CHAPTER 1

Design

In this chapter, we describe the methodology used to design our Venture Capital (VC) investment screening system. Figure 1.1 depicts the architecture of our system, structured into five components: data collection, dataset preparation, pipeline creation, pipeline selection, and prediction. We evaluate the performance of this system in the next chapter.

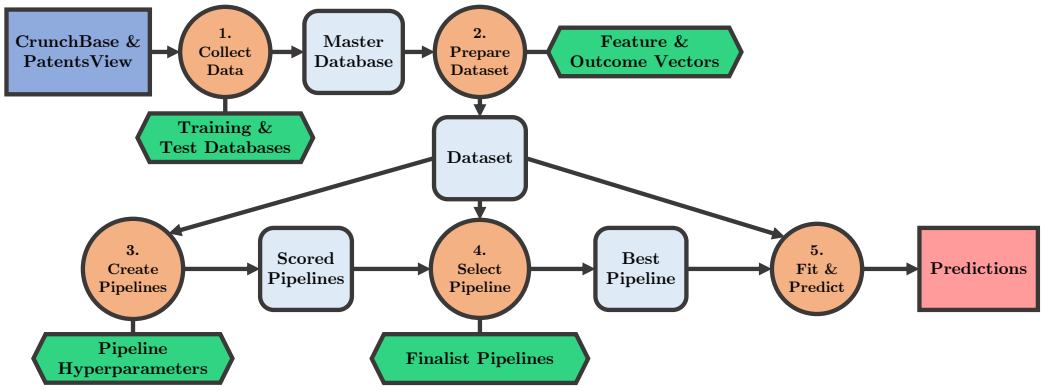


Figure 1.1: An overview of the system architecture proposed by this project. Legend: dark blue square = input, orange circle = system component, light blue rounded square = intermediate, red square = output, green hexagon: iterative process / search space.

1. Data Collection: We developed a conceptual framework to guide our feature and data source selection. Based on this framework, we decided to collect data from CrunchBase and PatentsView. Our system collects data from these sources and imports it into a relational database.

2. Dataset Preparation: The system extracts database slices and converts them into datasets suitable for supervised learning. The system cleans the databases to ensure only relevant companies are included. We report the descriptive statistics of an indicative dataset.

3. Pipeline Creation: We identified issues of sparsity, long-tailed distributions, imbalanced feature ranges, and non-orthogonality in our datasets. We developed a classification pipeline system to address these dataset issues. Our system performs a grid search across the hyperparameters of the pipeline to generate scored, candidate pipelines.

4. Pipeline Selection: Next, our system selects the best candidate pipelines and evaluates their robustness over time. The outcome of this process is a single pipeline optimised to suit the dataset and prediction task. To evaluate robustness over time, we recreated historical datasets using a technique that filters records by their timestamps.

5. Model Fit and Prediction: Finally, our system applies the optimised pipeline to a feature vector from a held-out test database, which generates a model and a set of predictions. We evaluate the models and predictions produced by our system in the next chapter.

1.1 Data Collection

In the previous chapter, we reviewed features and data sources used in entrepreneurship and Venture Capital (VC) investment research. In this section, we first discuss how we developed a conceptual framework to guide our feature and data source selection and then describe the process of collecting and storing data from CrunchBase and PatentsView.

1.1.1 Conceptual Framework

While previous studies into startup performance have explored a range of features, few individual studies have evaluated a comprehensive and diverse feature set. We developed a conceptual framework to ensure we included a comprehensive set of features in our VC investment screening system.

Our conceptual framework built on previous work by Ahlers (2015) to model investment decisions on equity crowd-funding platforms [1]. We sought to generalise Ahlers' framework beyond equity crowd-funding. While the first factor of their framework (venture quality) applies to startups generally, their second factor (investment confidence) was defined with respect to features specific to equity crowd-funding. We developed this second factor further, describing investment confidence as a product of third party validation, historical performance and contextual cues.

A high-level overview of our proposed conceptual framework is depicted in Figure 1.2. In Appendix **??**, we describe the features that underpin each factor and outline the theoretical and empirical evidence that justify their inclusion in our conceptual framework.

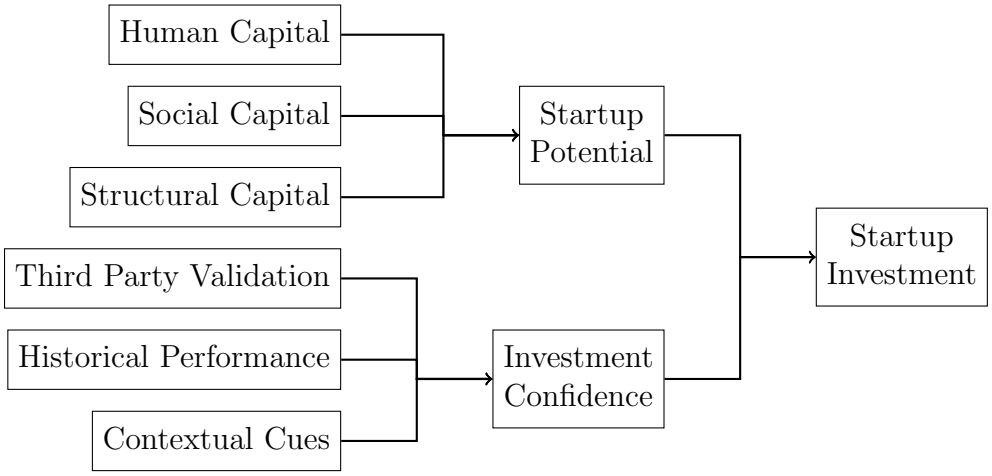


Figure 1.2: Proposed conceptual framework for VC investment. We adapted the framework proposed by Ahlers et al. [1], originally based on work by Baum and Silverman [2]. For an extended version of this framework, please refer to Figure **??**.

1.1.2 Data Sources

We sought data sources that could provide features that support the factors in our conceptual framework. We decided to collect data from CrunchBase with supplementation from PatentsView, a patent records database. These sources cover the majority of factors in our conceptual framework. We discuss the process of collecting data from these sources, as depicted in Figure 1.3.

1.1.2.1 CrunchBase

We were granted an Academic License to collect data from CrunchBase. CrunchBase provides database access in a few formats that offer trade-offs in terms of accessibility and comprehensiveness: REST API, CSV Dumps, MS Excel Summary. We chose to use the CSV Dumps because they provided a good trade-off of ease of use and comprehensiveness of access. CrunchBase provides a CSV file for each of CrunchBase's key endpoints (e.g. organizations, people, funding

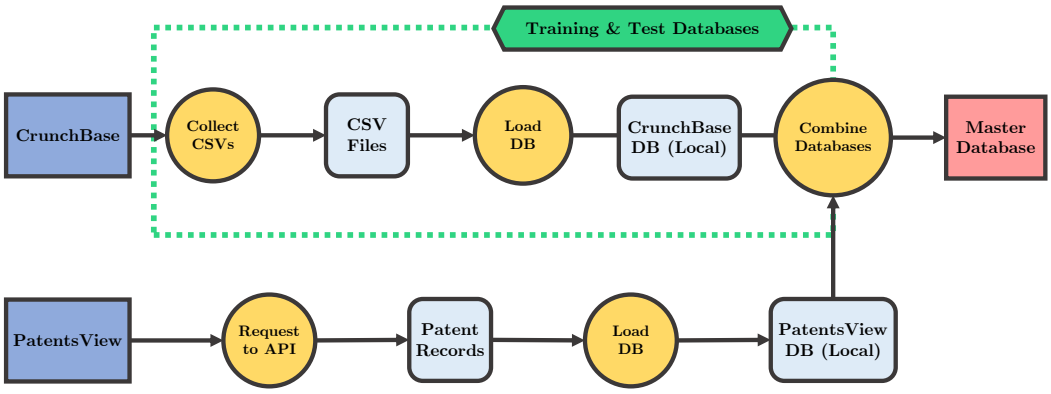


Figure 1.3: Data collection overview. CrunchBase data is collected multiple times to produce separate training and testing databases. Legend: dark blue square = input, yellow circle = process, light blue rounded square = intermediate, red square = output, green hexagon: iterative process / search space.

rounds) which can be loaded easily into relational databases (see Appendix **??** for the database schema). We downloaded CSV Dumps from CrunchBase on 09 September 2016 and 04 April 2017 which became our training and testing databases, respectively.

