the Bidirectional Nature. Instead of reading text input in one direction (i.e., left-to-right, or right-to-left), BERT reads the text in both directions at once, allowing it to understand the context of a word based on all its surroundings.

BERT generates context-sensitive embeddings. The length of the embeddings varies depending on the sentence or the word's context. This means that the same word or sentence can have different embeddings based on its surroundings. These embeddings are produced by the encoder part of the BERT model and are essentially the model's hidden states.

For tasks such as classification, a common approach is to utilize the embedding of the 'CLS' token. This token is appended at the beginning of each input text and is trained to encapsulate the overall context of a sentence. Hence, it serves as a representative vector that carries information about the text. Typically, researchers and practitioners use the last hidden state of the 'CLS' token because it has accumulated information from all previous layers, making it the most informative and rich representation. This state has proven to be effective in downstream tasks since it captures both low-level features and high-level semantic information of the text, making it a comprehensive representation.

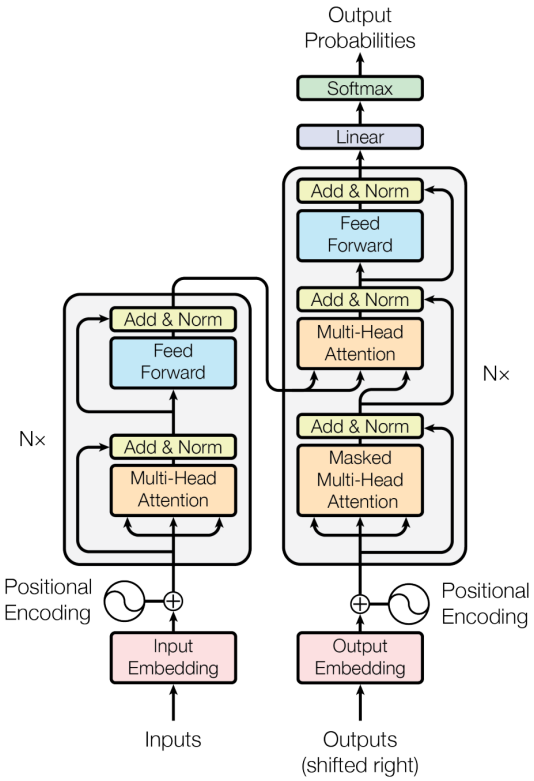

Figure 3.3: The model architecture of a Transformer

3.3 Models

Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable (also known as the 'outcome' or 'response' variable) and one or more independent variables (also known as 'explanatory' or 'predictor' variables) [34].

Given a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector ($n >> p$). The general form of the model is specified as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i \text{ with } \epsilon_i \sim N(0, \sigma^2),$$

for $i = 1, \ldots, n$.

If we were to formulate this using matrix notation, we get:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \text{ with } \boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_p \end{bmatrix}.$$

The linear regression model undertakes four major assumptions, that is,

1. Linearity: $\mu = [E(y_i|\boldsymbol{x}_i)]_{n\times 1} = \mathbf{X}\boldsymbol{\beta}$,
2. Independence: ϵ_i's are independent of each other,
3. Homoscedasticity: ϵ_i's have equal variance σ^2,
4. Normality: ϵ_i's are normally distributed.

The goal with linear regression is to identify the set of independent variables that influence the dependent variable. To discover this dependency, one must carry out the model estimation, which is the process of calculating the best, or optimal, fitting line through the observed data D. Typically, model estimation is carried out through either Ordinary Least Squares (OLS) or Maximum Likelihood (ML). If the four assumptions above are met, the OLS and ML estimate coincide, because under this assumption the likelihood function for a linear regression model is maximized at the same point where the sum of squared residuals (which is what OLS minimizes) is minimized. For OLS we have,

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Differentiating $Q(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ and setting the equation to $\mathbf{0}$, we get the least squares estimator (LSE) $\tilde{\boldsymbol{\beta}}$ that exists as a unique solution if the matrix $\mathbf{X}^T\mathbf{X}$ is invertible. It is given by:

$$\tilde{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

For ML we have,

$$L(\boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-n/2}\exp\left(-\frac{\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}}{2\sigma^2}\right) \cdot \exp\left(\frac{\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}}{\sigma^2} - \frac{\mathbf{y}^T\mathbf{y}}{2\sigma^2}\right),$$

and the log-likelihood is given by,

$$l(\boldsymbol{\beta}, \sigma^2) = -n/2 \cdot \log(2\pi) - n/2 \cdot \log\sigma^2 - (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})/(2\sigma^2).$$

By taking the first derivative with respect to $(\boldsymbol{\beta}, \sigma^2)$ and setting it to $\mathbf{0}$ we get the ML estimator (MLE), which for $\boldsymbol{\beta}$, and already stated, is the same as its LSE.

Logistic Regression

Logistic regression is a statistical method that is primarily used for binary classification problems, but it can be adapted for multiclass problems as well [35]. In the binary case, it models the probability that the dependent variable (or 'outcome') is a specific class given the values of the independent variables (also known as 'features' or 'predictors').

Given a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where $y_i \in \{0, 1\}$ is the ith binary response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector $(n >> p)$. The general form of the model is specified as:

$$\log\left(\frac{p(x_i)}{1 - p(x_i)}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

for $i = 1, \ldots, n$, where $p(x_i)$ represents the probability of the outcome being 1 given the predictor values x_i.

Using matrix notation, we can write the logistic regression model as:

$$\log\left(\frac{\mathbf{p}}{1 - \mathbf{p}}\right) = \mathbf{X}\boldsymbol{\beta},$$

where $\mathbf{p}$ is a vector of outcome probabilities, $\mathbf{X}$ is the matrix of predictors, and $\boldsymbol{\beta}$ is the vector of coefficients.

Logistic regression makes the following assumptions:

1. Linearity: The log odds ratio is a linear function of the predictor variables, $\log(\frac{p}{1-p}) = \mathbf{X}\boldsymbol{\beta}$.

2. Independence: The observations y_i's are independent of each other.

3. Absence of multicollinearity: There is no perfect linear relationship between the predictor variables.

4. Large Sample Size: logistic regression requires a larger sample size to ensure reliable results.

The goal with logistic regression, like linear regression, is to identify the set of independent variables that influence the dependent variable. However, unlike linear regression which uses either OLS or ML for model estimation, logistic regression typically uses the method of ML. The log-likelihood function for logistic regression is given by,

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))$$

The MLEs of the parameters are the values of $\boldsymbol{\beta}$ that maximize this log-likelihood function. However, unlike linear regression, there is no closed-form solution for the MLEs in logistic regression. Instead, numerical methods such as Newton-Raphson or gradient ascent are typically used to find the MLEs.

Decision Trees

Decision trees serve as a fundamental building block for ensemble models, like random forest and XGBoost, which are deployed in this project.

A decision tree is a hierarchical, non-parametric model that splits the feature space into several partitions following a tree-like structure [36]. This structure comprises of internal nodes, branches, and leaf nodes. The internal nodes represent features or predictors, each branch signifies a decision rule based on the value of the associated feature, and each leaf node indicates an outcome or decision (usually the mean or mode of the target variable for the observations falling in that partition). For clarity and better understanding, a specific example of a decision tree, classifying fruits based on their characteristics, is illustrated in Figure 3.4.

The tree construction begins at the root node, which uses the most predictive feature to make the initial binary split in the data. The selection of this feature and the value at which the split is made is typically based on an impurity measure like Gini Index or Information Gain.

Formally, suppose we have a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector. Let S be a subset of D represented by a node. If S is pure (all y_i in S are the same) or another stopping criterion is met, S becomes a leaf and is assigned the most common value of y_i in S. Otherwise, the feature x_j and the value t that minimizes the impurity in the resulting subsets is chosen, and the data is split into two subsets $S_{left} = \{(y_i, \boldsymbol{x}_i) \in S | x_{ij} \leq t\}$ and $S_{right} = \{(y_i, \boldsymbol{x}_i) \in S | x_{ij} > t\}$. The process is then repeated for S_{left} and S_{right}.

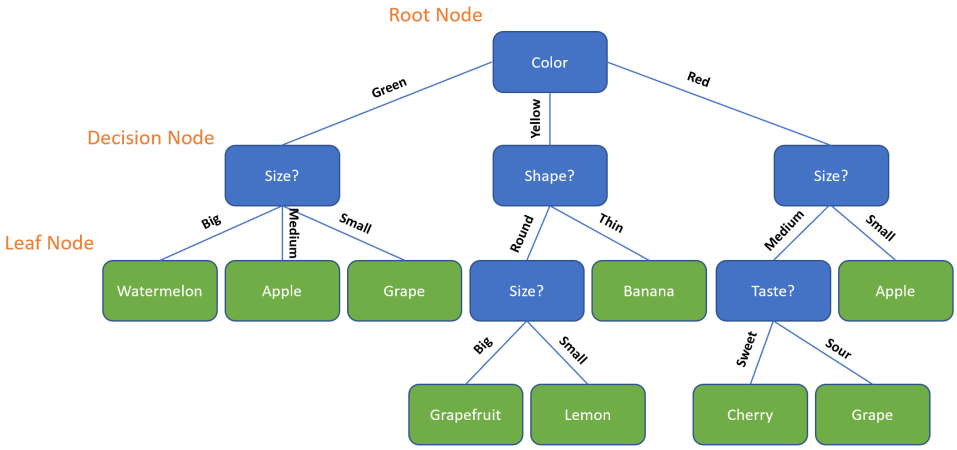

Figure 3.4: Diagrammatic representation of a decision tree used for classifying fruits based on their color, shape, size, and taste. The tree structure provides clear decision rules at each node to categorize the fruits, ultimately leading to leaf nodes that define the type of fruit.

This partitioning process, also known as binary recursive partitioning, is performed recursively, creating an increasing number of mutually exclusive and exhaustive groups. The partitioning continues until a predefined stopping criterion is met. Typical stopping criteria include a minimum node size, a maximum tree depth, or no further decrease in the impurity.

Bootstrapped Aggregating

Bootstrapped Aggregating (Bagging) plays a significant role in several ensemble-based ML models, including the random forest algorithm explored later in this document. Bagging is a technique that enhances the stability and accuracy of ML algorithms and is particularly effective in reducing overfitting by mitigating the variance of the model [37].

Bagging algorithm operates by creating multiple subsets of the original dataset, with replacement. Suppose we have a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector. For a Bagging algorithm with B iterations, we generate B different bootstrapped datasets $D_1, D_2, \ldots, D_B$, each of size n, by sampling with replacement from D.

For each bootstrapped dataset D_b, we train a separate instance of the base model (e.g., a decision tree), resulting in B different models $M_1, M_2, \ldots, M_B$. The final model output for a new input $\boldsymbol{x}_{new}$ is obtained by averaging the predictions

(for regression) or voting (for classification) from all B models. In mathematical terms, for regression, the bagging estimator is given by:

$$f_{bag}(\boldsymbol{x}_{new}) = \frac{1}{B}\sum_{b=1}^{B} M_b(\boldsymbol{x}_{new}),$$

and for classification, it is given by:

$$f_{bag}(\boldsymbol{x}_{new}) = \text{argmax}_k \sum_{b=1}^{B} I(M_b(\boldsymbol{x}_{new}) = k),$$

where I is the indicator function.