1.1.2.2 PatentsView

We used PatentsView to obtain the patent filing records of each company in our CrunchBase dataset, focusing on information relating to dates, citations, and patent types. Our system matches companies across CrunchBase and PatentsView by standardising the company names (removing common suffixes, punctuation etc.) and using normalised Levenshtein distances to determine similarity. Although approximate matching introduces error, the volume of companies in the database is too high to be matched manually and there are no other identifying records. We stored the PatentsView data in a relation which we merged into our CrunchBase data to form our master databases.

1.2 Dataset Preparation

Our system performs multiple steps to prepare datasets from our training and test databases for use in machine learning, as depicted in Figure 1.4. In this section, we describe the process of preparing our datasets, which involves two

components: generating historical databases from our relational database, and converting these relational database slices into clean datasets ready for machine learning. Finally, we present the descriptive statistics of our dataset.

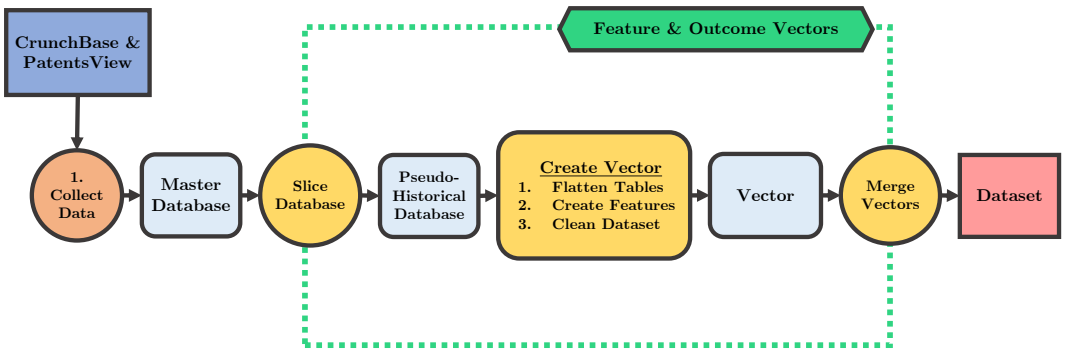


Figure 1.4: Dataset preparation overview. Feature and outcome vectors are created from the master relational database. Legend: dark blue square = input, yellow circle = process, light blue rounded square = intermediate, red square = output, green hexagon: iterative process / search space.

1.2.1 Database Slicing

We developed a procedure for generating historical databases from our CrunchBase and PatentsView data. CrunchBase provides created and last-updated timestamps for each record in their CSV-formatted dumps (and also in the JSON-formatted responses from their API). We used this to reverse-engineer previous database states by filtering the master database by records created prior to a given 'slice' date.

We evaluated our slicing technique by comparing a CrunchBase database collected in December 2013 with a slice engineered from our training database (collected in September 2016), as shown in Figure 1.5. There are only minor differences relative to the size of these datasets.

Figure 1.6 presents company counts by startup development stage from different dataset slices. Dataset counts for all stages have steadily increased over time. Prior to 2012 the datasets become too small to use to make meaningful predictions.

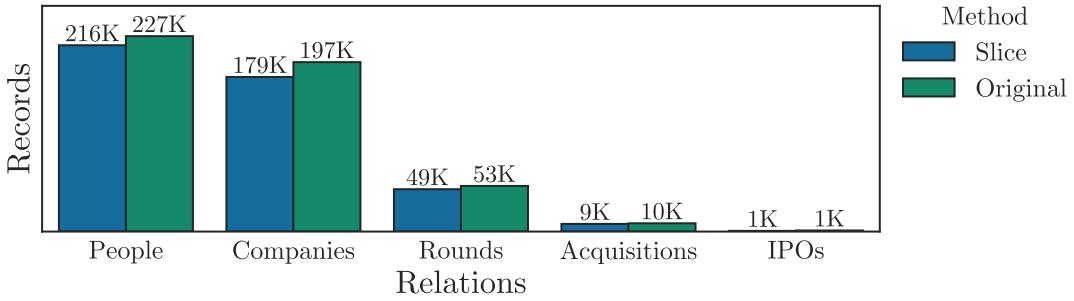


Figure 1.5: Database slice compared with original database. Original database collected in December 2013. Database slice generated from the training database collected in September 2016 and sliced to only include records created prior to December 2013.

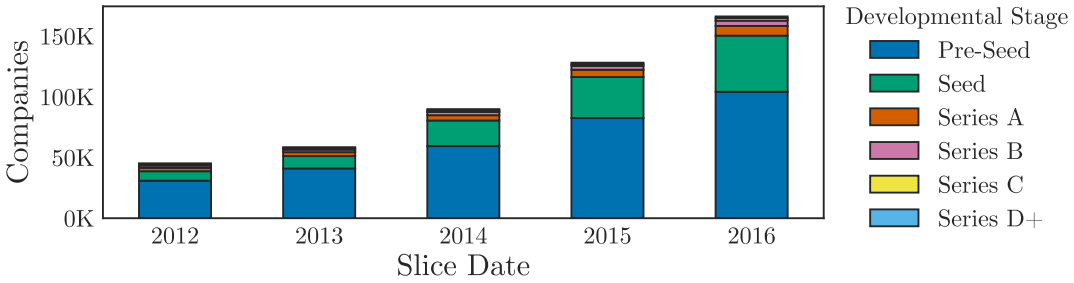


Figure 1.6: Dataset slice counts over time. Each slice was generated on April 04 of each respective year. Proportions by developmental stage stay relatively constant over this timeframe.

1.2.2 Vector Creation

To prepare our datasets for machine learning, our system performs aggregation, feature creation and preliminary screening. In the following section, we evaluate the effect of these processes on an indicative database sliced from the training database as of 09 September 2016 (N = 425,934).

First, our system flattens the relational database slices into a single file using Structured Query Language (SQL) aggregation queries. The system aggregates each relation in turn, grouping by Company ID and combining each aggregated table using left outer joins. Next the system convert tuples (e.g. Round Type and Round Date) and lists (e.g. Round Types) into dummy variables.

Our system performs preliminary screening to remove traditional, non-startup businesses from the dataset (e.g. consulting firms, companies that will not take

Venture Capital (VC) funding etc.). To do this, we explored two factors for each company: developmental stage and age. By developmental stage, we primarily refer to external funding milestones. These stages are associated with shifts in a startup company's functions and objectives. Our dataset as grouped by startup developmental stage is depicted in Figure 1.7.

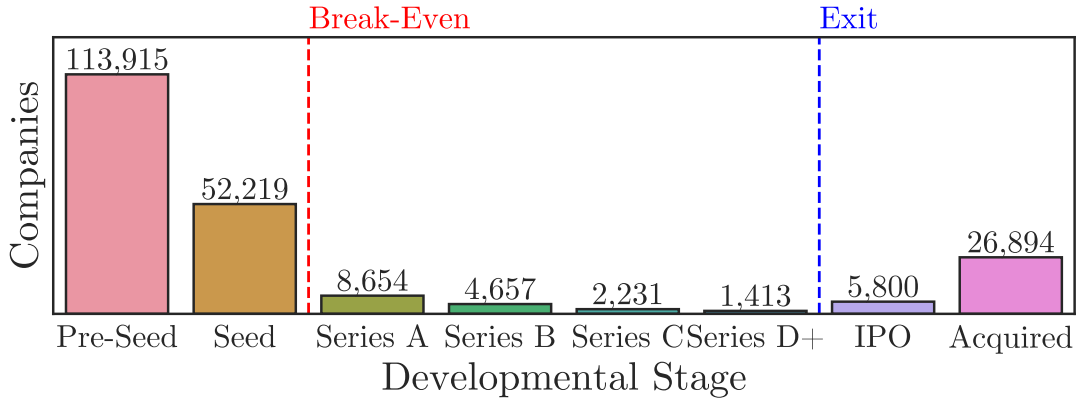


Figure 1.7: Companies grouped by stages of the startup development life-cycle. Companies at Pre-Seed and Seed stages are typically unprofitable and seek external funding to sustain their operations. Companies at Series A - Series D+ are typically either profitable or at least revenue-generating, and seek external funding to expand their operations.