Random Forest

Random forest is a widely used ML algorithm that utilizes an ensemble of decision trees and operates based on the principles of bagging and feature randomness [38]. It can be employed for both regression and classification tasks.

Suppose we have a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector. For a random forest algorithm with B iterations, we generate B different bootstrapped datasets $D_1, D_2, \ldots, D_B$, each of size n, by sampling with replacement from D.

However, random forest introduces an additional level of randomness. Instead of considering all predictors at each split in the decision tree, only a random subset of the predictors is considered. The optimal split is then found among this subset of predictors.

For each bootstrapped dataset D_b, we train a separate instance of the base model (a decision tree), resulting in B different models $M_1, M_2, \ldots, M_B$. The final model output for a new input $\boldsymbol{x}_{new}$ is obtained by averaging the predictions (for regression) or voting (for classification) from all B models. In mathematical terms, for regression, the random forest estimator is given by:

$$f_{RF}(\boldsymbol{x}_{new}) = \frac{1}{B}\sum_{b=1}^{B} M_b(\boldsymbol{x}_{new}),$$

and for classification, it is given by:

$$f_{RF}(\boldsymbol{x}_{new}) = \text{argmax}_k \sum_{b=1}^{B} I(M_b(\boldsymbol{x}_{new}) = k),$$

where I is the indicator function.

By introducing this additional layer of randomness, random forest tends to exhibit robust performance and is generally less prone to overfitting than a single decision

eXtreme Gradient Boosting (XGBoost)

Gradient Boosting Machine (GBM) is a powerful ensemble ML technique that builds a sequence of models, typically decision trees, by iteratively adding predictors to minimize the loss function. Each new tree helps to correct the errors made by the previously trained tree [39].

eXtreme Gradient Boosting (XGBoost) is an optimized and more efficient variant of the GBM algorithm. It employs the gradient boosting framework to produce this sequence of models in a stage-wise fashion [40]. XGBoost is frequently employed for both classification and regression tasks.

Given a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector. At each iteration m, a new model f_m is fit to the negative gradient of the loss function L evaluated at the previous iteration's prediction. In mathematical terms, for a differentiable loss function $L(y, \hat{y})$, the model f_m at iteration m is fit to:

$$r_{im} = - \left[\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} \right]_{\hat{y}_i = \hat{y}_{i,m-1}},$$

and then the update is given by:

$$\hat{y}_{i,m} = \hat{y}_{i,m-1} + \nu \cdot f_m(\boldsymbol{x}_i),$$

where ν is the learning rate.

XGBoost improves upon the standard Gradient Boosting Machine (GBM) algorithm in several ways. Firstly, it uses parallel processing, making it more efficient for large-scale tasks. Secondly, XGBoost includes regularization parameters (L1 and L2), which helps prevent overfitting. Specifically, XGBoost minimizes the following objective:

$$Obj = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

where K is the number of trees, $\Omega(f_k)$ is a regularization term that penalizes the complexity of model f_k.

Moreover, unlike GBM which stops splitting a node as soon as it encounters a negative loss, XGBoost splits up to the maximum depth specified and then prunes the tree backwards (a method known as "post-pruning"). Finally, XGBoost is designed to efficiently handle sparse data and missing values.

Neural Networks

A neural network is a computational model that is inspired by the structure of biological brain networks. It consists of interconnected layers of nodes, known as neurons, which are designed to imitate the neurons in a brain. Each of these neurons takes a set of inputs, applies a weighted linear transform, and then applies an activation function.

Given a dataset $D = \{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$ where y_i is the ith response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p$ is the corresponding predictor vector. A neuron j in the neural network calculates its output o_j as follows:

$$o_j = f\left(\sum_{i=1}^{m} w_{ij} x_i + b_j\right),$$

where w_{ij} is the weight of the connection from input x_i to neuron j, b_j is a bias term, m is the number of inputs, and f is the activation function, which introduces non-linearity into the model.

Neural networks typically consist of one input layer, one output layer, and one or more hidden layers. The input layer receives input features, and the output layer produces the prediction. Each neuron in the hidden layer transforms the values from the previous layer with its weights, bias, and activation function. For binary classification tasks, the final output layer typically uses a sigmoid function to produce a probability, and a threshold (e.g., 0.5) determines the class prediction. For multi-class classification tasks, the final output layer uses a softmax function to produce probabilities for each class, and the class with the highest probability is selected as the prediction. For regression tasks, the output layer usually contains a single neuron, and the output of this neuron is the predicted value.

The weights and biases of a neural network are learned by minimizing a loss function that represents the difference between the predicted and actual output. This is usually done using a method known as backpropagation in combination with an optimization method, such as stochastic gradient descent (SGD) [29].

An illustration of the architecture of a neural network can be seen in Figure 3.5.

3.4 Dimensionality Reduction

When dealing with large datasets with numerous features (or predictors), a common practice is to reduce the number of features, a process known as dimensionality reduction. There are several benefits to dimensionality reduction. This includes reduced model complexity, lesser storage required for computation leading

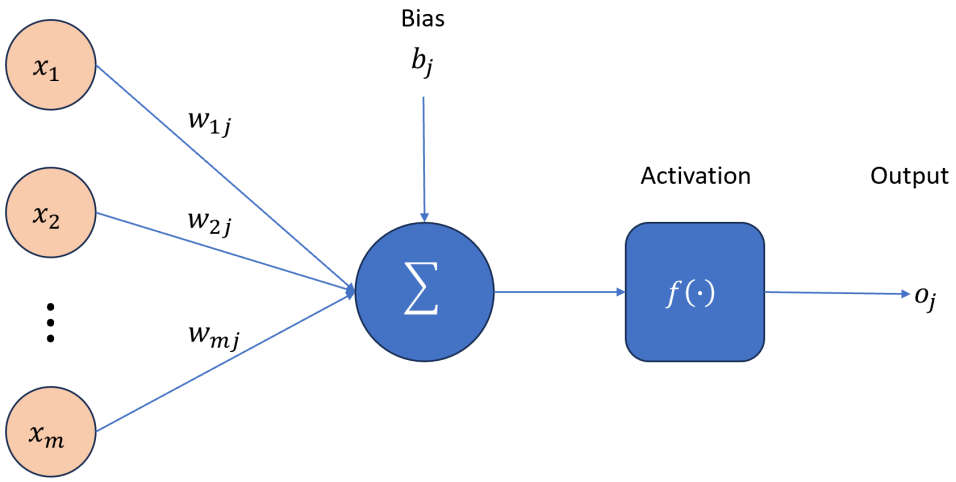

Figure 3.5: Graphical illustration of a neuron's computation in a neural network, highlighting the input neurons, weights, bias, and the activation function. The computation is driven by the formula: $o_j = f\left(\sum_{i=1}^{m} w_{ij}x_i + b_j\right)$.

to faster training times, and decreased noise in the data, which could potentially improve model accuracy.

However, the challenge lies in deciding which and how many features to eliminate without losing significant information. Several techniques have been developed to address this, including Principal Component Analysis (PCA) and Truncated Singular Value Decomposition (SVD), which will be discussed in the following subsections [22].

Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method used for dimensionality reduction. The method employs an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components. The first principal component accounts for the largest possible variance in the dataset, and each succeeding component in turn has the highest variance under the constraint that it is orthogonal to the preceding components. The number of principal components is less than or equal to the number of original variables.

To apply PCA, one must first center the data. Let's define the centered data matrix $\mathbf{Z}$ as:

$$\mathbf{Z} = \mathbf{X} - \mu, \tag{3.1}$$

where $\mathbf{X}$ is the original data matrix and μ is a matrix where each row is the mean of $\mathbf{X}$.

The covariance matrix of the centered data is then given by:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{Z}^\mathsf{T}\mathbf{Z}, \tag{3.2}$$

where n is the number of observations.

The principal components are the eigenvectors of the covariance matrix $\mathbf{C}$. Mathematically, this is represented by the eigenvalue equation:

$$\mathbf{C}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \tag{3.3}$$

where λ_i are the eigenvalues and the $\mathbf{u}_i$ are the eigenvectors of $\mathbf{C}$, ordered such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ [41].

Truncated Singular Value Decomposition

Singular Value Decomposition (SVD) is a matrix decomposition method for reducing a matrix to its constituent parts to make subsequent matrix calculations simpler. For the sake of demonstration, we will only focus on real numbers.

Given a matrix $\mathbf{X}$, the singular value decomposition of $\mathbf{X}$ is a pair of orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$, with a diagonal matrix $\mathbf{S}$, such that:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}, \tag{3.4}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices (i.e., $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}$) and $\mathbf{S}$ is a diagonal matrix consisting of the singular values of $\mathbf{X}$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are the left-singular vectors and right-singular vectors of $\mathbf{X}$, respectively [41].

Truncated SVD, as the name suggests, is a variant of SVD where we only keep the largest singular values and corresponding vectors. This method is effective for reducing the dimensionality of data and works well when there are a small number of meaningful singular values.

In the truncated SVD, the decomposition of $\mathbf{X}$ takes the form:

$$\mathbf{X} \approx \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\mathsf{T}, \tag{3.5}$$

where $\mathbf{U}_k$, $\mathbf{S}_k$ and $\mathbf{V}_k$ contain only the first k columns, singular values, and rows from $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$, respectively [41].

In essence, truncated SVD allows us to approximate the original matrix using fewer dimensions, which can be beneficial in a variety of ML applications for dealing with high-dimensional data.

3.5 Unbalanced Data

The dominance of one class can lead to poor performance of classifiers, especially towards the minority class, as most algorithms tend to maximize overall accuracy, which can be done by merely predicting the majority class [42]. Therefore, it is essential to use techniques that can address this issue effectively. In this context, two of the most prominent methods employed to tackle class imbalance, SMOTE and ADASYN, are detailed.

Synthetic Minority Over-sampling Technique (SMOTE)

The Synthetic Minority Over-Sampling Technique (SMOTE) addresses the issue of class imbalance by creating synthetic samples in the feature space. Rather than just duplicating examples from the minority class (which could result in overfitting) or removing examples from the majority class (risking the loss of valuable information), SMOTE works as follows: For a given instance from the minority class, it identifies a few nearest neighbors from the same class. It then chooses one of these neighbors and creates a new instance that lies in between the chosen instance and its neighbor.

In essence, SMOTE "draws lines" between existing instances and fills in new points on those lines. This method not only increases the number of instances in the minority class but also makes the decision boundary more general, avoiding an over-specific boundary that might not perform well on new data [43].