After attempting to place the companies into development stages we are left with a large group of companies that have not raised funding. These companies can be split into two groups – those that intend to raise funding and those that do not. We applied a cut-off equal to the 90th percentile of the age of companies in the Seed category, and excluded the older group from further analyses (N = 227,162, 53.3%). As we are only interested in companies that could seek investment, we also excluded Closed, Acquired and IPO groups from further analyses (N = 35,973, 8.4%).

Figure 1.8 depicts the ages of companies in the dataset grouped by developmental stage. There is a positive relationship between age and developmental stage. Most pre-Series A companies are under five years old, and the majority of Series D+ funded companies are under 10 years old and the 75th percentile is at 15 years old. On this basis, we excluded companies that are over the 75th percentile of the age of companies in the Series D+ category (N =9,756, 2.2%). Our preliminary screening reduced the dataset from 425,934 companies to 153,043 companies, a reduction of 64.1%.

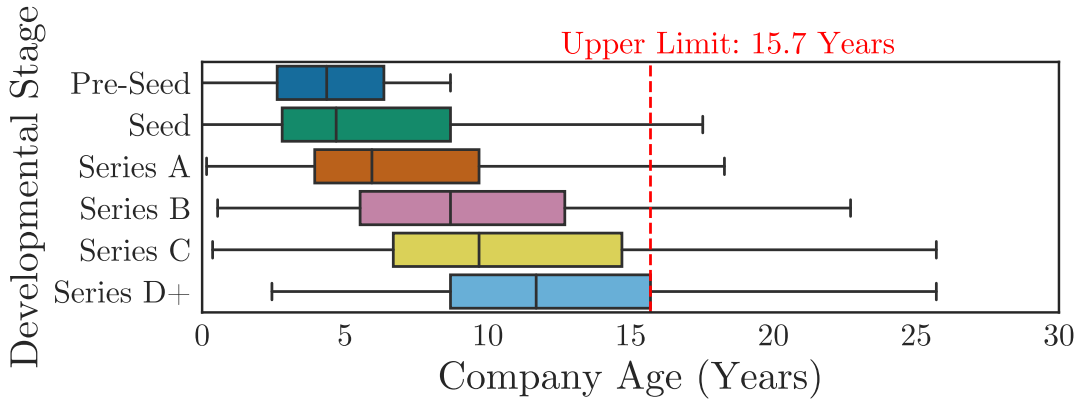


Figure 1.8: Company ages in years grouped by developmental stage. The dashed red line represents the 75th percentile of the age of companies in the Series D+ category (15.7 years).

1.2.3 Descriptive Statistics

Table 1.1 presents the descriptive statistics for the cleaned dataset from the previous section. The dataset is skewed towards Pre-Seed companies (i.e. companies that were recently founded and have not raised funding yet, 68.9%). These companies have few available features in comparison to later developmental stages. We investigate the impact of this on our predictions in Chapter **??**. The interquartile ranges imply significant variability in all measures. We do not believe that this implies that the data has not been cleaned effectively, but rather, that it reflects that startup companies vary in their traits.

Stage	Obs	Age (Years)		Funding Raised (USD, M)		Funding Rounds (N)		Patent Filings (N)		Available Features (N)	
	N	50th	75th	50th	75th	50th	75th	75th	90th	50th	75th
Pre-Seed	113,915	4.36	6.36	0.00	0.00	0	0	0	0	25	133
Seed	38,942	4.66	6.69	0.25	1.30	1	2	0	1	178	231
Series A	6,615	5.69	8.70	4.40	9.41	2	3	0	2	239	302
Series B	3,342	7.61	10.70	14.89	28.20	3	4	0	4	255	314
Series C	1,610	8.70	11.70	35.29	62.00	3	5	1	9	305	321
Series D+	998	9.70	12.70	74.39	130.8	5	7	4	19	319	330
Included	165,422	4.69	6.69	0.00	4.00	1	2	0	1	90	160

Table 1.1: Descriptive statistics grouped by developmental stage.

CrunchBase's industry classification is simplistic compared to other databases (e.g. USSIC, VentureSource) which take a structured, hierarchical approach. For

example, “Software”, “Internet Services” could describe the majority of companies included in the database and account for 16.4% and 13.4% of all companies in the dataset respectively (see Figure 1.9). Despite these vague labels, it is clear the dataset skews towards high technology startups, as opposed to biomedical, agricultural, or other technologies (which do not make the Top 10).

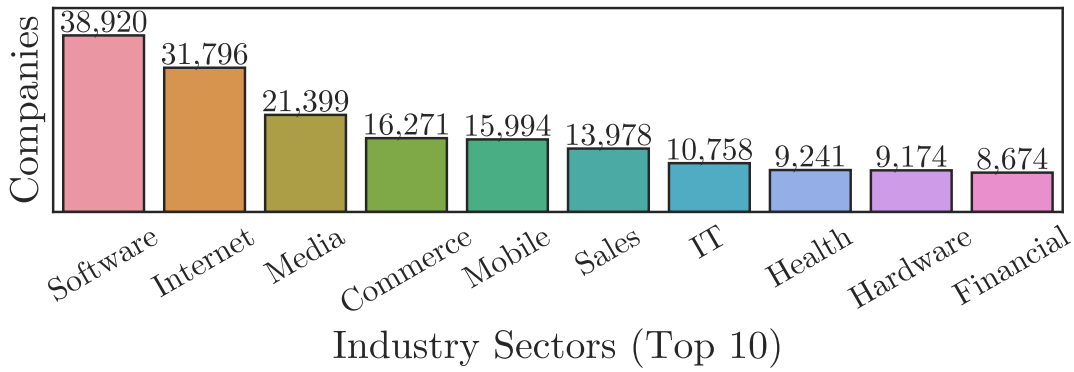


Figure 1.9: Companies grouped by industry sector. Industry labels are not mutually-exclusive. The 10 most common sectors are displayed.

1.3 Pipeline Creation

In the following section, we perform exploratory data analysis on our dataset and decide that a classification pipeline system could help us to address issues identified. The classification pipeline system is depicted in Figure 1.10.

1.3.1 Exploratory Analysis

We performed exploratory data analysis on our dataset to assess what techniques would be suitable and the need for any further pre-proceesing. We identified dataset issues that include sparsity, long-tailed distributions, imbalanced feature ranges, and non-orthogonality.

1.3.1.1 Sparsity

We explored the sparsity of the dataset. Sparsity is the distribution of null, zero or missing values. We expected the dataset to be highly sparse because CrunchBase is crowd-sourced. Figure 1.11 displays the distribution of features

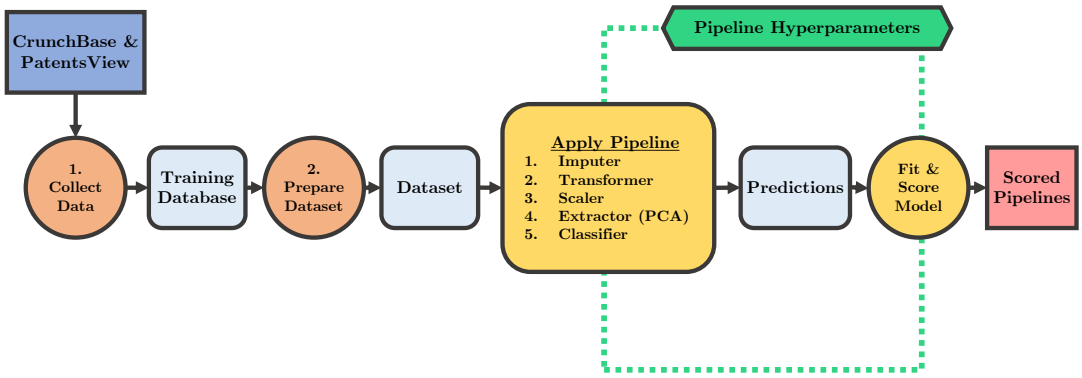
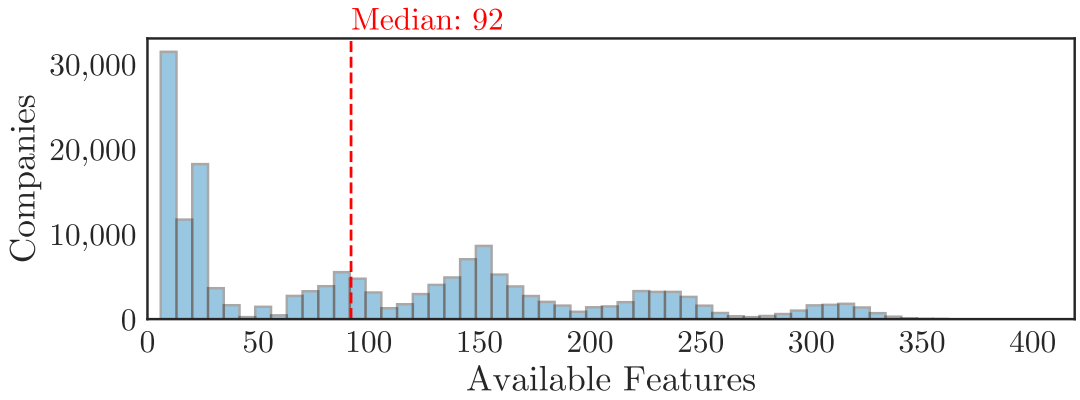