Adaptive Synthetic Sampling (ADASYN)

Adaptive Synthetic Sampling (ADASYN) is another over-sampling approach that builds on the idea of SMOTE but introduces a density distribution, where more synthetic data is generated for minority classes that are harder to learn. The algorithm calculates the class distribution density for each data point in the minority class and then generates synthetic instances accordingly. This means that those minority instances that lie closer to the decision boundary (and hence are harder to learn) will have more synthetic examples generated. The adaptive nature of ADASYN makes it particularly useful in scenarios where the class distribution is not merely imbalanced, but the minority class instances are also scattered and have varied densities [44].

4 Methods

In the following section, we outline the steps and strategies implemented for this study. Starting with the process of formatting data and converting text into embeddings, we detail our approach to model construction, both for predicting personality and startup success. We also delve into the necessary procedures of hyperparameter tuning and methods for evaluating model performance. Lastly, a brief note on the software tools used for implementation will be provided.

4.1 Formatting Data for Learning

TF-IDF Embeddings

The Term Frequency Inverse Document Frequency (TF-IDF) method, detailed earlier in the Theory section, was employed as the primary approach to convert tweets into embeddings. We tokenized words, capturing both unigrams and bigrams. For building the vocabulary, a subset of the data, distinct from the sets reserved for validation or testing, was used. To strike a balance between reducing computational overhead and preserving important information, only the 10,000 most frequent n-grams from this subset were retained in the final matrix. Consequently, the produced matrix had dimensions $M \times 10,000$, where M represented the number of unique Twitter users in our dataset.

BERT Embeddings

For the second approach to convert the collected tweets into text embeddings, the Bidirectional Encoder Representations from Transformers, commonly known as BERT, optimized specifically for English text, was utilized. A pre-trained BERT model and its associated tokenizer, available from the Hugging Face's transformers library, were employed.

Each tweet from a user was fed individually into the BERT model. This process generated contextual embeddings for every token within the tweet, producing a sequence of high-dimensional vectors, with each vector representing a token in the tweet.

To represent each tweet, the last hidden state output from BERT was taken and transformed into a singular vector through a max-pooling operation. As outlined in the Theory section, this max-pooling method identifies the maximum value across each dimension from all token vectors in the tweet, culminating in a single vector reflecting the entire tweet's content. Subsequently, embeddings of all tweets from a specific user were averaged, yielding a singular vector representing that user's collective tweets.

This procedure was repeated for all users' tweets. Finally, all the generated embeddings were compiled into a single matrix for downstream prediction tasks.

Truncated SVD

For the TF-IDF embedding matrix, truncated SVD was employed to reduce its dimensionality. The explained variance for different numbers of components (or lengths) was analyzed, and a decision was made based on the number of components that explained 95 percent of the variance in the matrix. By doing this, the most crucial information from the input was retained while eliminating any noise that might adversely affect predictions later.

Data Partitioning

To train, validate, and evaluate our models, we partitioned both our datasets: the Final Company data and the Tweet dataset.

The Tweet dataset was divided into three separate subsets: training, validation, and testing. This partitioning was done randomly, but once it was established, it was consistently maintained throughout the training and evaluation phases of the various models. The testing subset constituted 20% of the total data. The remaining 80% was split between training and validation. Specifically, the validation subset made up 20% of this split (equivalent to 16% of the total data), while the training subset constituted the remaining 80% (equivalent to 64% of the total data).

The Final Company dataset, on the other hand, was divided into two subsets: training and testing, following an 80/20% split. We did not reserve data for a validation set in the Final Company dataset. This decision was based on our use of Random Search with Cross-Validation for parameter selection.

Handling Unbalanced Classes

The Final Company dataset was unbalanced. It comprised 40 startups that had raised a Series B round ($\approx$ 13.4%) and only 10 that had progressed to a subsequent Series C round ($\approx$ 3.3%). Due to this imbalance, for all models, we adopted Synthetic Minority Over-sampling Technique (SMOTE) for oversampling

the minority class in the training data. It is important to note that SMOTE was only applied to the training data. This measure ensured that our models did not simply optimize for predictions on the predominant class (failure) and could discern genuine patterns in the data. We also explored the use of the Adaptive Synthetic (ADASYN) algorithm for managing unbalanced data. However, based on our evaluations, we opted for SMOTE.

Standardization

All input data, encompassing both tweets and company data, was standardized to have a mean of zero and a standard deviation of one. For the tweets, standardization primarily aimed to improve convergence speed during model training. For the company data, the goal was to ensure consistent and unbiased comparisons across various models.

4.2 Constructing the Personality Prediction Models

Before describing the specific models constructed for predicting the personality traits of the Five-Factor Model, we wanted to outline the common procedure used for all personality models. Initially, all models were created in pairs: one using the TF-IDF embeddings as inputs and the other using the BERT embeddings as input. For each unique set of input and model type, five distinct models were constructed with different parameters to predict each trait of the Five-Factor model. While this was our general approach, there was one exception: the Naive Model, which had its unique construction and purpose.

Naive Model

To set a benchmark for the performance of our advanced personality models, we incorporated a naive personality prediction model, referenced as the Naive Model. This basic model served as a foundational reference, enabling us to establish a standard of comparison and assess the additional benefits our more sophisticated models provided.

In constructing our Naive Model, we calculated the mean personality score for each of the Five-Factor traits using the training data. For each trait, its corresponding mean score was then applied as the predicted value for every observation in the test dataset, disregarding individual differences in the data.

Feed-forward Neural Network

After establishing our baseline with the Naive Model, we proceeded to develop more complex models, beginning with feed-forward neural networks.

The initial architecture for these neural networks utilized three hidden layers with a descending number of neurons, starting at 256. Each layer employed the ReLU activation function, while the final output layer used a linear activation function.

During training, we employed the Adam optimization algorithm. The models were trained with a batch size of 64, using the Mean Squared Error (MSE) as the loss function, over 150 epochs.

Random Forest

For our third personality prediction model, we turned to the random forest approach. This decision was motivated by the model's ability to efficiently manage our sizable dataset and its aptitude in capturing non-linear relationships between variables. Furthermore, the ensemble nature of random forest, which leverages multiple decision trees, offered us an added resistance to overfitting, a critical concern given the complexity of personality traits and their predictors. Initially, we utilized the default parameters from Scikit-learn's package.

XGBoost

For our fourth and final personality prediction model, we employed the XGBoost regression algorithm. The rationale behind this selection was its computational efficiency, largely attributed to its parallel processing capabilities. Just as with the random forest model, we were keen on methods that enhance the generalizability of our predictions and reduce overfitting. XGBoost's inherent regularization features align with this aim, reinforcing our confidence in its application.

We initiated the models using the default parameters of XGBoost.

4.3 Establishing the Success Prediction Models

Since the predicted personalities served as input to the startup success prediction models, we applied the best-performing personality prediction model for each personality trait to every founder's tweets.

Given that many companies had multiple founders, we needed a method to represent the personalities of the entire team. We opted for selecting the mean score for each trait among all the founders. Additionally, we also evaluated other approaches such as the median, maximum, and minimum scores.

After obtaining the predicted personality scores for all founders and determining a method to represent the entire team for each startup, we appended this information to the Final Company dataset. This enriched dataset then served

as input for constructing two sets of models for each modeling technique, one targeting the Series B outcome and the other targeting the Series C outcome.

Logistic Regression

Given our goal for the startup success prediction models was to primarily assess the impact of the predicted personalities, we chose a logistic regression model as our baseline. With this model, we aimed to determine whether a linear relationship existed between the predicted personalities and the success of startups and to gauge the effectiveness of the more complex models. Initially, we used the default parameters for the LogisticRegression module provided by Scikit-learn.

Random Forest

Our second startup success prediction model utilized the random forest approach to achieve greater model complexity. While we initially used the default parameters from Scikit-learn's RandomForest module, as with the personality prediction model, we specifically tailored this model for classification tasks. The benefits of random forests, such as its ability to handle complex interactions and its built-in feature importance metrics, made it an apt choice for our classification objectives.

XGBoost

Our third startup success prediction model employed the XGBoost algorithm, adapted for classification. We began with the default parameters of XGBoost, drawing on its renowned efficiency and scalability to optimize our model's performance.

4.4 Hyperparameter Tuning

Hyperparameter tuning was undertaken for all models, excluding the naive personality prediction model. For the machine learning models, we utilized random searches coupled with 5-fold cross validation. The evaluation metrics we used included mean MAE for personality prediction models and mean F1-score for startup success models.

Evaluation Metrics

To evaluate our models effectively, we employed various metrics tailored to the unique nature of each prediction task.

For our success prediction models, we utilized the Macro F1 score and the Area Under the Curve (AUC). The Macro F1 score, balancing precision, and recall, is especially pertinent for imbalanced datasets as it affords equal weight to the performance of both classes. It is given by:

$$\text{Macro F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where precision denotes the fraction of relevant instances among the retrieved ones, and recall is the fraction of the total relevant instances that were retrieved.

The AUC measures the ability of the model to differentiate between positive and negative classes. A value of 0.5 implies a performance no better than random guessing. As a benchmark, a study by [18] achieved a Macro F1 score of 0.43, emphasizing the relative success of models that exceed this value.

For the personality prediction aspect, we relied on the Mean Absolute Error (MAE). This metric captures the average magnitude of errors between the predicted and actual values, irrespective of their direction. The MAE is described by:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Here, y_i represents the actual value, $\hat{y}_i$ stands for the predicted value, and n is the total number of observations.

Details of Hyperparameter Tuning

For the neural network models, we tuned parameters related to the number of hidden layers and their units, the choice of optimizer, the learning rate, dropout rate, and the type of regularization. The parameters tested are found in table 4.1.

Hyperparameter	**Range Tested**
Number of Hidden Layers	2-6
Units per Layer	64-4096 (halved after every layer)
Dropout Rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
Learning Rate	0.005, 0.01, 0.05, 0.1
Optimizer	Adam, RMSprop
Loss	MSE, Huber

Table 4.1: Range of Parameters Tested for Neural Networks using Keras Tuner

In the logistic regression models, our tuning process encompassed the regularization strength, different penalty types, various solvers, and the ratio for the

Parameter	Values
C	0.001, 0.01, 0.1, 1, 10, 100, 1000
penalty	l1, l2, elasticnet
solver (l1 penalty)	liblinear, saga
solver (l2 penalty)	newton-cg, lbfgs, liblinear, sag, saga
solver (elasticnet penalty)	saga
l1_ratio (for elasticnet)	0.1, 0.2, ..., 0.9

Table 4.2: Parameter Grids for logistic regression used in RandomSearch

elasticnet penalty. All parameters tested for the logistic regression model can be found in table 4.2.

In the case of XGBoost, we adjusted parameters such as the number of boosting rounds, learning rate, maximum tree depth, subsample ratio, the fraction of columns used by each tree, and the minimum loss reduction required to make a split. The full range of parameters tested can be found in table 4.3.