Figure 1.10: Pipeline creation overview. Grid search is performed across the pipeline hyperparameters to generate a variety of scored, candidate pipelines. Legend: dark blue square = input, orange circle = system component, yellow circle = process, light blue rounded square = intermediate, red square = output, green hexagon: iterative process / search space.

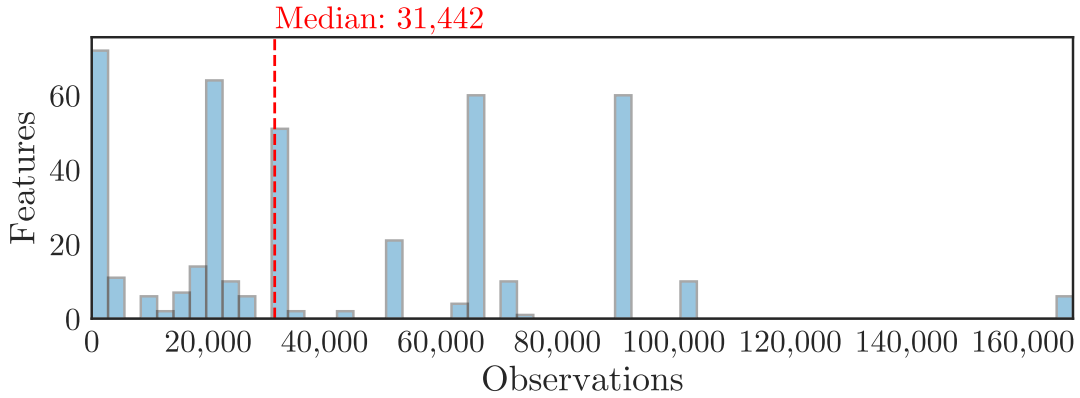
and observations in the dataset, with respect to each other. In Figure 1.11a we observe that many companies in the dataset have few available features (less than 50) and almost no companies have full feature sets. In Figure 1.11b we observe that few features have recorded observations for a large number of companies. The multi-modal peaks of both figures suggest that the availability of some features are linked to each other.

1.3.1.2 Normality

Next, we explored the normality of the dataset. Figure 1.12 shows the skewness and kurtosis of the features in our dataset. Skewness is is a measure of symmetry, or more precisely, the lack of symmetry. A feature is considered highly asymmetrical if its absolute skewness is above 1 [4]. Most of our features are more skewed than this cut-off. Kurtosis is a measure of the distribution of variance in a feature, the extent to which it is long-tailed. We use Fisher's measure of kurtosis, which has a normal value of 0. Our kurtosis distribution suggests we have many extreme values (outliers) in our dataset. In combination, these results suggest that most features in our dataset are not normally distributed, but rather are positively-skewed, long-tailed distributions.



(a) Distribution of available features by company.



(b) Distribution of available observations by feature.

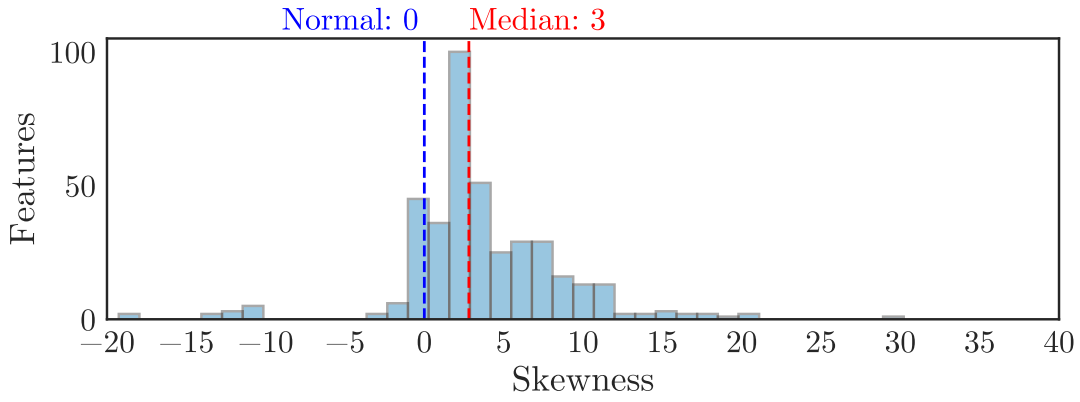
Figure 1.11: Density of features in our dataset.

1.3.1.3 Scale

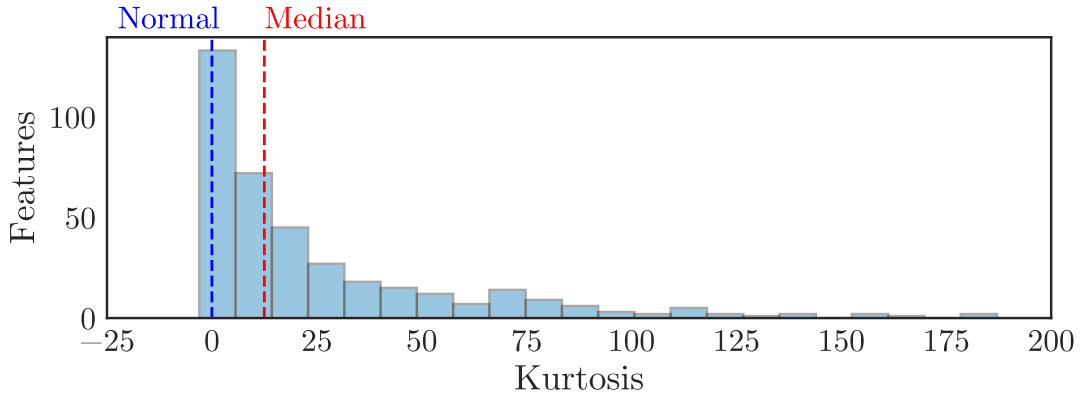
Next, we explored the scaling and range of each of our features. Figure 1.13 shows the Interquartile Range (IQR) of each feature. The distribution is extremely skewed in its original domain so we perform a log transformation to make the distribution easier to observe. Even the log-transformed distribution is highly skewed, which shows that our features have a wide range of magnitudes.

1.3.1.4 Orthogonality

Finally, we explored the orthogonality of our features: the extent to which the variance of our features is unrelated. We explored the distribution of pair-wise inter-correlations between our features, as depicted in Figure 1.14. We use two



(a)



(b)

Figure 1.12: Normality of features in our dataset.

correlation metrics: Pearson and Spearman. Pearson is more commonly used but Spearman is a ranked metric and may more accurately reflect our non-normal feature distributions [**CITE**]. Although most features have low inter-correlations ($\tilde{6}0\%$ below 0.2) there are still some that are highly correlated, so it might be efficient to remove these features using unsupervised feature extraction.