Parameter	Values
n_estimators	10 to 1000 (step 10)
learning_rate	0.001, 0.01, 0.1, 0.2, 0.3
max_depth	2 to 40 (step 1)
subsample	0.5 to 1.0 (step 0.1)
colsample_bytree	0.5 to 1.0 (step 0.1)
gamma	0, 0.1, 0.2, 0.3, 0.4, 0.5

Table 4.3: Consolidated Parameter Ranges for XGBoost used in random search (for both Personality and Success predictions)

Lastly, for the random forest models, we modified the number of trees in the forest, the maximum number of features considered for splitting, the maximum depth of the tree, the minimum number of samples required to split, and the minimum number of samples required at each leaf node. All parameters tested for the two sets of random forest models can be found in table 4.4.

The final parameters that resulted in the best performance for the personality prediction models can be found in tables 4.5 - 4.7, and the for the success prediction models in tables 4.8 - 4.10.

4.5 Performance Evaluation

For a comprehensive evaluation of our models, we used a set of metrics based on different objectives.

Parameter	Values
n_estimators	10 to 1000 (step 10)
max_features	sqrt, -
max_depth	2 to 40 (step 1), None
min_samples_split	2 to 80 (step 2)
min_samples_leaf	1 to 40 (step 1)

Table 4.4: Parameter Grids for random forest used in random search (for both Personality and Success predictions)

In the case of personality prediction models, we utilized the Mean Average Error (MAE) to evaluate the performance of the models while monitoring the performance of the Naive Models. In this scenario, the MAE provided a clear understanding of model performance since the personality scores ranged between 0-100 for each trait. This meant that an MAE of 10 indicated that, on average, we were 10 points away from the actual personality score.

Monte Carlo Sampling

For the assessment of our success prediction models, we employed Monte Carlo sampling to counteract potential biases that originated from model initialization and the inherent randomness in parameter selection. By drawing 1,000 seeds from a uniform probability distribution, we sought to ensure that our model evaluations were robust and that our conclusions were not influenced by any specific seed. This technique enabled us to create a distribution of potential outcomes, thereby giving us greater confidence in the stability and generalizability of our model outcomes.

Feature Importance

Regarding the startup success prediction models, our main approach to evaluate the impact of the predicted personalities was by examining the importance of the set of features over the 1,000 Monte Carlo samples. We defined feature importance in three distinct ways, depending on the specific model utilized. Since our input data was standardized, we used the absolute value of the coefficients from the logistic regression as an indicator of feature importance. For the random forest models, the feature importance represented the average reduction in the Gini impurity introduced by each feature across all trees in the forest. For the XGBoost models, the feature importance was computed as the (averaged) number of times a feature was used to split the data across all trees.

	Open-ness	Conscien-tiousness	Extro-version	Agree-ableness	Neuro-ticism
TF-IDF					
Num Layers	4	5	2	6	4
Units	2112	3648	576	3392	1600
Dropout	0.1	0.3	0.4	0.4	0.2
Learning Rate	0.005	0.05	0.005	0.01	0.05
Optimizer	Adam	Adam	RMSprop	Adam	Adam
Loss	MSE	MSE	MSE	MSE	MSE
BERT					
Num Layers	2	5	2	2	2
Units	256	1024	256	1024	1280
Dropout	0.4	0.3	0.2	0.3	0.3
Learning Rate	0.01	0.1	0.005	0.1	0.1
Optimizer	RMSprop	Adam	Adam	Adam	Adam
Loss	MSE	MSE	MSE	MSE	MSE

Table 4.5: Best Parameters for Feed-Forward neural network Personality Prediction using TF-IDF and BERT

4.6 Software Implementation

The entirety of our project's implementation was grounded in the Python programming language, taking full advantage of its extensive library ecosystem. We heavily relied on several Python libraries for specific tasks.

To extract English tweets, we utilized the `nltk` library. Feature extraction from the text data, specifically using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, was achieved with the `Scikit-learn` library. For deeper language representations, the BERT model from the `Huggingface Transformers` library was employed.

Dimensionality reduction, especially using Truncated Singular Value Decomposition (SVD), was done with `Scikit-learn`. This library was also indispensable for data partitioning and to address class imbalances using the Synthetic Minority Over-sampling Technique (SMOTE).

For modeling, we utilized a variety of algorithms. Logistic regression (LR) and random forest (RF) were implemented via `Scikit-learn`, while XGBoost was handled through the `xgboost` library. neural network architectures were crafted and trained using `TensorFlow` and `Keras`.

Finally, for hyperparameter optimization, random searches for traditional machine learning models were facilitated by `Scikit-learn`. For optimizing the neural

	Openness	Conscientiousness	Extroversion	Agreeableness	Neuroticism	
BERT						
n_estimators	117	74	117	117	94	
max_leaves	1	-	1	1	-	
max_depth	3	2	3	3	2	
learning_rate	0.0742	0.07	0.0742	0.0742	0.1357	
TF-IDF						
n_estimators	50	65	25	50	35	
max_leaves	4	6	4	4	3	
max_depth	13	4	17	13	13	
learning_rate	0.1136	0.0818	0.1909	0.1136	0.2318	

Table 4.6: Best Parameters for XGBoost Personality Prediction using BERT and TF-IDF

networks, we leveraged the `Keras Tuner` library.

	Openness	Conscientiousness	Extroversion	Agreeableness	Neuroticism
BERT					
n_estimators	190	60	10	25	300
min_samples_split	8	14	8	10	4
min_samples_leaf	16	1	20	20	18
max_features	sqrt	-	sqrt	sqrt	-
max_depth	None	10	2	None	10
TF-IDF					
n_estimators	22	110	70	210	150
min_samples_split	7	8	2	26	24
min_samples_leaf	25	25	7	7	5
max_features	sqrt	-	sqrt	sqrt	-
max_depth	9	8	9	2	8

Table 4.7: Best Parameters for random forest Personality Prediction using BERT and TF-IDF

	Series B	**Series C**
solver	saga	saga
penalty	elasticnet	elasticnet
l1_ratio	0.1	0.1
C	0.01	0.01

Table 4.8: Best Hyperparameters for LR Success Prediction Model

	Series B	**Series C**
n_estimators	225	880
min_samples_split	3	24
min_samples_leaf	4	9
max_features	None	150
max_depth	10	22

Table 4.9: Best Hyperparameters for RF Success Prediction Model

	Series B	**Series C**
subsample	0.8	0.5
n_estimators	160	360
max_depth	8	5
learning_rate	0.001	0.1
gamma	0.5	0.2
colsample_bytree	0.5	0.7

Table 4.10: Best Hyperparameters for XGBoost Success Prediction Model

5 Results

The results are organized in two distinct parts: first, we examine the personality predictions and highlight which models perform best. We then turn our attention to the success prediction models, which use the top-performing personality models as input for predicting personality features. For each model, we report the significance of the personality predictions across different iterations of a Monte-Carlo sampling run. We also compare the predicted personality features to other key features to gauge their influence on success predictions. To demonstrate the models' efficacy, we present the macro F1 and AUC scores for both the 'Series B' and 'Series C' targets.

5.1 Personality Predictions

Upon examining Table 5.1, it was evident that, on average, all models outperformed the Naive model (with an MAE of 10.65) except for the TF-IDF+NN, which exhibited a slightly higher MAE of 10.69. The random forest model with BERT-embeddings (BERT+RF) demonstrated the best performance, registering an average MAE of 10.44. This model was closely followed by the XGBoost model using the same embeddings, which recorded an MAE of 10.45.

When assessing individual personality trait predictions, it was found that at least one model surpassed the Naive predictor for every trait:

- For the Openness trait, the neural network with BERT embeddings (BERT+NN) performed the best with an MAE of 7.11.
- For the Conscientiousness trait, the BERT+XGB model performed the best with an MAE of 11.11.
- For the Extroversion trait, the TF-IDF+RF model performed the best with an MAE of 10.41.
- For the Agreeableness trait, the BERT+RF model performed the best with an MAE of 10.53.

	Openness	Conscientiousness	Extroversion	Agreeableness	Neuroticism	Average
Naive	7.87	11.81	10.49	10.54	12.53	10.65
BERT + NN	**7.11**	11.47	10.85	10.94	12.40	10.55
BERT + RF	7.35	11.48	10.95	**10.53**	11.88	**10.44**
BERT + XGB	7.88	**11.11**	10.49	10.54	12.21	10.45
TF-IDF + NN	7.69	11.24	11.21	10.82	12.51	10.69
TF-IDF + RF	7.87	11.49	**10.41**	10.63	12.07	10.49
TF-IDF + XGB	7.83	11.22	10.71	10.96	**11.85**	10.51
Average	**7.66**	11.40	10.73	10.71	12.21	10.54

Table 5.1: Comparison of Mean Absolute Error (MAE) across various model and embedding combinations on the test data, segmented by personality traits. Trait scores can range from 0 to 100. Bold values highlight the best performing model for each trait.

- For the Neuroticism trait, the TF-IDF+XGB model performed the best with an MAE of 11.85.

In a comparison between the Naive predictor and the average MAEs for each trait, the average MAEs for Openness, Conscientiousness, and Neuroticism are lower than that of the Naive model. However, for the traits of Extroversion and Agreeableness, the average MAEs are slightly higher than the Naive predictor's MAE.

5.2 Success Predictions

The results for the Success Predictions are also organized into two sections: Feature Importance and Model Performance. The first aims to display the significance of the predicted personality features, and the second showcases the performance of the models where these features were incorporated.

Feature Importance

The histograms in Figures 5.1-5.2 illustrate the distribution of importance of the five personality trait features for the random forest (RF) and eXtreme Gradient Boosting (XGBoost) models, based on 1,000 Monte-Carlo (MC) samples for the Series B target variable. Both figures indicate that all personality traits hold non-zero significance in both the RF and XGBoost models, with Neuroticism emerging as the most important feature in both cases. In contrast, the results for the logistic regression model showed significance for only two out of the five traits: Openness

and Neuroticism, effectively diminishing the importance of Conscientiousness, Extroversion, and Agreeableness. The histogram for the logistic regression model was not included due to the minimal randomness associated with logistic regression, and the figures did not offer substantial value.

The bar charts in Figures 5.3-5.5 display the average importance of the top 35 features for each respective model for the Series B target, representing roughly the top 2.3% of all features. For the logistic regression (Figure 5.3), both Neuroticism and Openness appear within this upper tier. For the RF and XGBoost models, all personality traits are found within the top 2.3%. Notably, in the random forest model, Neuroticism ranks as the fourth most important feature, whereas in XGBoost, it holds the fifth spot.

Regarding the Series C target, the results were somewhat different. However, for the RF and XGBoost models, all five traits remain significant, as shown in Figures 5.6 and 5.7, with all feature importance being non-zero. The variations are that, for the logistic regression, the Agreeableness trait had non-zero importance, and in general, all models placed a higher emphasis on the Agreeableness trait. This emphasis is evident in the bar charts showcasing the top 35 features for each model for the Series C target in Figures 5.8-5.10. Furthermore, the Neuroticism trait was not as pivotal for the Series C round as it had been for the Series B round, with Openness (or Agreeableness) seeming more critical. Again, for both the RF and XGBoost models, all five predicted traits are within the top 2.3% of most important features.

Model Performance

	Macro F1 (C.I.)	AUC (C.I.)
Logistic regression	0.5852 (0.5935, 0.5867)	0.6440 (0.6412, 0.6467)
random forest	0.6192 (0.6172, 0.6213)	0.6485 (0.6463, 0.6507)
XGBoost	**0.6868** (0.6846, 0.6890)	**0.6947** (0.6922, 0.6973)

Table 5.2: Comparison of model performance for predicting Series B raises in terms of Mean Macro F1 and AUC on the test set. Both metrics are provided with 95% confidence intervals, derived from 1,000 Monte-Carlo sampling iterations using varied seeds.

The performance metrics when predicting Series B raises are displayed in table 5.2. Derived from Monte Carlo sampling, the results show the Macro F1 and AUC values on the test dataset. The XGBoost model stands out, achieving the top scores for both Macro F1 (0.6868) and AUC (0.6947). The RF model follows, securing the second-best scores for both metrics: Macro F1 (0.6192) and AUC

	Macro F1 (C.I.)	AUC (C.I.)
Logistic regression	0.5321 (0.5318, 0.5324)	0.6007 (0.6005, 0.6009)
random forest	0.5849 (0.5844, 0.5855)	0.6307 (0.6300, 0.6315)
XGBoost	**0.6188** (0.6165, 0.6212)	**0.6372** (0.6349, 0.6394)

Table 5.3: Comparison of model performance for predicting Series C raises in terms of Mean Macro F1 and AUC on the test set. Both metrics are provided with 95% confidence intervals, derived from 1,000 Monte-Carlo sampling iterations using varied seeds.

(0.6485). The logistic regression model, although trailing in terms of Macro F1 (0.5852), has an AUC score (0.6440) that is competitive, being relatively close to that of the RF model.

For predictions concerning Series C raises, as shown in Table 5.3, there's a general decline in model performance when compared to Series B predictions. However, XGBoost remains at the forefront, leading in both Macro F1 (0.6188) and AUC (0.6372) metrics. Logistic regression produces a Macro F1 of 0.5321 and an AUC of 0.6007. The RF model is closer to the XGBoost in performance, achieving a Macro F1 of 0.5849 and an AUC of 0.6307. Notably, the AUC scores between RF and XGBoost are quite similar, indicating comparable performances in ranking predictions.

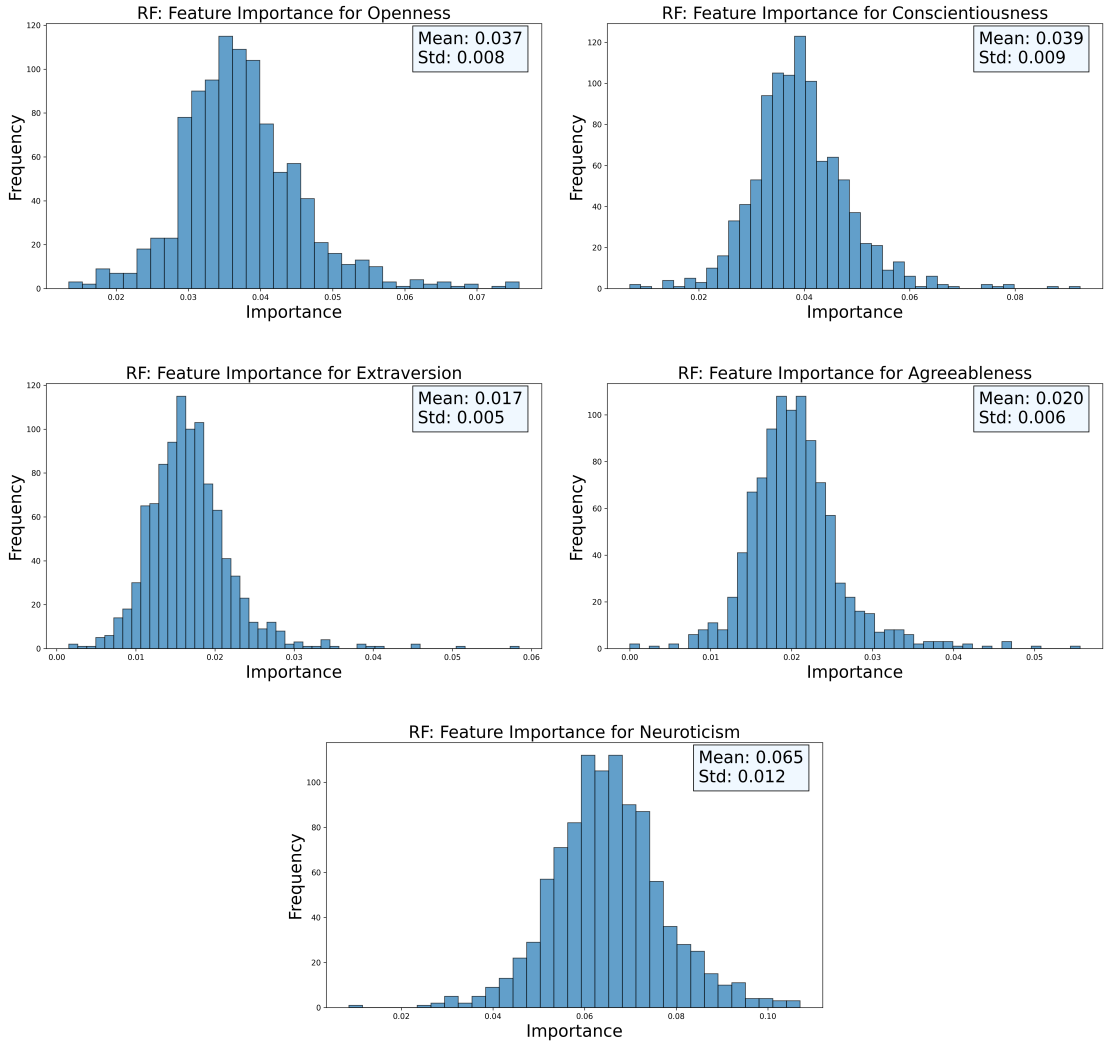

Figure 5.1: Feature importance histograms for random forest Model, derived from 1000 Monte Carlo samples. In the context of random forest, feature importance reflects the average reduction in the Gini impurity brought by each feature across all trees in the forest.

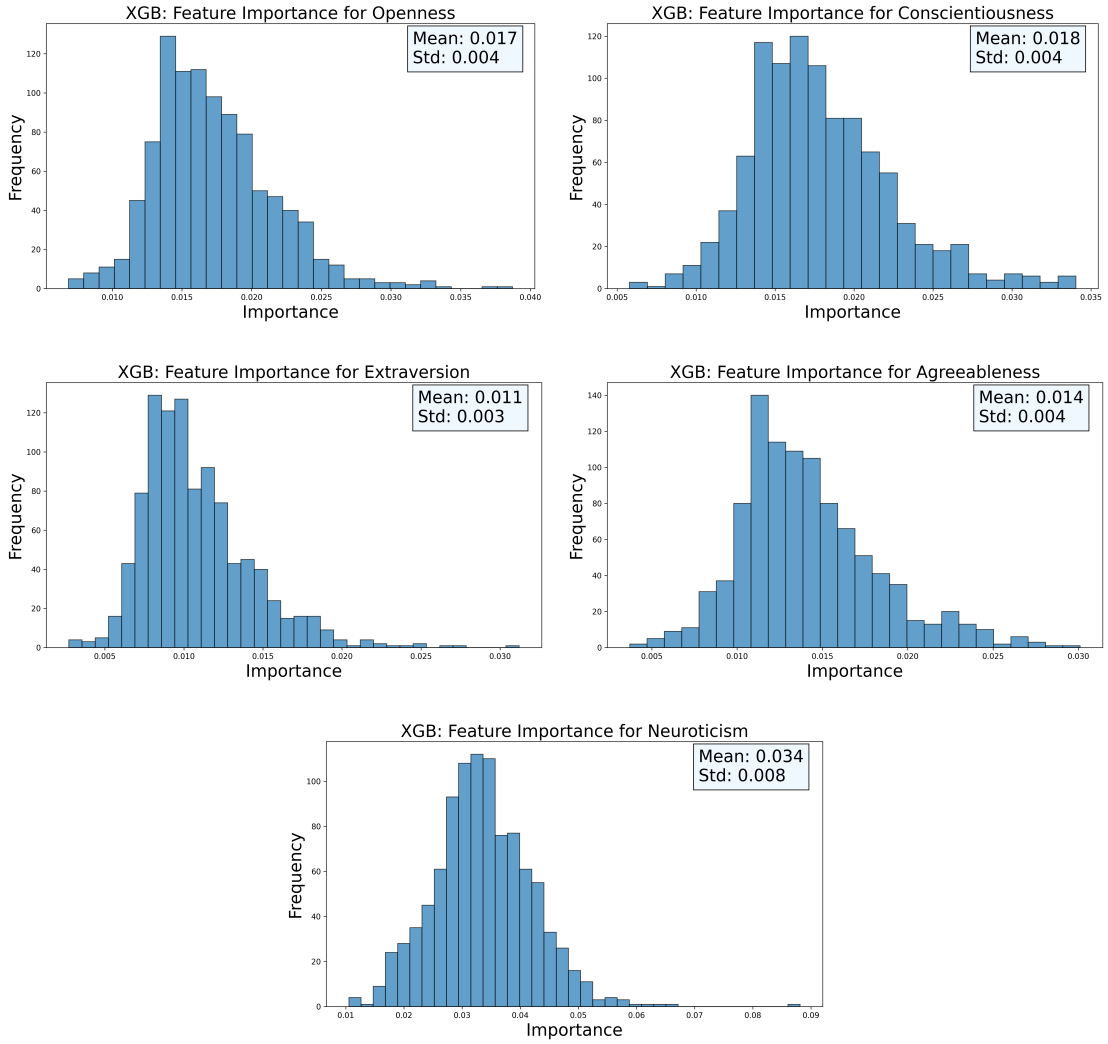

Figure 5.2: Feature importance histograms for XGBoost model, derived from 1000 Monte Carlo samples. In the context of XGBoost, feature importance indicates the (averaged) number of times a feature is used to split the data across all trees.