1.3.2 Hyperparameter Evaluation

To address the issues identified in the previous section, we developed a classification pipeline using the popular Python-based machine learning library Scikit-learn [3]. The classification pipeline construct allows us to search across hyperparameters at each step in the pipeline (e.g. imputation strategy, number of

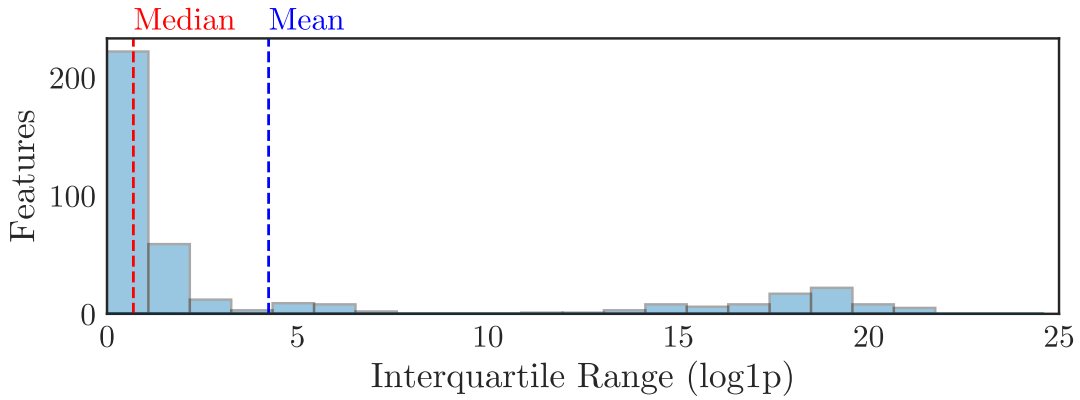


Figure 1.13: Distribution of interquartile ranges (log-transformed) in our dataset.

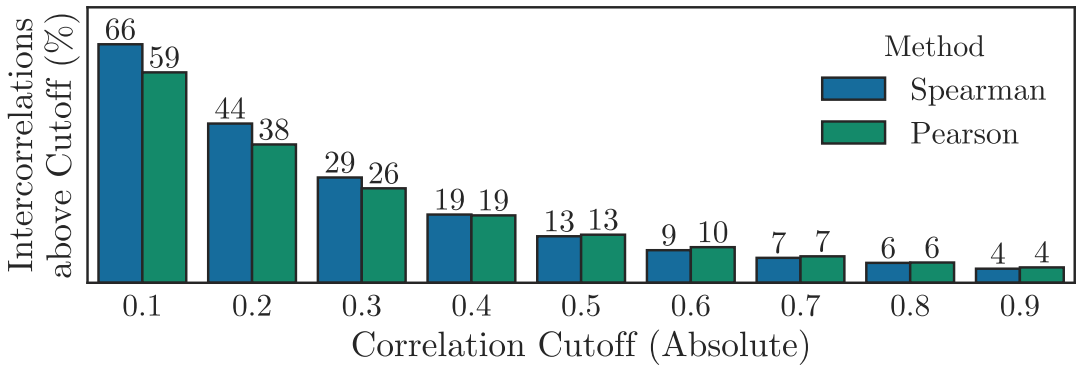


Figure 1.14: Distribution of inter-correlations in our dataset.

components extracted in Principal Component Analysis (PCA), see Appendix **??** for the full hyperparameter list). The following section explores the evaluation of the pipeline hyperparameters against an indicative datset generated from our training database. The indicative dataset is composed of a feature vector from April 2012 and an outcome vector from April 2014.

1.3.2.1 Imputation

After reviewing the distribution of missing data, we decided to perform further investigation into imputation methods. Common imputation strategies include replacing missing values with the mean, median or mode of each feature. Figure 1.15 shows the distribution of mean, median and modes for each feature in the dataset. We apply a log-transformation to this figure to make the highly-skewed distribution easier to observe. There is minimal variance between the

mean, median and mode of the features. For the majority of features, all three measures of central tendency are equal to zero. This resolves the issue of distinguishing missing data from negative observations because, following imputation, all of these data points will map to zero. Figure 1.16 shows the performance of the imputation strategies. All three imputation strategies produce similar results.

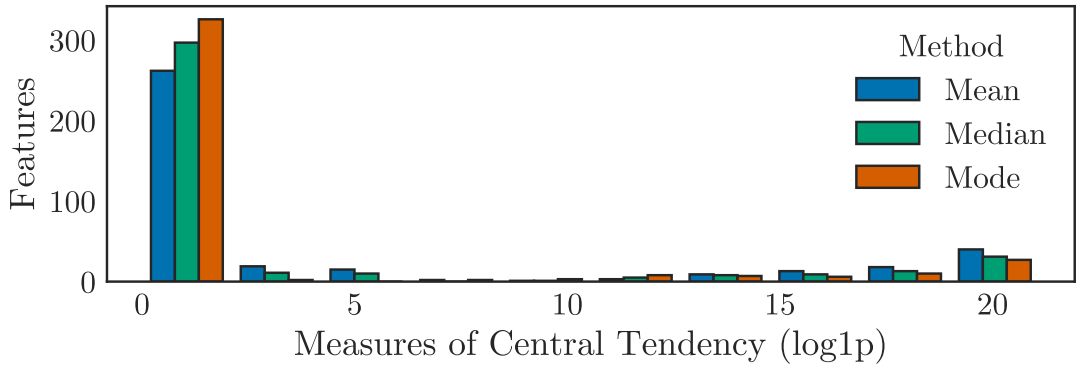


Figure 1.15: Distribution of measures of central tendency (mean, median and mode) in our dataset.

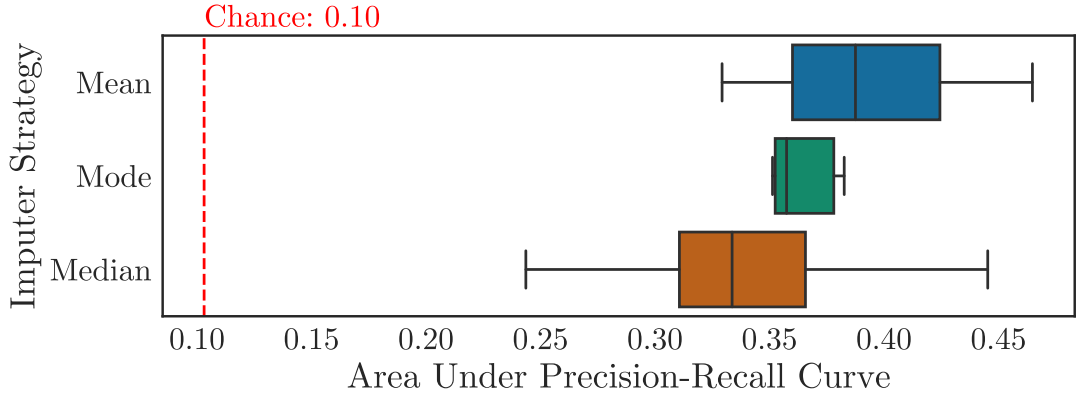


Figure 1.16: Area under Receiver Operating Characteristic (ROC) for different imputation strategies. Imputation strategies include replacing missing values with the most frequent (mode), median and mean value of each respective feature.

1.3.2.2 Transformation

While the classification algorithms we identified in the previous chapter are robust to violations of normality, it may be beneficial to transform the data if the

feature distributions are extreme. Figure 1.17 shows one of the key features, Total Funding Raised, under different transformations. Like many features in our dataset, the distribution of Total Funding Raised is highly skewed. The log transformation reduces this skewness and square root transformation also reduces this skewness (to a lesser extent). Figure 1.18 shows the performance of these transformation functions. Both functions provide a small improvement, with square root narrowly best.

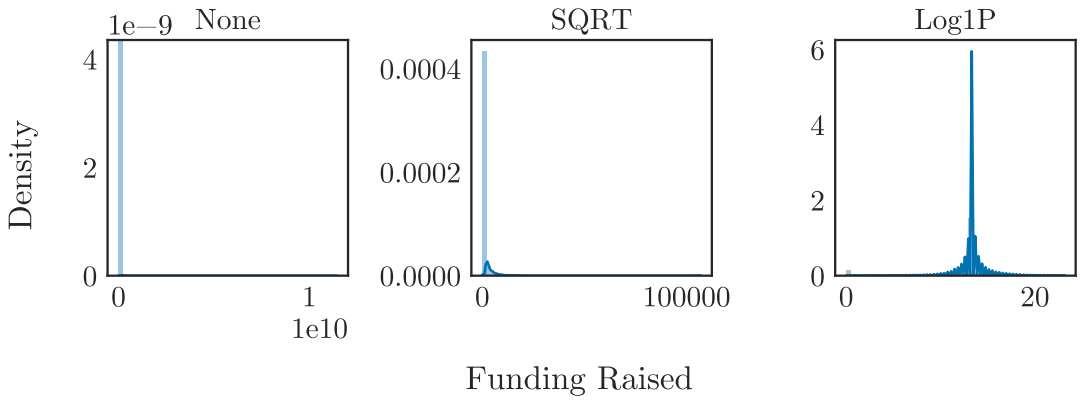


Figure 1.17: Funding raised transformed by functions.