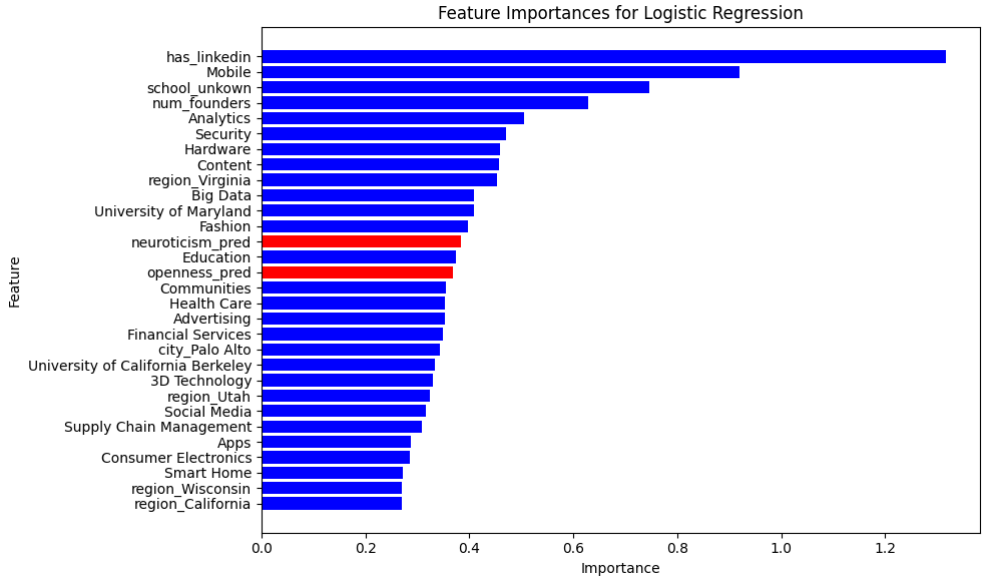

Figure 5.3: Top 35 feature importances for logistic regression, averaged over 1,000 Monte-Carlo samples. Feature importance here represents the absolute value of the model's coefficients. Personality trait features are highlighted in red.

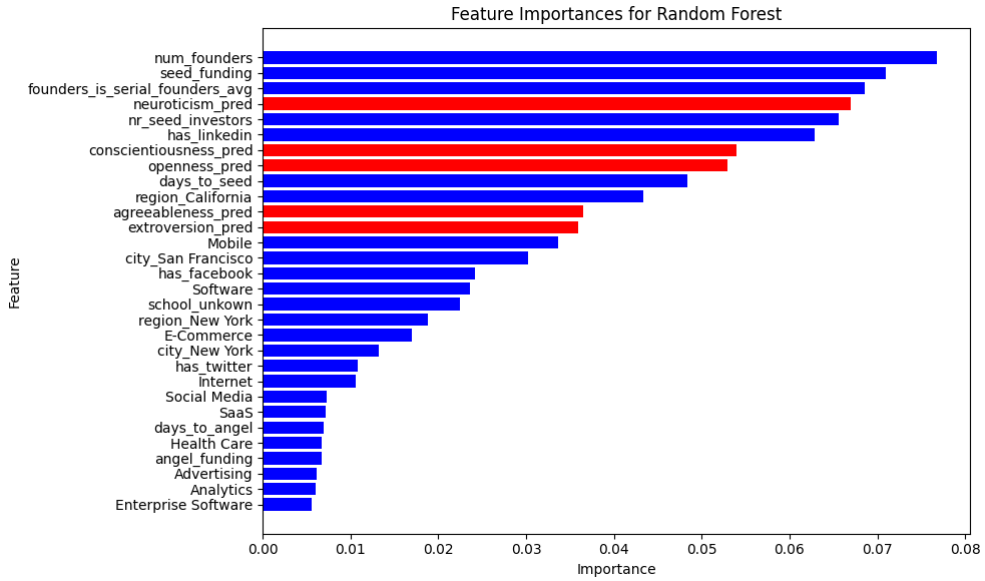

Figure 5.4: Top 35 feature importances for random forest, averaged over 1,000 Monte-Carlo samples. In the context of random forest, feature importance reflects the average reduction in the Gini impurity brought by each feature across all trees in the forest. Personality trait features are highlighted in red.

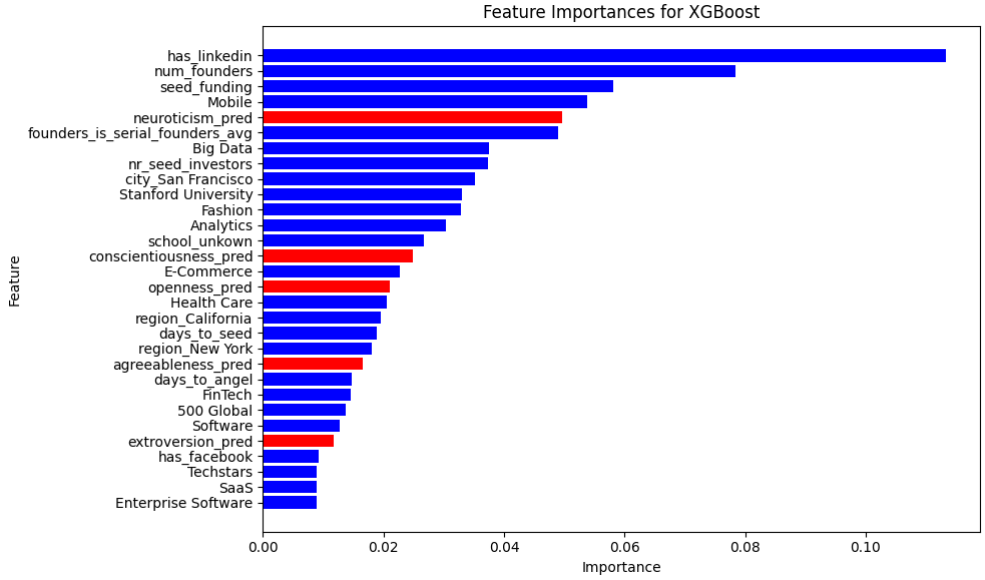

Figure 5.5: Top 35 feature importances for XGBoost, averaged over 1,000 Monte-Carlo samples. In the context of XGBoost, feature importance indicates the (averaged) number of times a feature is used to split the data across all trees. Personality trait features are highlighted in red.

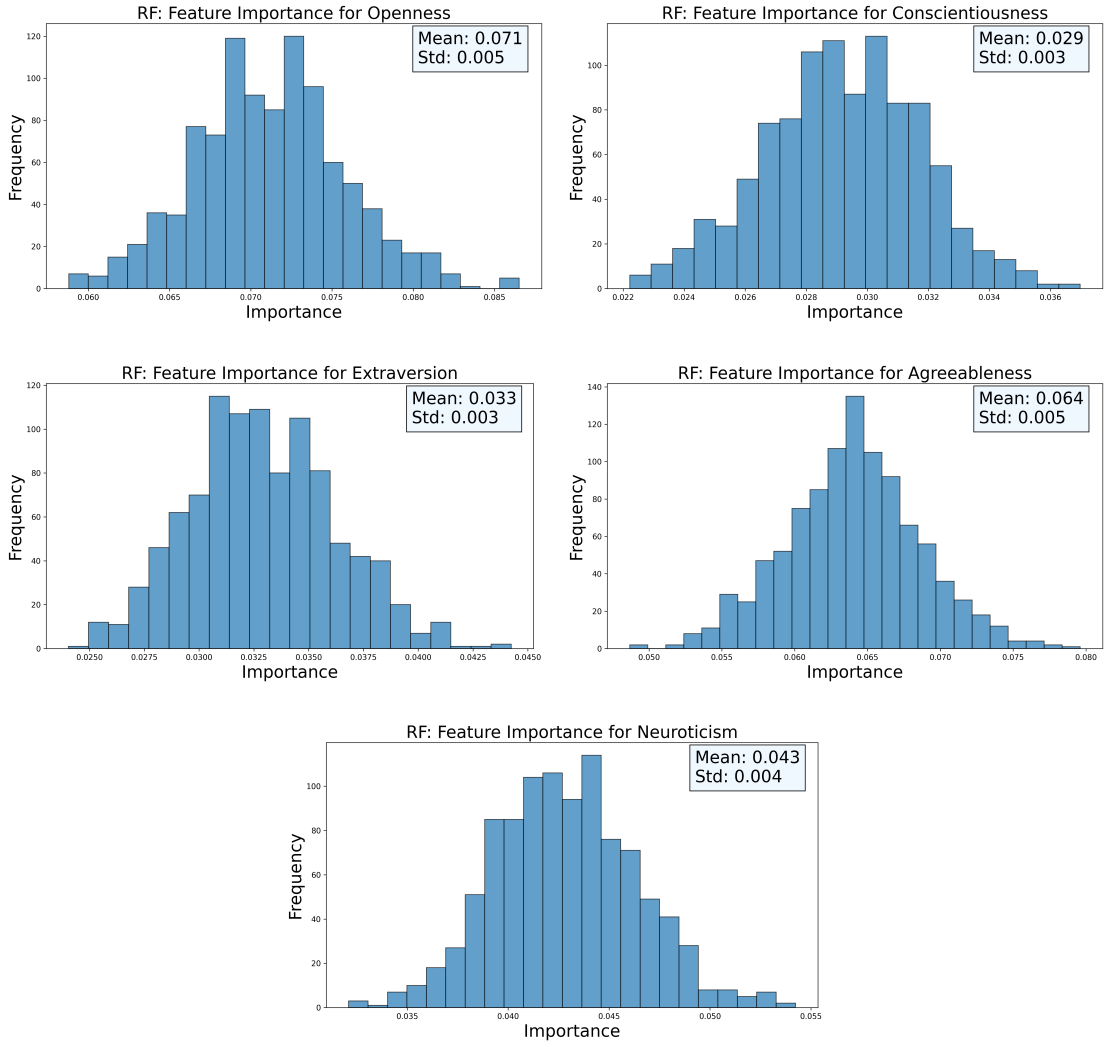

Figure 5.6: Series C: Feature importance histograms for random forest Model, derived from 1000 Monte Carlo samples. In the context of random forest, feature importance reflects the average reduction in the Gini impurity brought by each feature across all trees in the forest.