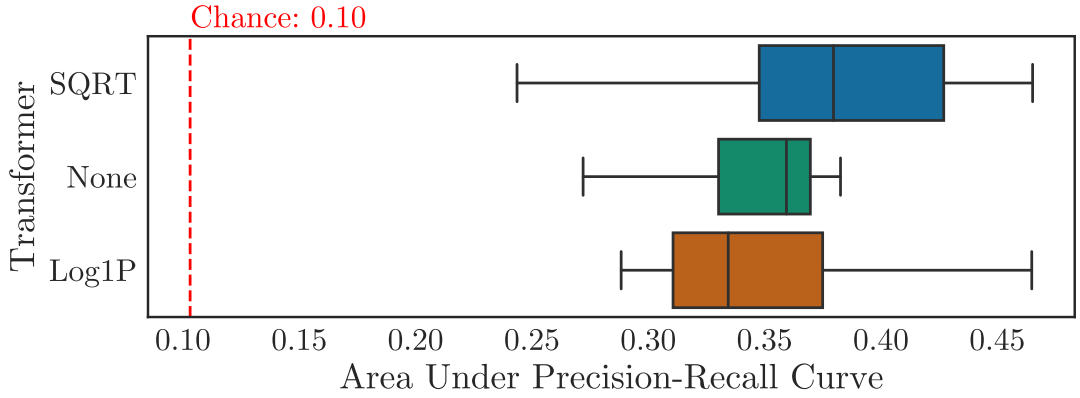


Figure 1.18: Area under ROC for different transformation functions. Transformations include: None (identity transformation), Log1p (natural logarithm of one plus the input array, element-wise), and SQRT (the square root of the input array, element-wise).

1.3.2.3 Scaling

Standardisation of datasets is a common requirement for many feature extraction methods and machine learning estimators. Sci-kit learn provides three primary scaling functions: StandardScaler, RobustScaler and MinMaxScaler. RobustScaler is intended to alleviate the effect of outliers while MinMaxScaler is intended to preserve zero entries in sparse data - both of these are relevant properties for the dataset. Figure 1.19 shows the performance of these scaling functions. None of the scaling functions outperform the null condition. This may be caused by previously applied transformations.

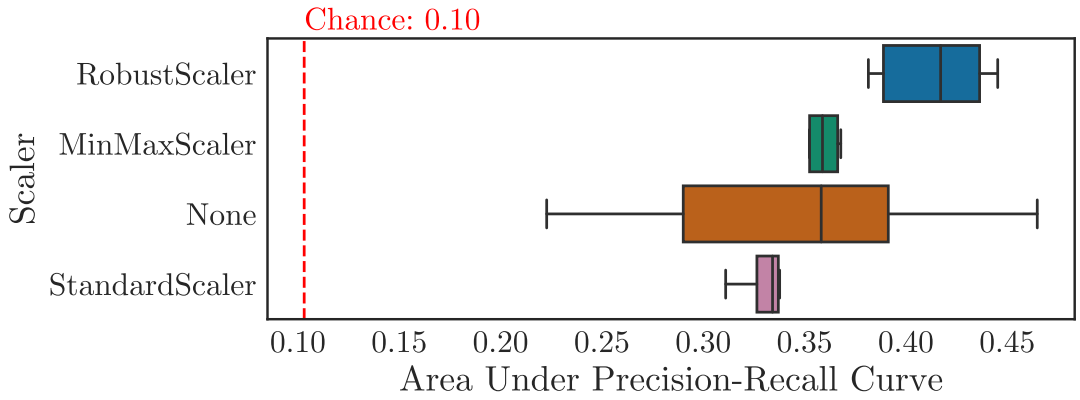


Figure 1.19: Area under ROC for different scaling functions. Scaling functions include: None, StandardScaler (mean: 0, variance: 1), RobustScaler (median: 0, IQR: 1) and MinMaxScaler (min: 0, max: 1).

1.3.2.4 Extraction

Feature extraction reduces high-dimensional data into lower-dimensional data in such a way that maximises the variance of the data. The most common approach to dimensionality reduction is PCA. PCA is a technique which takes a set of vectors and finds an uncorrelated coordinate system in which to represent these vectors [5]. The magnitude of each eigenvector (its eigenvalue) is displayed in Figure 1.20. The majority of explained variance is captured in the first 10 components, and the eigenvalues drop below 1 by 100 components – this suggests that these are reasonable values for further hyperparameter search. Figure 1.21 shows the ROC for different numbers of extracted components. All curves produce similar classification results (within margin of error) which implies that we should extract between 1-20 components because it will provide us with more efficient computation.

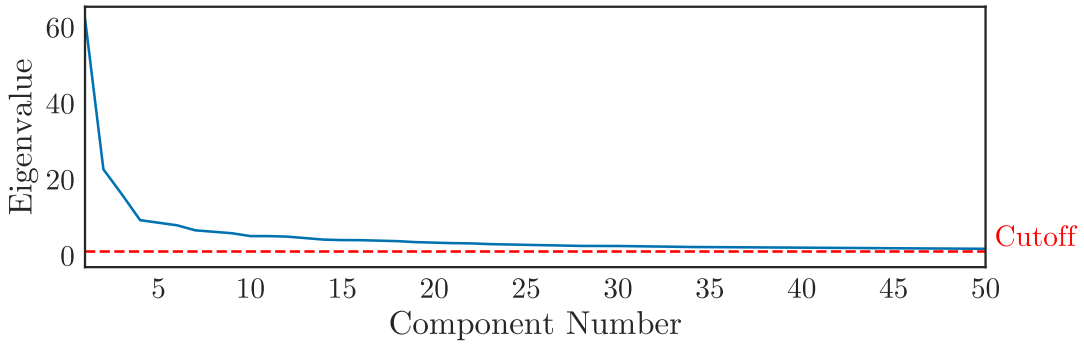


Figure 1.20: Eigenvalues extracted from PCA model. Horizontal line drawn at an Eigenvalue of 1 – this theoretically represents the 'contribution' of one original feature and is commonly used as an approximate threshold for included components.

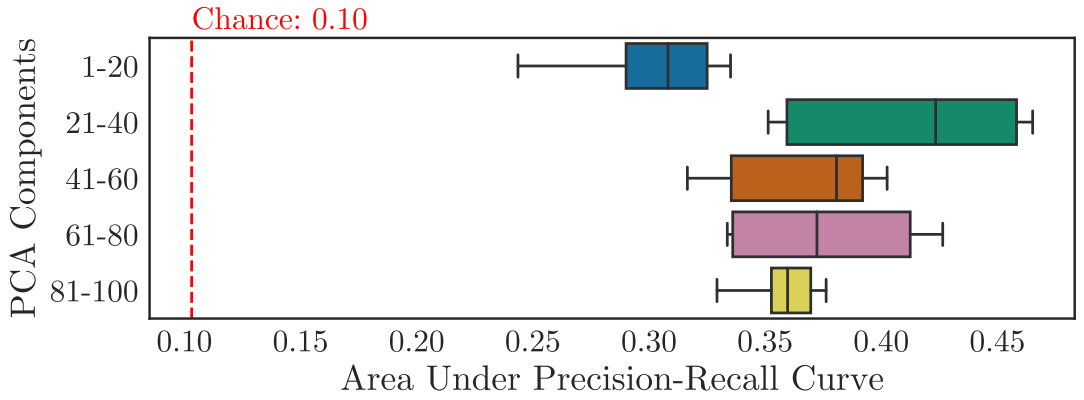


Figure 1.21: Area under ROC for different number of extracted components from PCA. Curves have been grouped by the quotient of the number of components divided by 20 to result in five ordered groups (e.g. Range [0, 19] becomes 0).

While PCA is efficient at reducing features, the resultant components are not interpretable. However, individual analysis of 400+ features is also difficult to interpret. A compromise is to group features using our conceptual framework. We apply an approach that weights each feature to maximise the inter-correlations within each group. We use Spearman Correlation which is robust to skewness [**CITE**]. Figure 1.22 displays the inter-correlations between each grouped factor. Overall, there is little correlation between these grouped factors. This is promising because it implies that each group is providing unique information. There are some minor correlations: Advisors are somewhat correlated with Founders and Investors, and Economic factors are somewhat correlated with Executives

and Funding factors.

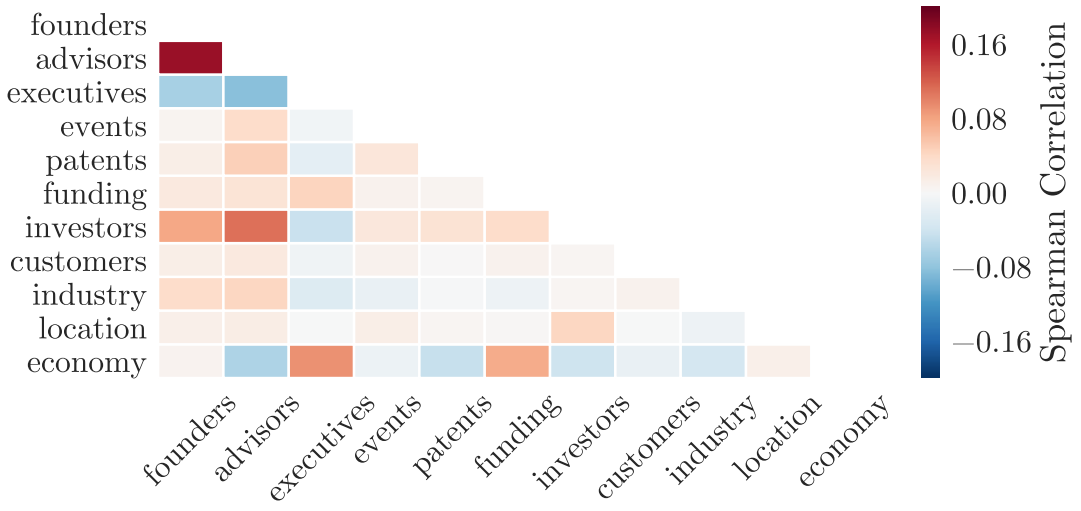


Figure 1.22: Inter-correlations of each factor from conceptual framework. Spearman ranking correlation is used. Individual features are grouped by applying weights that maximise the inter-correlations within each group from our conceptual framework (see Figure 1.2).

1.3.2.5 Classification Algorithms

The literature review we performed in the previous chapter identified common supervised classification algorithms potentially suitable for application to Venture Capital (VC) investment screening. Our review suggested that Random Forests were most likely to provide a successful trade-off between predictive power, interpretability and time taken. We empirically tested each of these classifiers and compared their performance against a range of metrics, as displayed in Table 1.2. We report maximum and median scores to not penalise algorithms that have unfavourable hyperparameter search spaces.

We take a closer look at the Precision-Recall (PR) curves for each classifier in Figure 1.23. While all classifiers perform better than chance, Logistic Regressions and Random Forests come out ahead, and Support Vector Machines and Artificial Neural Networks appear to under-perform. Delving into the cross-validated learning curves for each classifier (Figure 1.24) we see that Naive Bayes, Logistic Regression, Artificial Neural Networks and Support Vector Machines quickly converge, whereas Decision Trees, Random Forests and K-Nearest Neighbours require more observations to converge. This suggests that we might expect Ran-

	AUC PRC		AUC ROC		F1		MCC		Fit Time (s)	
Classifier	Median	Max	Median	Max	Median	Max	Median	Max	Median	75th
LR	0.417	0.465	0.675	0.710	0.339	0.358	0.255	0.288	7.3	412.7
RF	0.376	0.465	0.619	0.709	0.332	0.360	0.271	0.288	68.3	69.0
DT	0.388	0.429	0.651	0.659	0.305	0.314	0.212	0.224	15.3	16.8
NB	0.354	0.367	0.623	0.638	0.303	0.321	0.212	0.239	8.6	26.8
KNN	0.335	0.353	0.532	0.565	0.131	0.226	0.137	0.210	8.5	20.8
ANN	0.320	0.335	0.517	0.523	0.072	0.096	0.111	0.140	9.1	21.0
SVM	0.233	0.244	0.503	0.504	0.014	0.017	0.038	0.045	29.0	29.0
Total	0.357	0.465	0.623	0.710	0.300	0.360	0.209	0.288	15.3	29.0

Table 1.2: Overview of classification algorithm performance. Algorithms are: NB = Naive Bayes, LR = Logistic Regression, KNN = K-Nearest Neighbours, DT = Decision Trees, RF = Random Forests, SVM = Support Vector Machines, ANN = Artificial Neural Networks.

dom Forests to do better in final testing (as testing will not be cross-validated), as well as in the future as the dataset naturally grows.

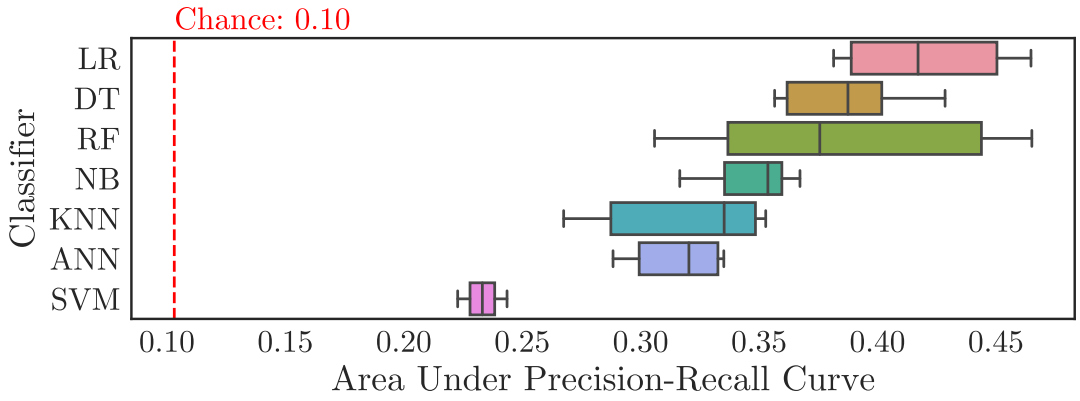


Figure 1.23: Area under ROC for different classification algorithms. All algorithms are implementations from the Sci-kit learn library. Algorithms are: NB = Naive Bayes, LR = Logistic Regression, KNN = K-Nearest Neighbours, DT = Decision Trees, RF = Random Forests, SVM = Support Vector Machines, ANN = Artificial Neural Networks.

1.4 Pipeline Selection

In this step, we evaluate the best pipelines from the previous step overdifferent dataset slices. This process, depicted in Figure 1.25, ensures our final pipeline

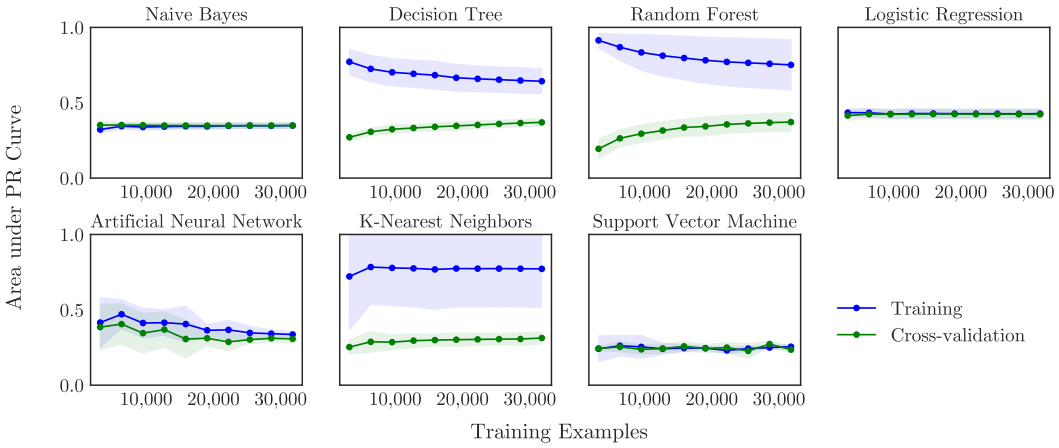


Figure 1.24: Learning curves by classification algorithms.

is robust in its performance over time. We aggregate the results for each finalist pipeline across these dataset slices and rank the finalist pipelines on their overall performance. Finally, we select the best pipeline.