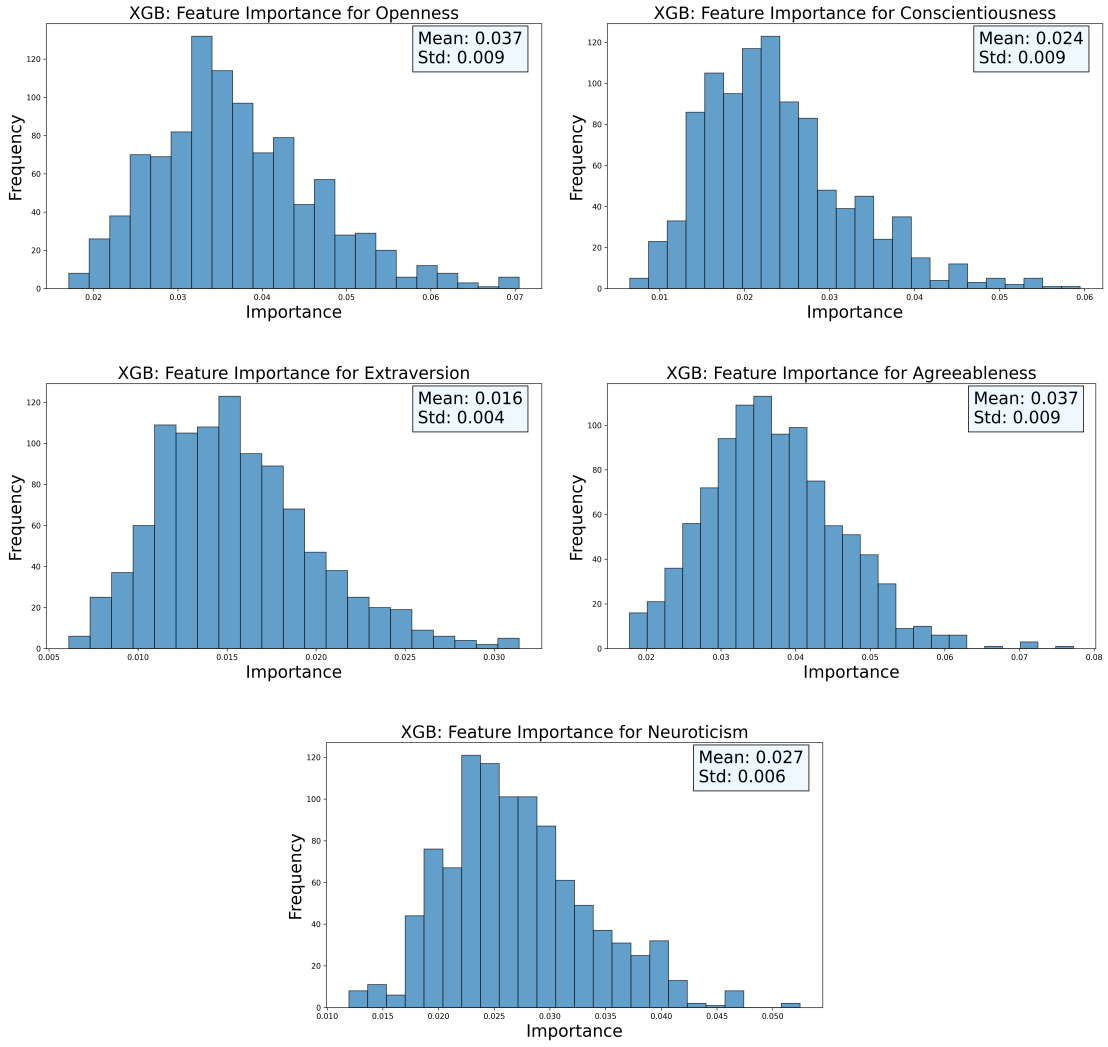

Figure 5.7: Series C: Feature importance histograms for XGBoost Model, derived from 1000 Monte Carlo samples. In the context of XGBoost, feature importance indicates the (averaged) number of times a feature is used to split the data across all trees.

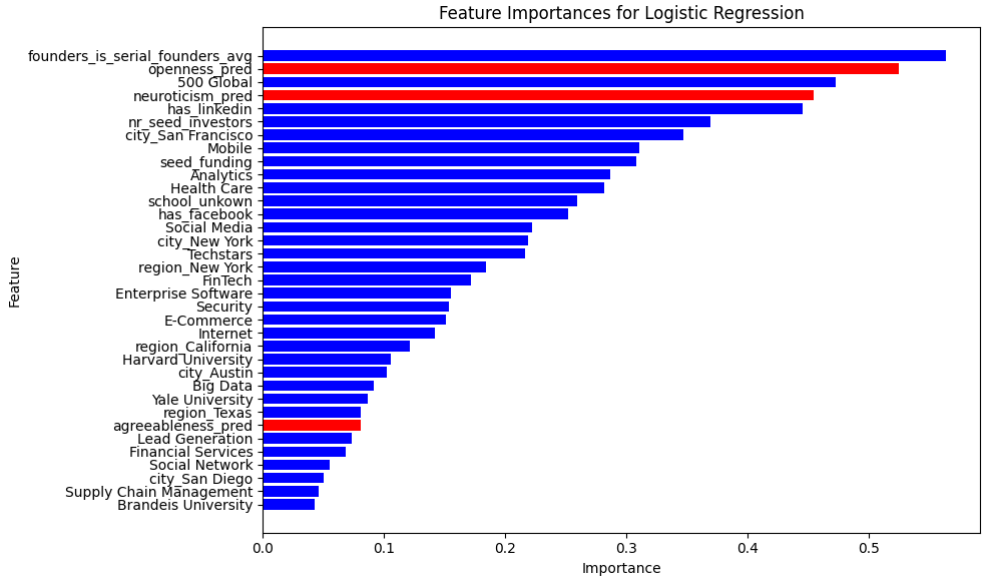

Figure 5.8: Series C: Top 35 feature importances for logistic regression, averaged over 1,000 Monte-Carlo samples. Feature importance here represents the absolute value of the model's coefficients. Personality trait features are highlighted in red.

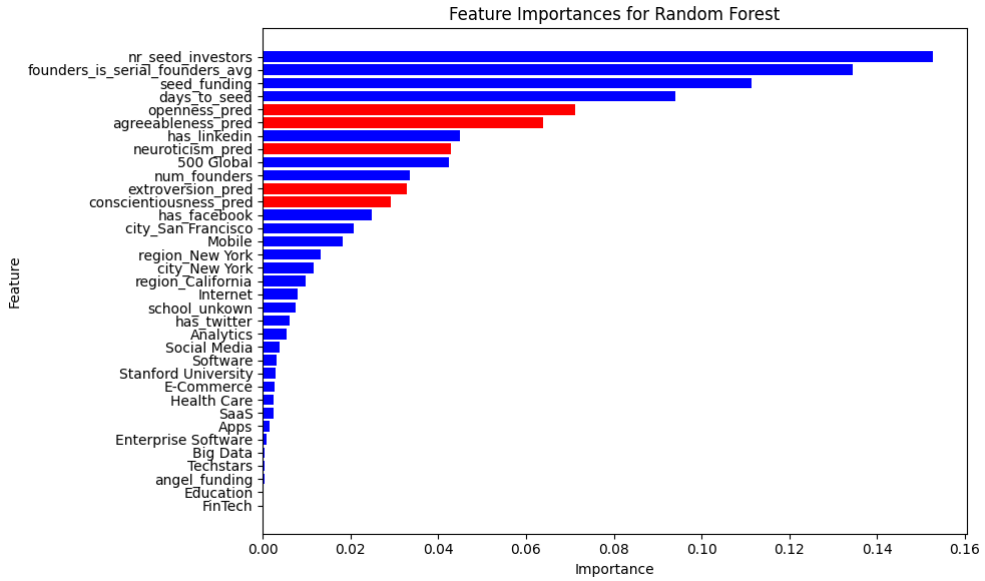

Figure 5.9: Series C: Top 35 feature importances for random forest, averaged over 1,000 Monte-Carlo samples. In the context of random forest, feature importance reflects the average reduction in the Gini impurity brought by each feature across all trees in the forest. Personality trait features are highlighted in red.

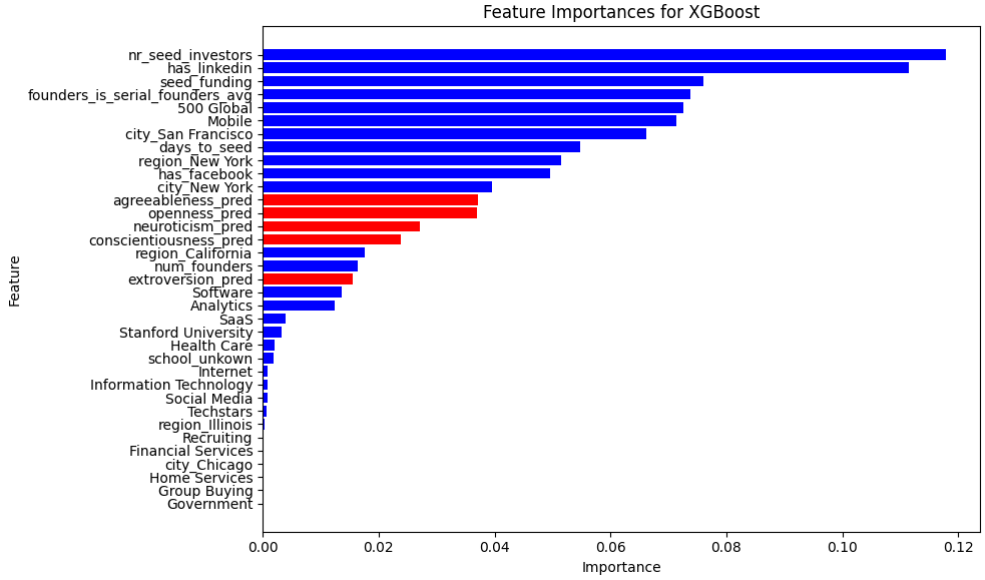

Figure 5.10: Series C: Top 35 feature importances for XGBoost, averaged over 1,000 Monte-Carlo samples. In the context of XGBoost, feature importance indicates the (averaged) number of times a feature is used to split the data across all trees. Personality trait features are highlighted in red.

6 Discussion

In this section, we dissect the results from our modeling efforts aiming to understand the potential of integrating founder personalities into early-stage startup predictions. Through a closer examination of feature importance, model performance, and the predictability of different personality traits, we aim to provide a comprehensive view of our findings and their implications.

6.1 Significance of Founder Personalities

Understanding the significance of founder personalities necessitates a multifaceted approach, considering the specific trait, model used, and target variable in question. A key insight for our results is that the advanced models, namely random forest and XGBoost, heavily rely on all Big 5 personality traits in determining a company's success. This is evident from the feature importance observed in figures 5.1-5.2 and 5.6-5.7, where all traits were among the top 2.2%.

In contrast, the simpler logistic regression models recognize only a subset of these traits as significant, namely, Openness and Neuroticism for both targets, and Agreeableness for the Series C target. This discrepancy might indicate that there is a nonlinear relationship between some traits and startup success. However, as the feature importance for logistic regression is the absolute value of the coefficients, we cannot make any claims about whether the success increases with the scores for these traits.

Of note is the pronounced influence of the Neuroticism trait, which consistently ranks as one of the most influential for all models and targets. Based on our understanding of Neuroticism, this finding is quite intriguing. However, we cannot determine whether a lower or higher level of this trait is advantageous. Similarly, Openness consistently ranks among the top traits across all models and targets. Building on this, a distinct differentiation arises when comparing Series B and Series C fundings. Specifically, for Series C, the Agreeableness trait becomes more prominent, even emerging as the top trait for the XGBoost model.

Similarly, to the pronounced influence of Neuroticism and Openness, Extroversion emerges as a contrasting point of interest. Specifically, it consistently ranks either last or second-to-last in feature importance across all models. Despite this,

in the context of both the random forest and XGBoost models, Extroversion still finds a place among the top 35 most significant features for Series B and Series C targets. Building on this observation, it is worth noting that Extroversion does not play as significant a role as one might anticipate, particularly given that social skills are often deemed essential for startup success.