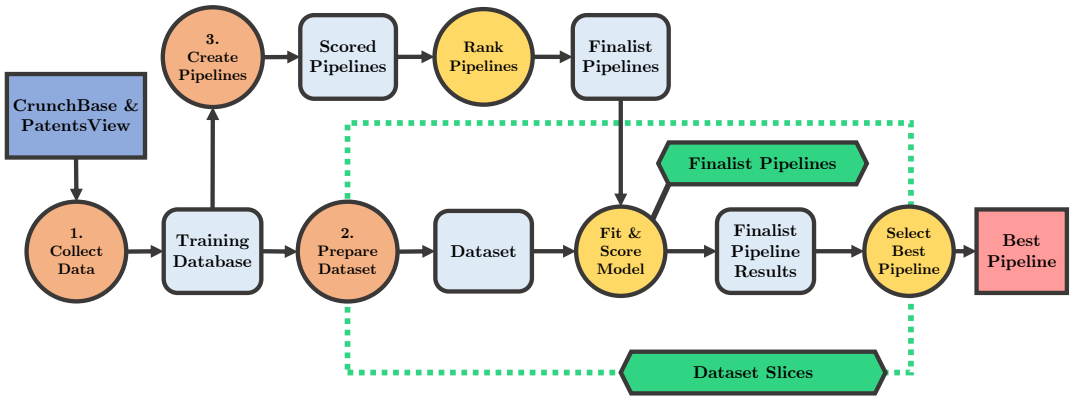


Figure 1.25: Pipeline selection overview. Legend: dark blue square = input, orange circle = system component, yellow circle = process, light blue rounded square = intermediate, red square = output, green hexagon: iterative process / search space.

1.4.1 Evaluation Metrics

Next, we decided how to select finalist pipelines that we can evaluate further. There are a variety of metrics used to evaluate binary classification algorithms. Accuracy is rarely used in practice because it gives misleading results in the

case of imbalanced classes. Receiver Operating Characteristic (ROC) curves are commonly used, and show how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. The area under these curves gives a standardised result across a spectrum of decision thresholds. Precision-Recall (PR) curves are similar to ROC curves but instead map the trade-offs between precision and recall. They are less commonly used than ROC curves but have been shown to produce more accurate results for imbalanced classes than ROC curves [6]. Given our dataset is highly imbalanced, we decided to proceed with PR curves. We will also use this metric to rank our finalist pipelines.

1.4.2 Finalist Pipeline Evaluation

Our hypothesis is that the performance of our pipelines may vary with respect to the date our datasets were collected. To evaluate this hypothesis, first we explored variance between the pipelines on aggregate against the slice dates, presented in Figure 1.26. There is no relationship observed between slice date and score.

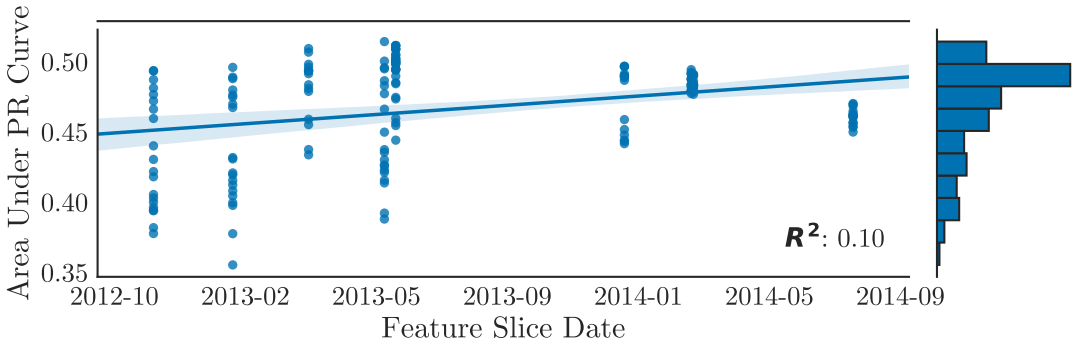


Figure 1.26: Pipeline performance by slice date.

Next, we study the variance within the individual pipelines, presented in Figure 1.27. Although there is a positive correlation between the pipelines initial ranking and their scores, there are deviations. For example, the top-ranked pipeline from the first stage has a lower median score than the second-ranked pipeline. These results suggest that the top 3-5 pipelines should be evaluated in this manner to ensure that the best pipeline is selected. The chosen pipeline is depicted in Table **??**. We adopted this pipeline for our following experiments.

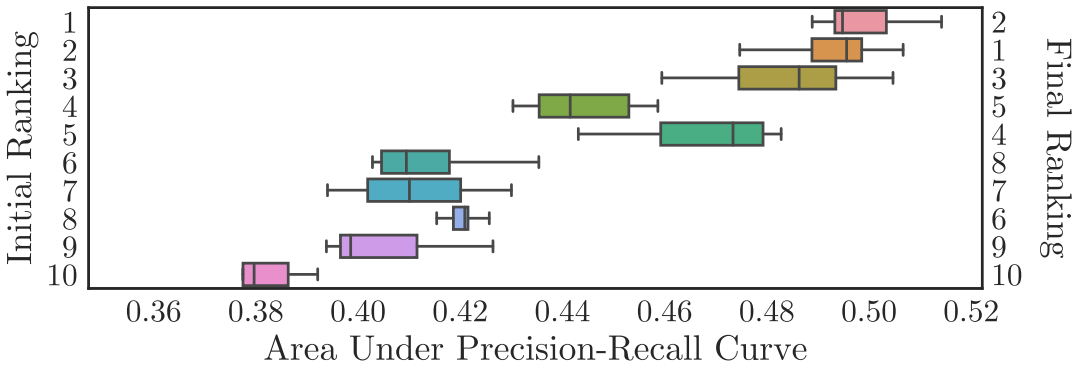


Figure 1.27: Overview of finalist pipeline performance.

1.5 Model Fit and Prediction

Finally, our system applies the best classification pipeline to a feature vector from a held-out test database to generate a model and a set of predictions, as shown in Figure 1.28. We evaluate the accuracy of the models produced by our system in the next chapter.

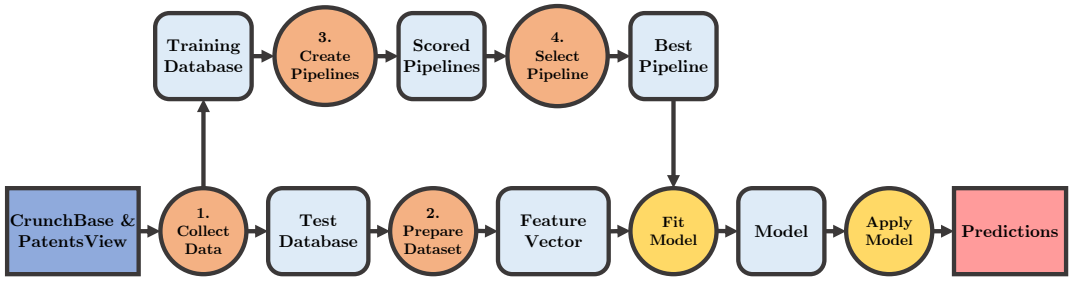


Figure 1.28: Model fit and prediction overview. Legend: dark blue square = input, orange circle = system component, yellow circle = process, light blue rounded square = intermediate, red square = output.

Bibliography

[1] Ahlers, G. K., et al. Signaling in equity crowdfunding. *Entrepreneurship Theory and Practice* 39, 4 (2015), 955–980.

[2] Baum, J. A., and Silverman, B. S. Picking winners or building them? Alliance, intellectual, and human capital as selection criteria in venture financing and performance of biotechnology startups. *Journal of Business Venturing* 19, 3 (2004), 411–436.

[3] Pedregosa, F., et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[4] Bulmer, M. G. *Principles of statistics*. Courier Corporation, 1979.

[5] Jolliffe, I. *Principal component analysis*. Wiley Online Library, 2002.

[6] Davis, J., and Goadrich, M. The relationship between Precision-Recall and ROC curves. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, 233–240.