6.2 Performance of Personality Prediction Models

Delving into the performance of our personality models, discerning the effectiveness of TF-IDF and BERT embeddings is not straightforward. A direct observation indicates that three traits show superior performance when modeled using BERT embeddings. This leads to the preliminary conclusion that BERT, known for its advanced contextual understanding, is generally better suited for predicting personality traits.

However, a closer look reveals that Neuroticism, which stands out as the most crucial trait in our earlier discussions, was more accurately predicted using TF-IDF. This distinction suggests a nuanced understanding is necessary; some traits might be better predicted by the word-focused approach of TF-IDF, rather than BERT's ability to understand context.

As detailed in table 5.1, no singular model architecture consistently outperforms the rest across all traits. Interestingly, when averaging performance metrics over all traits, the random forest model exhibited the lowest MAE for both BERT and TF-IDF embeddings. The absence of a universally superior model and embedding combination can be attributed to the intricate nature of embedded text data and the diverse characteristics of the personality traits themselves. Such variations emphasize the importance of a tailored approach, considering the specificities of individual personality traits and their corresponding embeddings.

A vital aspect of this modeling was including a Naive Model to assess the performance and reliability of the more advanced models. Surprisingly, predicting the traits more accurately than the Naive proved harder than initially thought. Eventually, we found combinations of embeddings and models that outperformed the Naive Model for all traits, but only after numerous fine-tuning and hyperparameter optimization iterations. The difficulty in efficiently constructing models for predicting personalities might indicate potential biases in the data collected, such as specific demographics being overrepresented in tweets, individuals attempting to emulate others online, or tweets being auto generated through other apps. With the emergence of foundation models for text generation, it will be interesting to see how this approach to predicting personalities fares.

6.3 Startup Success Prediction

The best-performing model for predicting the outcomes of both Series B and Series C success was XGBoost. For Series B, it achieved a Macro F1 Score of 0.6868 and an AUC score of 0.6947. For Series C, it achieved a Macro F1 Score of 0.6188 and an AUC score of 0.6372. While these models are not infallible, they effectively learn from the data and distinguish successful ventures from unsuccessful ones.

The gradient boosting mechanism of XGBoost, which corrects errors in parallel, might have an edge over bagging due to its built-in regularization that mitigates overfitting. Moreover, XGBoost's ability to construct deeper trees may enable it to capture intricate data relationships, thereby enhancing its performance. A prevalent assumption for these models is that $n >> p$, but this was not our scenario. Notably, both XGBoost and the random forest have inherent dimensionality reduction properties, making them adept at sidelining noise without resorting to other methods for feature reduction.

Even though we did not exclude personality traits in our experiments, most models, regardless of the target, highlighted their importance. This indicates the positive influence of these traits on model performance. Given the robustness of models like the random forest and XGBoost against overfitting and their generalization capabilities, it is plausible they could perform well even in the absence of these personality traits.

6.4 Challenges and Limitations

A primary constraint of our study was the relatively small dataset we used to build our success models. This limitation arose mainly because we focused on founders with active Twitter accounts who tweeted before securing their seed rounds. While our results are promising, and the methods chosen aimed to reduce biases and prevent overfitting, further research on a larger dataset is needed to verify these findings. Additionally, the brief timeframe in which we considered company founding dates could introduce bias to certain features. For instance, VC investments often target future trends, and per definition trends change over time. Beyond personality features, we relied exclusively on Crunchbase data for feature creation. To enhance accuracy, future research could incorporate diverse data sources, including details on founder experiences, academic credentials, financial records, mentions of startup features in the news, and more.

7 Conclusion

The primary objective of this thesis was to ascertain the significance and potential value of integrating predicted founder personalities into ML models with the goal of enhancing the accuracy of early-stage startup success predictions. Our research indicates that the integration of the predicted Big 5 personality traits into ML models has a notable impact on the accuracy of startup success predictions. Notably, models such as the random forest and XGBoost demonstrated a significant reliance on these traits, underlining the importance of considering founder personalities in predictive modeling.

7.1 Implications

Our research findings offer valuable insights into the startup ecosystem.

Investors and VC firms can benefit from a broader evaluation perspective. Beyond the hard facts like job titles and experience, considering the personalities of founding teams can be pivotal. This thesis lays out a clear method for this: from how to gather the right data to utilizing these personality insights for better startup success predictions.

Founders, on the other hand, have a tool to self-reflect. By understanding where their team's personality strengths and potential gaps lie, they can strategically recruit individuals with complementary traits, paving the way for a more cohesive and balanced startup team.

However, it is essential to recognize the potential limitations. The training data for personality predictions could be biased, and hence may not capture the founders' true personality nuances accurately. Additionally, while this research has focused on the personalities of the founding team, considering the personalities of other key company members could be essential for a more comprehensive understanding.

7.2 Recommendations for Future Research

Our research, while offering valuable insights, has inherent limitations. A crucial step forward would be to expand the dataset, encompassing a wider timeframe

and ensuring a varied geographic representation of startups and founders. Alongside this expansion, introducing data from sources beyond Crunchbase and Twitter could potentially enhance all models under consideration. This broader scope can further validate or refine our initial findings.

In expanding the dataset, especially to non-English regions, the choice of embeddings becomes paramount. Given the multilingual nature of the data, mBERT would likely be a more suitable choice for predicting personality.

Additionally, the current research could be aided by testing a more diverse set of embeddings and models, such as Glove and RoberTa embeddings coupled with more non-tree-based models such as support vector machines. For the case of startup success predictions, a neural network would also be interesting to evaluate.

Furthermore, while this study focused on the Big 5 personality framework, other personality assessment systems, like the Myers-Briggs Type Indicator or the DISC assessment, could be integrated to ascertain which provides the most predictive power. A more granular approach to evaluating team personalities could also be adopted – instead of combining scores to produce a singular team metric, future research could explore the influence of individual roles, such as the CEO, CFO, etc., on startup success.

7.3 Final Thoughts

The intersection of machine learning with entrepreneurship has presented an intriguing frontier for exploration. By investigating the influence of founder personalities on startup success, we have ventured into a domain where human characteristics intersect with algorithmic prediction. Our findings not only showcase the potential of combining human-centric data with machine learning models but also stress the need for balanced consideration when using such tools for predictive purposes. The emphasis of this research on the human element, as represented by personality traits, serves as a reminder that while technology and data play crucial roles, startups' success is equally about the people behind them. As we continue to harness the power of machine learning in entrepreneurship, it is vital to maintain a nuanced approach, recognizing the technology's potential and its boundaries. The evolution of the startup landscape will undoubtedly be influenced by research endeavors like this, emphasizing the complementary roles of data-driven insights and the innate human elements that underlie any business venture.

Bibliography

[1] M. Regona, T. Yigitcanlar, B. Xia, and R. Y. M. Li, "Opportunities and adoption challenges of ai in the construction industry: a prisma review," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 8, no. 1, p. 45, 2022.

[2] R. Baldridge, "What Is A Startup? The Ultimate Guide — forbes.com." https://www.forbes.com/advisor/business/what-is-a-startup/. [Accessed 06-Jul-2023].

[3] "Venture Capital: What Is VC and How Does It Work? — investopedia.com." https://www.investopedia.com/terms/v/venturecapital.asp#:~:text=Venture%20capital%20provides%20funding%20to,gain%20equity%20in%20promising%20companies. [Accessed 06-Jul-2023].

[4] V. H. Fried and R. D. Hisrich, "Toward a model of venture capital investment decision making," *Financial management*, pp. 28–37, 1994.

[5] P. A. Gompers, W. Gornall, S. N. Kaplan, and I. A. Strebulaev, "How do venture capitalists make decisions?," *Journal of Financial Economics*, vol. 135, no. 1, pp. 169–190, 2020.

[6] J. Santisteban and D. Mauricio, "Systematic literature review of critical success factors of information technology startups," *Academy of Entrepreneurship Journal*, vol. 23, no. 2, pp. 1–23, 2017.

[7] P. T. Costa and R. R. McCrae, "A five-factor theory of personality," *The five-factor model of personality: Theoretical perspectives*, vol. 2, pp. 51–87, 1999.

[8] E. C. Tupes and R. E. Christal, "Recurrent personality factors based on trait ratings," *Journal of personality*, vol. 60, no. 2, pp. 225–251, 1992.

[9] P. T. Costa Jr and R. R. McCrae, "From catalog to classification: Murray's needs and the five-factor model.," *Journal of personality and social psychology*, vol. 55, no. 2, p. 258, 1988.

[10] G. Saucier, "Effects of variable selection on the factor structure of person descriptors.," *Journal of personality and social psychology*, vol. 73, no. 6, p. 1296, 1997.

[11] R. Helson, V. S. Kwan, O. P. John, and C. Jones, "The growing evidence for personality change in adulthood: Findings from research with personality inventories," *Journal of research in personality*, vol. 36, no. 4, pp. 287–306, 2002.

[12] A. Krishna, A. Agrawal, and A. Choudhary, "Predicting the outcome of startups: less failure, more success," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 798–805, IEEE, 2016.

[13] S. Tomy and E. Pardede, "From uncertainties to successful start ups: A data analytic approach to predict success in technological entrepreneurship," *Sustainability*, vol. 10, no. 3, p. 602, 2018.

[14] F. R. d. S. R. Bento, *Predicting start-up success with machine learning*. PhD thesis, 2018.

[15] B. Sharchilev, M. Roizner, A. Rumyantsev, D. Ozornin, P. Serdyukov, and M. de Rijke, "Web-based startup success prediction," in *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 2283–2291, 2018.

[16] D. Dellermann, N. Lipusch, P. Ebel, K. M. Popp, and J. M. Leimeister, "Finding the unicorn: Predicting early stage startup success through a hybrid intelligence method," *arXiv preprint arXiv:2105.03360*, 2021.

[17] C. Ünal, "Searching for a unicorn: A machine learning approach towards startup success prediction," Master's thesis, Humboldt-Universität zu Berlin, 2019.

[18] K. Żbikowski and P. Antosiuk, "A machine learning, bias-free approach for predicting business success using crunchbase data," *Information Processing & Management*, vol. 58, no. 4, p. 102555, 2021.

[19] J. Golbeck, C. Robles, M. Edmondson, and K. Turner, "Predicting personality from twitter," in *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pp. 149–156, IEEE, 2011.

[20] H. Christian, D. Suhartono, A. Chowanda, and K. Z. Zamli, "Text based personality prediction from multiple social media data sources using pre-trained

language model and model averaging," *Journal of Big Data*, vol. 8, no. 68, 2021.

[21] P. H. Winston, *Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1993.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[25] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[26] C. D. Manning and H. Schütze, *Natural Language Processing*. Cambridge University Press, 2019.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[32] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.

[35] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.

[36] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[37] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[38] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[39] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[40] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," pp. 785–794, 2016.

[41] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A practical approach to microarray data analysis*, pp. 91–109, Springer, 2003.

[42] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[43] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[44] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328, Ieee, 2008.